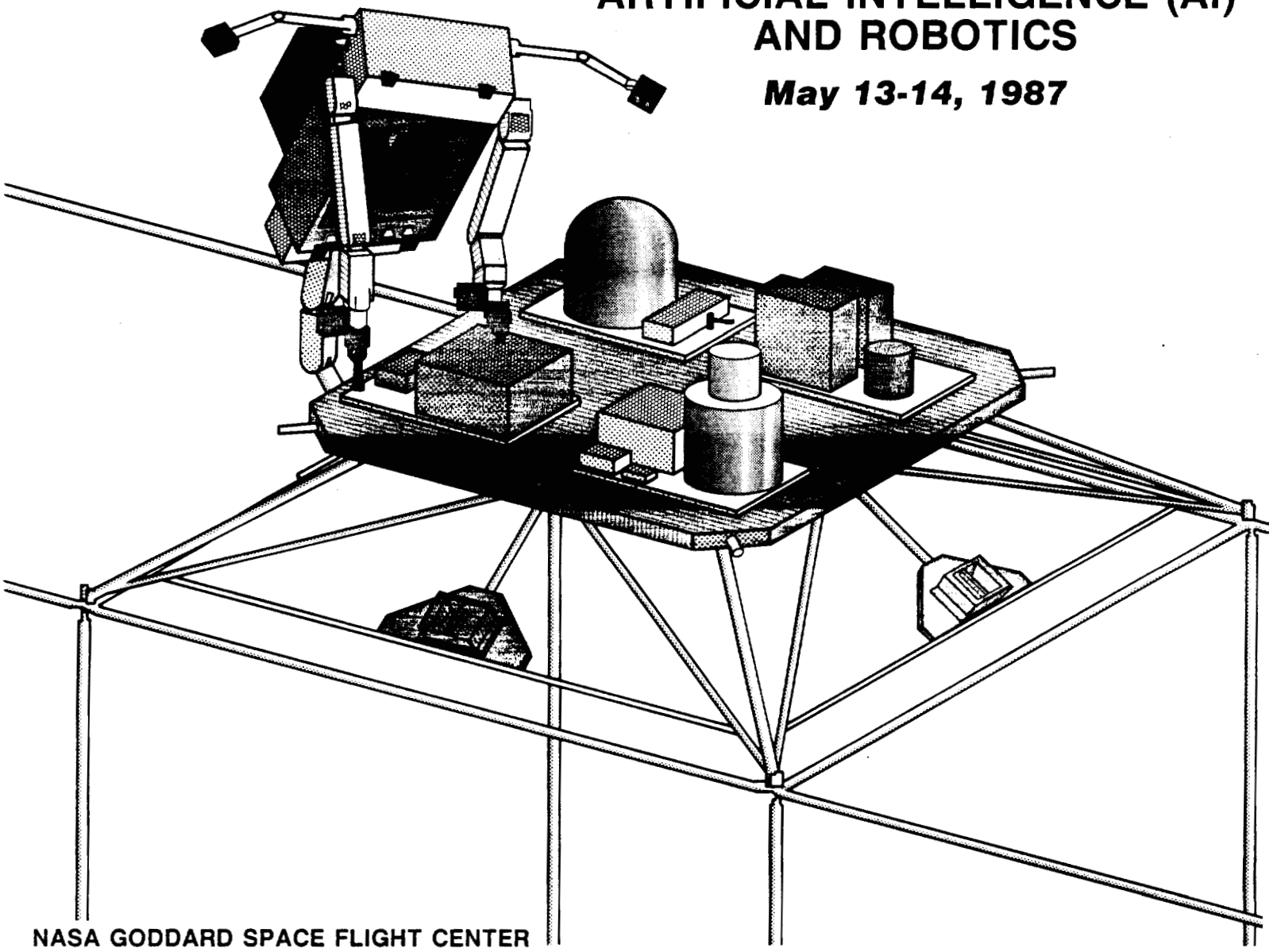
PROCEEDINGS OF

1987 GODDARD CONFERENCE on SPACE APPLICATIONS OF ARTIFICIAL INTELLIGENCE (AI) AND ROBOTICS

May 13-14, 1987



NASA GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND

OPENING REMARKS

William Macoughtry, Assistant Head
Spacecraft Control Programs Branch, Goddard Space Flight Center

Welcome to the 1987 Goddard Conference on Space Applications of Artificial Intelligence and Robotics. That long and rather bureaucratic sounding title was discussed at some length by the Conference steering committee and was chosen to try to convey the essence of what we want this conference to be.

First, the word "Goddard" was used, not only because the conference is obviously at Goddard Space Flight Center, and sponsored by Goddard, but also because we want the people of Goddard to benefit from the conference, by receiving information from the aerospace community, and by letting the community know what we are doing here at Goddard in these two important fields. We focused on "space" because while the interesting and innovative uses of AI and Robotics are expanding throughout many fields, we felt a need to narrow our attention to those uses which contribute directly to the work we are doing here at Goddard.

The word "applications" was included because this Conference started last year as a grass-roots conference, by those of us who are primarily involved with applying the ever-more powerful tools which AI and expert systems give us to meet the challenges of today's and tomorrow's space program, rather than doing basic research in the theoretical foundations of the field. We feel that there are other appropriate opportunities to learn about that aspect of the art.

While last year's conference was devoted solely to expert systems, we have a two day conference this year to examine both expert systems and robotics, as there is a symbiotic relationship between these two fields of study and because Goddard's responsibilities toward the Flight Telerobotic Servicer for the Space Station makes robotics particularly of interest at this time and place. Our co-chairman, Lloyd Purves, will give you more information on the subject of robotics tomorrow.

Our major problem in preparing for this year's conference has been coping with the high level of interest in attending, and the lack of facilities here at Goddard which could house in one room all those who would like to have participated. We especially want to welcome those of you viewing the conference in Building 26 and at other locations around the Center by closed-circuit television, and thank you for demonstrating your committment to the conference by participating in this way. We wish to invite you to view the software demonstrations and poster sessions in the building 8 venue at the times indicated on your agenda.

We would welcome any suggestions from the attendees concerning ways to improve this conference in the future. If you wish to, please leave a note with the Birch and Davis coordinators at the

registration areas.

Finally, I would like to thank our conference committee members, whom you will find listed in your agenda package, for contributing ideas and labor to make the conference a reality, and Goddard management for backing and encouraging our efforts. Not least, I wish to recognize the contributions of our supporting contractors, Bendix Field Engineering Corporation and ORI Incorporated, who have supported the conference both monetarily and by their diligence, and of Birch and Davis, who have provided conference coordination services.

Now, I would like to introduce our keynote speaker. Mr. John Quann, Deputy Director of Goddard Space Flight Center, has a distinguished career developing and managing science and applications uses of space technology. He formed and headed the Information Extraction Division which developed computational and analytical methods required by remote sensing. Upon direct request from the Executive Office of the President, he led the development of the Decision Information Display System, which produces full color demographic maps of statistical information for use by legislative and executive branches of state, local, and Federal governments. In 1980, he was appointed Director of the Mission and Data Operations Directorate, in which post he served until his appointment as Deputy Director of Goddard Space Flight Center in September, 1982. Among his other responsibilities, he is co-chairman of the 1987 AIAA/NASA International Conference on Information Systems in the Space Station Era, and he is the general chairman of the 1988 AIAA/NASA/DOD Conference on Automation and Robotics.

The 1987 Goddard Conference on Space Applications of Artificial Intelligence (AI) and Robotics was sponsored jointly by the following groups at Goddard Space Flight Center:

Spacecraft Control Programs Branch (Code 514)
Data Systems Technology Division (Code 520)
Telecommunication Systems Branch (Code 531)
Office of Telerobotic Engineering (Code 706)

The conference committee responsible for planning and organizing the conference consisted of:

Willian Macoughtry (co-chairman), Code 514.0
Lloyd Purves (co-chairman), Code 706
Dorothy Perkins, Code 522.1
James Rash, Code 531.1
Carolyn Dent, Code 514.0
Peter Hughes, Code 522.1
Ellen Stolarik, Bendix Field Engineering Corp.
David Beyer, Bendix Field Engineering Corp.
Ronald Littlefield, Bendix Field Engineering Corp.
Beryl Hosack, ORI

These Proceedings were edited and produced as a joint effort by the Bendix Field Engineering Corporation MOSS Software Services and Documentation Services.

TABLE OF CONTENTS

SECTION C: Data Processing/Analysis Expert Systems

SECTION D: Expert System Tools/Techniques

ROBOTICS

An Optimal Resolved Rate Law for Kinematically Redundant Manipulators - B. Bourgeois

An Adaptive Control Scheme for a Flexible Manipulator - T. Yang, J. Yang, and P. Kudva

Robot Sensor Language - S. Leake

Advanced Data Management Design for Autonomous Telerobotic Systems in Space using Spaceborne Symbolic Processors - A. Goforth

Robot Hands and Extravehicular Activity - B. Marcus

Dynamic Task Allocation for a Man-Machine Symbiotic System - L. Parker and F. Pin

NASREM -- Standard Reference Model for Telerobot Control - J. Albus, R. Lumia, and H. McCain

Kinematic Study of Flight Telerobotic Servicer Configuration Issues - R. Lewis, R. Scott, and W. Howard

Actuators for a Space Manipulator - W. Chun and P. Brunson

Cable Applications in Robot Compliant Devices - J. Kerley

Computer Hardware and Software for Robotic Control: The KSC RADL - V. Davis

Teleoperated Position Control of a Puma Robot - E. Austin and C. Fong

Performance Improvement of Robots using a Learning Control Scheme - R. Krishna, P. Chiang, and J. Yang

EXPERT SYSTEM APPLICATIONS

Section A

Planning/Scheduling Expert Systems

Maintaining an Expert System for the Hubble Space Telescope Ground Support

May 13, 1987

Kelly Lindenmayer [1]
Astronomy Programs, Computer Sciences Corporation

Shon Vick
Space Telescope Science Institute [2]
3700 San Martin Drive
Baltimore, MD 21218

and
Don Rosenthal
NASA Ames Research Center

Abstract

The Transformation portion of the HST Proposal Entry Processor System converts an astronomer-oriented description of a scientific observing program into a detailed description of the parameters needed for planning and scheduling. The Transformation system is one of a very few rulebased experts systems that has ever entered an operational phase. The day to day operation of the system and its rulebase are no longer the responsibility of the original developer. As a result, software engineering properties of the rulebased approach become more important. In this paper, we discuss maintenance issues associated with the coupling of rules within a rulebased system and offer a method for partitioning a rulebase so that the amount of knowledge needed to modify the rulebase is minimized. This method is also used to develop a measure of the coupling strength of the rulebase.

[1] Staff Member of the Space Telescope Science Institute

[2] Operated by the Association of Universities for Research in Astronomy for the National Aeronautics and Space Administration

1 Introduction

The Hubble Space Telescope (HST) is an orbiting optical observatory to be launched by the Space Shuttle in late 1988. Using a rulebased expert system written in OPS5, the Transformation system converts an astronomer-oriented description of a scientific observing program into a detailed description of the parameters needed by the planning and scheduling portion of the HST Science Operations Ground Support System.[3] Transformation has been in an operational phase since December 1985. During its early stages of the operational phase, the primary designer and implementer of the rulebased portion of the Transformation System remained responsible for development and maintenance of the rulebase. Eight months later, the system was turned over to a member of the software group who had limited prior exposure to the project. The Transformation system is one of a very small number of rulebased expert systems that has entered an operational phase. Rulebased systems have traditionally been developed as research projects, and have been maintained by their original implementers. As expert system techniques and technolgy mature, a trend towards developing rulebased systems for practical applications will occur. These systems will be developed with the expectation that they will grow and evolve throughout the life of the project. With this evolution, the notion of a software life cycle becomes relevant and the software life cycle that rulebase systems undergo does not correspond to that of conventional software systems.[4] Unlike software problems that are solved by a more algorithmic approach, AI programs tend to implement problems which have not been completely specified. As a result, expert systems raise some interesting software engineering considerations. In this paper, we discuss some of the maintenance issues associated with the coupling of rules for a rulebased system. We present a method for partitioning a rulebase in such a way that the knowledge required to make a modification to the rulebase is minimized. The method is also used to derive a metric that can be used as a measure of coupling strength.

2 Background

In OPS5, like many expert system languages, knowledge is encoded as a set of rules or productions with the following format:

IF antecedents THEN consequents

Facts from a global database, usually termed working memory, are matched with the antecedents (also referred to as left hand side or LHS). The matched rule/facts pairs (instantiations) are put into a conflict set. A conflict resolution strategy is applied to the set to determine the instantiation to be *fired*. During the *firing process*, the consequent prescribed by the rule corresponding to the instantiation is established. In general, this process creates new facts, and the inference cycle proceeds to the match step. This process continues until the conflict set is empty or the process is halted explicitly by the programmer. This paradigm encourages the *opportunistic* behavior of rules and when the situation is right (e.g. when the antecedents match the facts), the rules are instantiated and fired.

[3] Transformation is described in detail in [7]

[4] See [5] for more information

This phenomenon distinguishes rulebase systems from an algorithmic approach to software problems. In other words, the order of application of the program's functional pieces (in this case rules), is not built into the program, and will not be determined until runtime.

On another level, such inferencing systems may seem to be common coupled. In common coupling, functional modules are linked through a global data structure. In OPS5 any rule may potentially read or write any part of working memory since rules share the entire working memory data area. However, working memory is partitioned into elements of data classes (analogous to traditional data structures), and rules refer to working memory elements by class name, and attribute (field) name. Therefore, coupling in rulebased systems more closely resembles stamp coupling. Stamp coupling is similar to common coupling except that the global data items are shared more selectively between components that use them.[5]

The partitioning of groups of rules according to a goal strategy is presently one of the more viable method of controlling inferencing in OPS5 (compared to, for example running several smaller ruebased programs in sequence). There is no way for the developer to "program" the conflict resolution strategy, she may only choose between MEA and LEX.[6] Meta rules are not presently possible due to the strict partitioning of rule memory, working memory, and the conflict set.[7]

The Transformation rulebase is a goal oriented rulebase which consists of seven goals, some of which contain sub goals and task lists, but all control is imposed at a very high level. The rulebase is "partitioned" into goal sets which reside in files; that is, rules pertaining to a specific goal or subgoal are found in a single file. When writing rules, the natural tendency is to group rules according to their functionality. For instance, a knowledge engineer might group in a file all rules that merge exposures. From the maintainers standpoint, this method has its disadvantages as well as advantages. The most obvious advantage is that if a new rule to merge exposures needs to be added, there are several examples for a maintainer to follow. In addition, if a problem arises in how exposures are merged, the problem area may be localized, and hence easier to debug. The disadvantage lies in what occurs downstream from this problem. For instance, if a new rule is added to merge exposures, what else might that change affect? This is a formidable problem for a maintenance programmer who is not well versed in the structure inherent in the rulebase.

Other AI developers [8] have suggested an automated approach to partitioning a rule set based on a measure of "relatedness". Rules would be grouped according to the facts that they share - where a fact is some data representation that if changed in one rule would affect another rule in some way. Since there are several ways in which two rules could reference the same fact, the facts would be weighted, and the measure of "relatedness" would be based on these facts. The rulebase would then be partitioned into groups according to how strongly rules were related. Although these arguments and the related tools have merit, there are still some issues that need to be addressed. For instance, choosing a measure of relatedness is still a rather arbitrary process, although a more sophisticated clustering algorithm to partition the rulebase might be helpful. In addition, within a partitioned rulebase, there

[5] See [4] and [2] for a complete discussion of common and stamp coupling

[6] Refer to [1] for an explanation of MEA and LEX

[7] See [6] for a discussion of meta rules

[8] See [3] for details

may be subtle interactions between rules within a set of grouped rules or between the groups themselves. Moreover, the grouping procedures do not guarantee a small number of groups. What is needed then, is a method for maintenance programmers to understand the coupling of rules in general.

3 A Model for Partitioning a Rulebase

We first take a look at the possible ways in which rules may be coupled. We know that rules are related to one another through the facts that they share. In OPS5 there are essentially three different ways of altering facts in working memory: through a make, modify or remove. A *make* creates a new working memory element. A *remove* deletes a working memory element from the database. Conceptually, a *modify* is a combination of a make and a remove. It deletes a working memory element, and replaces it with the appropriate new working memory element.

Below is an example of two rule which are coupled by a *make* action. (All of the following examples are actual rules from the Transformation rulebase. For the purposes of understanding how rules are coupled, it is not necessary to comprehend the semantic content of the rule.) The first rule, *Find-parallel-with-mergeable-exposures* uses the make command to create a working memory element with class name *mergeable-exposures*. Since the specified attributes of this working memory element "match" a subset of the antecedent of rule *Remove-mergeable-exposures-if-same-mode-diff-aperture*, we consider these rules to be related.

```
(p find-parallel-with-mergeable-exposures

   (goal
       ^has-name        merge-exposures
       ^has-status      active
       ^task-list       find-potential-exposure-merges)

   (exposure-link
       ^has-exposure-number          <parallel-exposure>
       ^is-linked-to                 <primary-exposure>
       ^has-link-type                parallel-with )

   (mergeable-level
       ^symbol                       parallel-with
       ^value                        <parallel-with-level>)
-->

(make  mergeable-exposures
       ^first-exposure-number        <primary-exposure>
       ^second-exposure-number       <parallel-exposure>
       ^is-unmergeable               false
       ^is-mergeable-level           <parallel-with-level>
       ^merge-type                   parallel-with
       ^has-unique-label             (genatom) ) )
```

```
(p Remove-mergeable-exposures-if-same-mode-diff-aperture

   (goal
       ^has-name          merge-exposures
       ^has-status        active
       ^task-list         find-potential-exposure-merges)

   (exposure-specification
       ^is-internal-target-type      <> I
       ^has-exposure-number          <first-exposure>
       ^first-aperture-used          <aperture>

   (mergeable-level
       ^symbol                       parallel-with
       ^value                        <parallel-with-level> )

   {<mergeable-exposure-link>
   (mergeable-exposures
       ^first-exposure-number        <first-exposure>
       ^is-mergeable-level           <parallel-with-level>
       ^second-exposure-number       <second-exposure> ) }

   (exposure-specification
       ^has-exposure-number          <second-exposure>
       ^is-internal-target-type      <> I
       ^first-aperture-used          <> <aperture>
-->

   (modify <mergeable-exposure-link>
       ^1 unmerged-exposures))
```

When discussing relatedness, it is important to keep in mind that this is a static analysis of the problem. Because rulebased systems are data driven, rules will interact differently with different sets of data. A static approach to the problem is simply, what portion of the rulebase might *possibly* be affected by a particular rule change, given any set of data.

Figure 1 gives a pictorial representation of the relationship between these two rules. It shows that if *Find-parallel-with-mergeable-exposures*were modified, there is a *possibility* that *Remove-mergeable-exposures-if-same-mode-diff-aperture* might be affected by the change.

Figure 1.0 *Coupling Rules through a make action*

Next we look at two rules that are coupled by a *remove* action. The rule, *Remove-mergeable-exposures-if-first-is-an-acquisition* removes a working memory element that may have been created by *Find-parallel-with-mergeable-exposures*:

```
(p find-parallel-with-mergeable-exposures

   (goal
       ^has-name        merge-exposures
       ^has-status      active
       ^task-list       find-potential-exposure-merges)

   (exposure-link
       ^has-exposure-number          <parallel-exposure>
       ^is-linked-to                 <primary-exposure>
       ^has-link-type                parallel-with )

   (mergeable-level
       ^symbol                       parallel-with
       ^value                        <parallel-with-level>)
-->

(make  mergeable-exposures
       ^first-exposure-number        <primary-exposure>
       ^second-exposure-number       <parallel-exposure>
       ^is-unmergeable               false
       ^is-mergeable-level           <parallel-with-level>
       ^merge-type                   parallel-with
       ^has-unique-label             (genatom) ) )



(p Remove-mergeable-exposures-if-first-is-an-acquisition)

   (goal
       ^has-name          merge-exposures
       ^has-status        active
       ^task-list         find-potential-exposure-merges)

  {<mergeable-exposure-link>
   (mergeable-exposures
       ^first-exposure-number       <first-exposure> ) }

   (exposure-link
       ^has-exposure-number         <first-exposure>
       ^has-link-type               <<ONBOARD-ACQ INT-ACQ > )
-->

(remove <mergeable-exposure-link>) )
```

We will represent the relationship of two rules coupled by a *remove* as is shown in figure 2.0. Note that we picture the relationship slightly different for a *remove* coupling than we did for a *make* coupling. In the case of a *remove* , the arrow is pointing in the

opposite direction to show that in order for *Find-parallel-with-mergeable-exposures* to be affected by a modification of the rule *Remove-mergeable-exposures-if-first-is-an-acquisition*, *Find-parallel-with-mergeable-exposures* must already have been instantiated based on this working memory element.

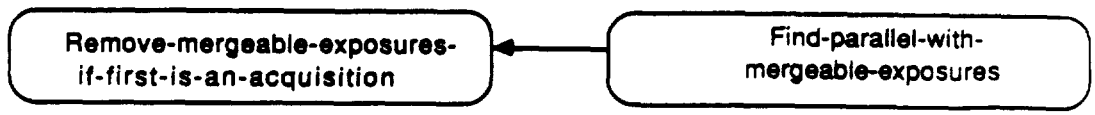


Figure 2.0 *Coupling Rules through a remove action*

In order to demonstrate the coupling created by a *modify*, it is necessary to use three rules. The first rule *Remove-fos-hrs-merge-if-only-second-mode-is-rapid* uses the *modify* action to change the class name of *mergeable-exposures* to *unmerged-exposures*. Since the specified attributes of the working memory element with class name *unmerged-exposures* "matches" the antecedent of the rule *link-alignments-with-unmerged-exposures*, these rules are said to be coupled. On the other hand, when *Remove-fos-hrs-merge-if-only-mode-is-rapid* uses the *modify* action to change the class name of a working memory element it is in affect removing the working memory element *mergeable-exposures*. So, based on the same principle as the *remove* example, we consider these two rules also to be coupled.

```
(p Find-same-orientation-mergeable-exposures

   (goal
      ^has-name            merge-exposures
      ^has-status          active
      ^task-list           find-potential-exposure-merges )

   (exposure-link
      ^has-exposure-number          <linked-exposure>
      ^is-linked-to                 <main-exposure>
      ^has-link-type                SAME-ORIENT )

   (exposure-specification
      ^has-exposure-number          <main-exposure>
      ^uses-SI-configuration        <SI-configuration> )

   (exposure-specification
      ^has-exposure-number          <linked-exposure>
```

```
        ^uses-SI-configuration          <SI-configuration> )

    (mergeable-level
        ^symbol                         same-orientation
        ^value                          <same-orientation-level>)
-->

(make   mergeable-exposures
        ^first-exposure-number          <main-exposure>
        ^second-exposure-number         <linked-exposure>
        ^is-unmergeable                 true
        ^is-mergeable-level             <same-orientation-level>
        ^merge-type                     same-orientation
        ^has-unique-label               (genatom) ) )


(p Remove-fos-hrs-merge-if-only-second-mode-is-rapid

    (goal
        ^has-name           merge-exposures
        ^has-status         active
        ^task-list          find-potential-exposure-merges )

    {<mergeable-exposure-link>
     (mergeable-exposures
        ^first-exposure-number          <first-exposure>
        ^second-exposure-number         <second-exposure>
        ^merge-type                     <<sequence-no-gap consecutive
                                          same-orientation>>
        ^is-unmergeable                 true) }

     (exposure-specification
        ^has-exposure-number            <first-exposure>
        ^uses-SI-configuration          << fos/bl fos/rd hrs >>
        ^uses-SI-operating-mode         <> rapid )

    (exposure-specification
        ^has-exposure-number            <second-exposure>
        ^uses-SI-operating-mode         rapid )
-->

(modify <mergeable-exposure-link>
        ^1          unmerged-exposures ))



(p link-alignments-with-unmerged-exposures

    (goal
        ^has-name merge-alignments
        ^has-status active
        ^task-list find-potential-alignment-merges)
```

```
    (unmerged-exposures
        ^first-exposure-number <first-exposure-number>
        ^first-copy-number <first-copy>
        ^second-exposure-number <second-exposure-number>
        ^second-copy-number <second-copy> )

    (assignment-record
        ^has-Pepsi-exposure-number <first-exposure-number>
        ^is-assignment-record-copy-number <first-copy>
        ^has-alignment-order <first-alignment-order>
                                                              )

    (assignment-record
        ^has-Pepsi-exposure-number <second-exposure-number>
        ^is-assignment-record-copy-number <second-copy>
        ^has-alignment-order { <second-alignment-order> <>
          <first-alignment-order> }
                                                                  )
-->

(make mergeable-alignments
        ^has-first-alignment-order <first-alignment-order>
        ^has-second-alignment-order <second-alignment-order>
        ^has-unique-label (genatom) ) )
```

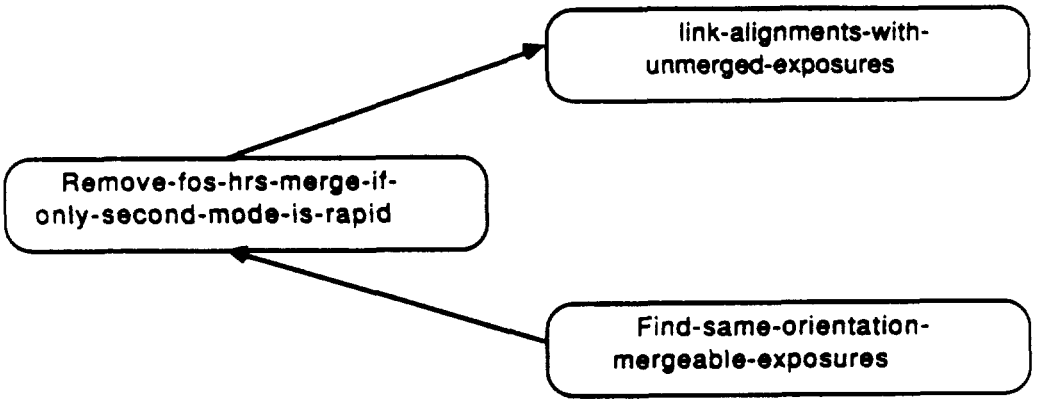


Figure 3.0 *Coupling Rules through a modify action*

4 A More Formal Way to Express Coupling

What we have seen so far is that if two rules are coupled by our definition, the **rhs** of one rule *feeds* the **lhs** of another rule. But what is actually meant by one rule *feeding* another? Let's introduce a notation to make the notion precise.

We can think of a rule as consisting of three parts: its name, its associated conditions, and its actions. A rule j is denoted by R_j.

$$R_j((c_{j,1} \ldots c_{j,n}) \rightarrow (a_{j,1} \ldots a_{j,m}))$$

The first condition of the **lhs** of R_j is called $c_{j,1}$, the second $c_{j,2}$ and so on. Similarly the first action will be $a_{j,1}$ and so forth.

We say R_j feeds R_i if the pattern corresponding to some action of R_j , say $a_{j,x}$ matches some condition clause of rule R_i, say $c_{i,y}$. This is expressed in a predicate calculus as :

$$\exists\, x\, \exists\, y\ \ (p_m\ (f_p\ a_{j,x}\ c_{i,y}))$$

where f_p is a function that given some rule action returns the pattern to which the rule action corresponds, and p_m is a predicate that takes two patterns and returns true if they match and false otherwise.

As examples of the notation suppose R_j is the rule *find-parallel-with-mergable-exposures* and R_i is the rule *Remove-mergable-exposures-if-first-is-an-acquisition*. Then:

$(f_p\ a_{j,1}) =$

```
(mergeable-exposures
    ^first-proposal-id          <parallel-proposal-id>
    ^first-version              <parallel-version>
    ^first-exposure-number      <primary-exposure>
    ^second-proposal-id         <parallel-proposal-id>
    ^second-version             <parallel-version>
    ^second-exposure-number     <parallel-exposure>
    ^is-unmergeable             false
    ^is-mergeable-level         <parallel-with-level>
    ^merge-type                 parallel-with
    ^has-unique-label           (genatom)
```

and

$c_{i,2}$ is

```
(mergeable-exposures
    ^first-proposal-id          <proposal-id>
    ^first-version              <version>
    ^first-exposure-number      <first-exposure> )
```

and $(p_m\ (f_p\ a_{j,1})\ c_{i,2})$ is *true*.

The general operation of the predicate p_m on the simple type matches should now be presented. If the element class of two patterns differs, then the predicate returns false. Otherwise, an attribute by attribute match is attempted. If an attribute is paired with a variable in one or both patterns, then the patterns match on that attribute. If they both are paired with the same constant, then again they match. If the attribute is not mentioned in the condition pattern, then it matches the action pattern for the attribute. Note that the p_m predicate returns true if all corresponding pairs of attributes match.

5 Using the Network

Clearly if every rule is coupled with every other rule in the system then the system is completely coupled. The maximum number of arcs is then n^2, where n is the number of rules in the rulebase. The ratio of actual arcs in the network to n^2 is a measure of the degree to which the rulebase is coupled. If the ratio is unity then any rule could interfere with any other. The lower the ratio the more local is the effect of a typical rule in the rulebase and the higher the degree of stamp coupling in the system.

To find the group of rules whose behavior **may** be directly affected by the addition of a new rule we need only regard the network of rules. Figure 4.0 represents a part of the network for some rulebase. The dotted arcs in the figure represent the new coupling that will occur if rule *new-remove* is added to the rulebase. Suppose for the sake of simplicity that both R1 and R6 contain a single make action on their **rhs**. The solid coupling lines flowing from R1 represent the match that occurs between the newly created wme from the make action of R1 and the conditional elements on the **lhs** of R2 and R3. (This is also true for the solid lines between R6 and R3,R4, and R5). The dotted line from R1 to *new-remove* shows that the make-pattern which creates a working memory element will match some set of conditional elements on the LHS of *new-remove*.

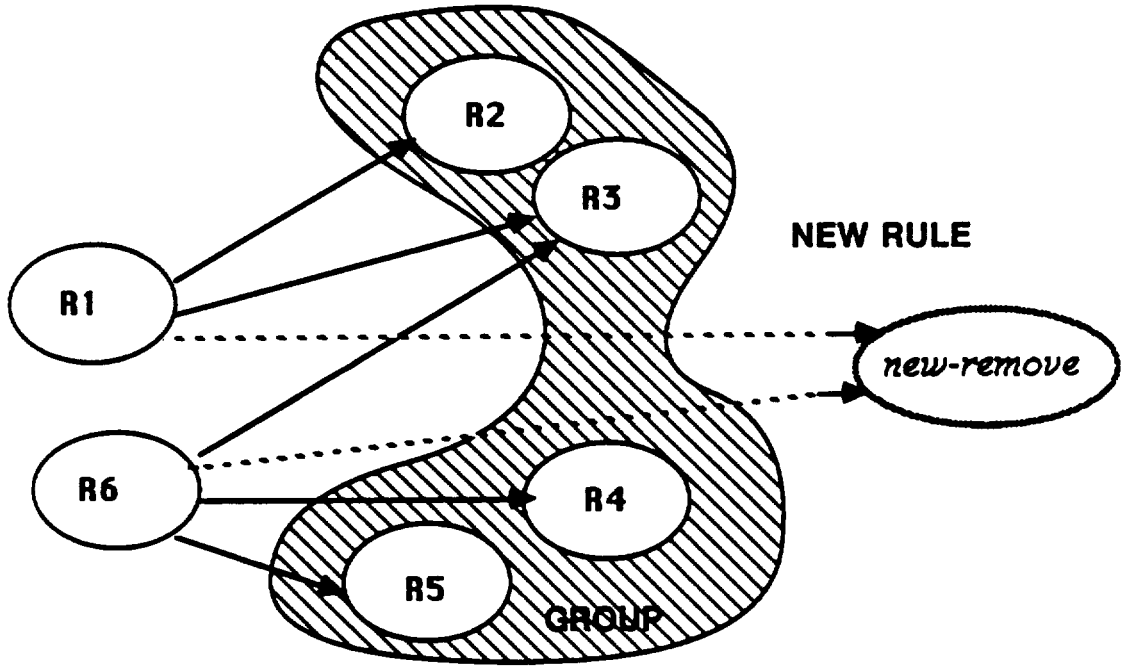


Figure 4.0 *Grouping Rules*

Now, note that it is only possible to remove something from the right hand side of a rule, if the pattern is matched on the left hand side of that same rule. Therefore, the pattern of the element removed in *new-remove* will match some condition in R2 and R3. If *new-rule* fires, and instantiations corresponding to rules R2 and R3 are in the conflict set, the

instantiations will be removed. Thus, the behavior of rules R2 and R3 is altered by the addition of *new-remove*. If R2 and R3 are not fully matched, they will exist in the Rete network. In this case, the *new-remove* will push these rules farther away from the conflict set. Note also that the behavior of R1 is not affected by the addition of *new-remove*.

We can make a similar argument for rules R4 and R5 based on the coupling between R6 and *new-remove*. Simply put, the group of rules affected by the addition of *new-remove* will be those rules with the same parent as *new rule* (i.e. the siblings of *new-remove*).

We therefore see that the group of rules that the new rule could **possibly** affect **directly** is $\{ R_2 , R_3, R_4, R_5 \}$. Naturally the instantiations actually affected by the firing of a rule depends on the data on which a system is operating and the conflict set resolution strategy.

The case for adding a new make rule is similar.The group of rules whose behavior might be affected directly is the set of all immediate successors to the new rule. A new modify, since it is functionally the same as a make and remove, will affect the union of the two groups directly.

Although we have not yet implemented the tool for construction and transversal of the network, its implementation in Lisp should be fairly straightforward. The formal presentation is written mostly in terms of predicates and functions. A dialect of Lisp that supports object-oriented programming could be used to represent nodes by making each node in the net an instance of a "rule class". Methods could be attached to these classes that would retrieve a rule's predecessors, successors, etc. Having devised this general framework for determining coupling between rules, our work is now directed toward implementing this tool and exercising it on the Transformation system.

6 Conclusions

The Transformation system is one of a small group of rulebased systems that has entered into an operational phase. Because the original developer is no longer involved with the day to day operations of the system, the software engineering attributes of the system have become more important. This paper has focused on the nature of rule coupling in the system and has presented a method for understanding the coupling properties of the OPS5 rulebase. We have constructed a general network depicting how rules are coupled within a rulebase, and this network is the basis for deriving a measure of the degree of common coupling for the rules within the rulebase. Furthermore, we have shown how a tool which operates on the principles of this network will allow a maintainer to modify a rulebase with a clearer understanding of how the modification will impact the existing rulebase.

References

[1] Brownston, L., Farrell, R., Kant, E., and Martin, N., *Programming Expert Systems in OPS5*, Addison-Wesley, 1985, pp. 62-64.

[2] Fairley, Richard E., *Software Engineering Concepts*, McGraw-Hill, 1985, pp. 148-151.

[3] Froscher, J. and Jacob, R., *Designing Expert Systems For Ease of Change*, IEE Proceedings of the Expert Systems in Government Symposium, October 1985, pp. 246-251.

[4] Myers, Glenford J., *Reliable Software Through Composite Design,* Van Nostrand Reinhold Company, 1975, pp. 37-39.

[5] Partridge, D. *Engineering Artificial Intelligence Software,* Artificial Intelligence Review, Vol 1., No.1, 1986, pp. 27-41.

[6] Rosenthal, D., *Adding Meta Rules to OPS5 A Proposed Extension,* ACM SIGPLAN Notices, vol 20, no. 10, October 1985, pp. 79-86.

[7] Rosenthal, D., Monger P., Miller G., and Johnston, M., *An Expert System for Ground Support of the Hubble Space Telescope*, Proceedings of 1986 Conference on Artificial Intelligence Applications,NASA Goddard Space Flight Center, May 15, 1986, pp. 43-54.

An Expert System for Scheduling Requests for Communications Links Between TDRSS and ERBS

David R. McLean, Ronald G. Littlefield and David S. Beyer
Bendix Field Engineering Corporation
10210 Greenbelt Rd., Suite 310
Lanham, MD 20706

ABSTRACT

This paper describes the ERBS-TDRSS Contact Planning System (ERBS-TDRSS CPS) that has been built to help Earth Radiation Budget Satellite (ERBS) flight operations personnel generate requests for service from the Tracking and Data Relay Satellite System (TDRSS). The ERBS-TDRSS CPS is written in the C language and runs on an IBM PC-AT. It uses a graphics interface and the Transportable Inference Engine (TIE-1) developed for NASA-GSFC. First, ERBS-TDRSS Ground Track Orbit Prediction data are electronically transferred to the ERBS flight operations area where, in its batch mode, the ERBS-TDRSS CPS automatically generates requests for TDRSS service (usually for one week). In this process, a series of scheduling strategies is used to generate requested events while TIE-1 determines whether each requested event is consistent with ERBS scheduling constraints. As requested events are rejected, alternative context-sensitive strategies are used to generate new requested events until a schedule is completed. In the interactive scheduling mode the ERBS planner can edit a schedule that has been previously built or choose which alternative scheduling strategies the ERBS-TDRSS CPS should use to build a schedule. Finally, a report generator builds "Schedule Requests" for each separate ERBS-TDRSS contact.

1. INTRODUCTION

Generating weekly schedules of requests for communications links (contacts) between the Earth Radiation Budget Satellite (ERBS) and the Tracking and Data Relay Satellite System (TDRSS) has been a manual, labor-intensive task. In this activity, the ERBS flight operations planner first obtains large Ground Track Orbit Prediction data printouts from the Flight Dynamics Facility and performs a visual inspection to find desired ERBS-TDRSS contact times. In selecting good contact times, the planner must consider all of the ERBS scheduling constraints, make experienced judgements, and avoid errors. After a schedule of contacts has been determined, separate "Schedule Requests" for each separate ERBS-TDRSS contact have to be prepared and sent to the Network Control Center (NCC) via the Mission Planning Terminal. If particular requests for TDRSS service are rejected by the NCC, additional effort is required by the ERBS planner to generate alternative requests to satisfy ERBS requirements.

A system to automate this planning process has been built and is being tested by ERBS flight operations personnel. This system is called the ERBS-TDRSS Contact Planning System (ERBS-TDRSS CPS). The ERBS-TDRSS CPS is written entirely in the C language and operates on an IBM PC-AT. It features a user-friendly graphics interface and the Transportable Inference Engine (TIE-1) developed for NASA-GSFC/Code 514 by Bendix Field Engineering Corporation. The Transportable Inference Engine (TIE-1) was described at the 1986 Goddard AI Conference (McLean [1986]). A NASA-GSFC/Code 514 document describing the ERBS-TDRSS CPS and two describing the Interactive Experimenter Planning System Version 3.0 Prototype (on which the ERBS-TDRSS CPS is based) are listed in the reference section.

This paper will first describe how ERBS-TDRSS Ground Track Orbit Prediction data are transferred to the IBM PC-AT in the ERBS flight operations area and used in generating resource windows for the ERBS-TDRSS CPS scheduling process.

Next, the batch scheduling mode of the ERBS-TDRSS CPS will be described. In this mode a schedule of requests for TDRSS service (usually for a 1-week period) can be generated without any interaction with the ERBS flight operations planner. Here, the ERBS-TDRSS CPS uses a series of context-sensitive scheduling strategies (McLean and Littlefield [1987]) from a strategies knowledge base to generate requested events. The embedded inference engine TIE-1 determines whether each requested event is consistent with ERBS scheduling constraints contained in a constraint knowledge base. As requested events are rejected by TIE-1, alternative strategies generate new requested events until a schedule is completed. Examples will be given to show how this strategy and constraint information is represented in knowledge bases.

Following the description of the batch scheduling mode, this paper will describe the various scheduling and editing features of the ERBS-TDRSS CPS in its interactive scheduling mode. In this mode the ERBS flight operations planner can interact with the automated scheduling capabilities of the ERBS-TDRSS CPS to build a schedule or edit a schedule of events that has previously been built.

Finally, the report generator which is used to produce "Schedule Requests" for each separate ERBS-TDRSS contact will be described, and some general conclusions related to the ERBS-TDRSS CPS will be presented.

2. RESOURCE WINDOW GENERATION

ERBS-TDRSS Ground Track Orbit Prediction data are transferred from the Flight Dynamics Facility (FDF) at NASA-GSFC to an IBM 4341 mainframe in the Command Management Facility (CMF) area where a "filter" program is run to obtain a desired subset of the orbit data. The filtered orbit data are then transferred electronically to an IBM PC-AT located in the ERBS flight

operations area where the "ERBS to IEPS" (ETOI) program reformats the data, builds files containing data for one week or less (because of IBM PC-AT memory limitations), and generates resource windows for the ERBS-TDRSS CPS scheduling process.

In running the ETOI program, the ERBS flight operations planner must specify: (1) whether ERBS is flying "forward" or "backward" along its orbital track (the ERBS flight orientation is periodically switched and this affects antenna obstruction angles) and (2) the azimuth and altitude obstruction angles to be used for the S-band Single Access-2 (SSA2) TDRS antenna and the Multiple Access (MA) TDRS antenna. Presently there is only one TDRS in the TDRS System for the ERBS-TDRSS CPS to consider. From the orbit data the ETOI program reads: (1) ERBS daylight period information, (2) ERBS orbit number information, and (3) TDRS antenna angle information. As output, the ETOI program specifies:

a. Start and stop dates and times for ERBS orbital daylight periods.

b ERBS orbit numbers with start and stop dates and times

c. TDRS MA antenna viewing periods for ERBS

d. TDRS SSA2 antenna viewing periods for ERBS

3. THE BATCH SCHEDULING MODE

In the batch scheduling mode the ERBS-TDRSS CPS builds a schedule of ERBS-TDRSS contacts for two different types of spacecraft activities: (1) tape recorder dumps and (2) Stratospheric Aerosol and Gas Experiment (SAGE) monitor events. Tape recorder dumps are for sending tape recorded data from ERBS to TDRSS. SAGE monitor events are for sending real-time data obtained by the ERBS SAGE-II instrument (see ERBE Program document listed in the references) during selected sunrise or sunset periods.

In building a schedule of ERBS-TDRSS contacts, either the MA TDRS antenna or the SSA2 TDRS antenna can be requested. Tape recorder dump events (TRU dumps) are planned every other orbit (if possible) and require an event duration of either (1) 30 minutes via the MA TDRS antenna or (2) 20 minutes via the SSA2 TDRS antenna. A 30-minute TRU dump via the MA TDRS antenna is requested first (by the "START" strategy). If the MA TDRS viewing period is not available, the ERBS-TDRSS CPS next requests a 20-minute TRU dump during a SSA2 TDRS viewing period (using the "EVENT" strategy). If neither the original MA TDRS viewing period nor the SSA2 TDRS viewing period is available, the ERBS-TDRSS CPS requests (as a third choice) a 30-minute TRU dump via MA in the next orbit (using the NEXT strategy). The way this strategy information is represented in the strategies knowledge is illustrated in the example below.

```
TRU Dump via MA

;event to be scheduled every 2 orbits
     2
;duration of event:
     0:30
;strategies:
     START MA TDRS viewing periods
     EVENT TRU Dump via SSA2
     NEXT
.

TRU Dump via SSA2

;event to be scheduled every 2 orbits
     2
;duration of event:
     0:20
;strategies:
     START SSA2 TDRS viewing periods
.
```

One sunrise SAGE monitor event and one sunset SAGE monitor event are requested every 16 orbits. These events must each have a duration of 13 minutes for either the MA TDRS antenna or the SSA2 TDRS antenna and must take place while ERBS is in an appropriate sunrise or sunset window. Sunrise SAGE events must start 10 minutes before ERBS daylight while sunset SAGE events must start 10 minutes before ERBS night. A 13-minute MA TDRS antenna contact is requested first. If the MA TDRS viewing period is unavailable, a 13-minute SSA2 TDRS antenna contact is requested. If neither the MA TDRS nor the SSA2 TDRS antenna contact is available, a forward-backward search algorithm (NEXT-PRIOR strategy) is initiated to look for the MA TDRS antenna viewing period that is as close as possible to the initial requested event. If a previously scheduled TRU dump event is encountered, the ERBS-TDRS CPS tries to "shift" the TRU dump event within its resource window in order to make room for the SAGE monitor event.

An example of the way these primary and alternative strategies for sunrise SAGE events are represented in the strategies knowledge base is shown below and on the next page.

```
Sunrise SAGE via MA

;event to be scheduled every 16 orbits
     16
;duration of event:
     0:13
;bias
     -0:10
;shift
     TRUE
;strategies:
     START Daylight
     EVENT Sunrise SAGE via SSA2
     NEXT
     PRIOR
```

```
NEXT 2
PRIOR 2
NEXT 3
PRIOR 3
NEXT 4
PRIOR 4
NEXT 5
PRIOR 5
NEXT 6
PRIOR 6
NEXT 7
PRIOR 7
```

The constraint knowledge base contains the ERBS scheduling constraints used by the inference engine (TIE-1) to determine whether requested events from the strategies knowledge base can he scheduled. For TRU dumps via the SSA2 TDRS antenna, the constraint knowledge base requires: (1) that the start and stop times requested be within the appropriate antenna availability window, (2) that no other events be scheduled during the interval requested, and (3) that the event be of the proper duration. If a constraint is violated a diagnostic message from the constraint knowledge base is written to a log file. In the interactive mode this diagnostic message is displayed on the screen. The manner in which this information is represented in the constraint knowledge base is shown below and on the next page.

```
TRU Dump via SSA2

event name                  eq      TRU Dump via SSA2

in SSA2 TDRS window         eq      true

     "The SSA2 TRU Dump you are trying to schedule must
      be within a SSA2 TDRS window."
```

event duration eq 20

"The duration of the SSA2 TRU Dump you are trying to schedule is not 20 minutes. A PE-S02 event requires 20 minutes."

conflicting event ne TRU Dump via MA

"The SSA2 TRU Dump you are trying to schedule conflicts with a previously scheduled TRU Dump via MA."

conflicting event ne Sunrise Sage via MA

"The SSA2 TRU Dump you are trying to schedule conflicts with a previously scheduled Sunrise SAGE event via MA."

conflicting event ne Sunrise Sage via SSA2

"The SSA2 TRU Dump you are trying to schedule conflicts with a previously scheduled Sunrise SAGE event via SSA2."

conflicting event ne Sunset Sage via MA

"The SSA2 TRU Dump you are trying to schedule conflicts with a previously scheduled Sunset SAGE event via MA."

conflicting event ne Sunset Sage via SSA2

"The SSA2 TRU Dump you are trying to schedule conflicts with a previously scheduled Sunset SAGE event via SSA2."

For sunrise SAGE events via the MA TDRS antenna, the constraint knowledge base requires: (1) that the start and stop times requested be within the appropriate antenna availability window, (2) that the start and stop times requested be within the appropriate sunrise window, (3) that no other event is scheduled during the interval requested, and (4) that the event is of the proper duration. If a constraint is violated, a message is written to a log file. In the interactive mode, this diagnostic message is displayed on the screen. The way in which this information is represented in the constraint knowledge base is shown below and on the next page.

Sunrise SAGE via MA

event name eq Sunrise SAGE via MA

in SAGE morning interval eq true

"The sunrise MA SAGE event you are trying to schedule does not start 10 minutes before onset of orbit daylight. Respecify the event times."

in MA TDRS window eq true

"The Sunrise MA SAGE event you are trying to schedule must be within an MA TDRS window."

event duration eq 13

"The duration of the MA SAGE event you are trying to schedule is not equal to 13 minutes. A PE-M02 event requires 13 minutes."

conflicting event ne TRU Dump via SSA2

"The MA SAGE event you are trying to schedule conflicts with a previously scheduled TRU Dump via SSA2."

conflicting event ne TRU Dump via MA

"The MA SAGE event you are trying to schedule conflicts with a previously scheduled TRU Dump via MA."

conflicting event ne Sunrise Sage via SSA2

"The MA SAGE event you are trying to schedule conflicts with a previously scheduled sunrise SAGE event via SSA2."

conflicting event ne Sunset Sage via MA

"The MA SAGE event you are trying to schedule conflicts with a previously scheduled sunset SAGE event via MA."

conflicting event ne Sunset Sage via SSA2

"The MA SAGE event you are trying to schedule conflicts with a previously scheduled sunset SAGE event via SSA2."

4. INTERACTIVE SCHEDULING MODE

The ERBS flight operations planner begins an interactive scheduling session by calling up a schedule (either a blank schedule or a schedule that has been previously built) and selecting the types of items to be displayed in the schedule-building window. These items are initially displayed on 24-hour timelines where the start and stop times of each event are

shown graphically. From the main menu located just below the schedule-building window, the planner may select one of eight options by (1) using cursor positioning or (2) using the key representing the first letter of the menu option, and then pushing the ENTER key. The eight options that can be selected from the main menu are:

HELP: Pushing the "h" key at any time provides help messages concerning the operation of the scheduling system.

ITEMS: Pushing the "i" key allows the planner to select a different group of timeline items to be displayed in the schedule-building window.

DAY: Pushing the "d" key allows the planner to select the day for which 24-hour timelines are shown in the schedule-building window.

EDIT: Pushing the "e" key provides the planner with several options for editing schedules (described in more detail below and on the next page).

LIST: Pushing the "l" key provides numerical listings of the start and stop times for the items shown graphically in the schedule-building window.

ZOOM: Pushing the "z" key allows timelines shown in the schedule-building window to be zoomed from 24 hours to as few as 8 minutes.

SCROLL: Pushing the "s" key allows the planner to scroll to new items or to new time values.

QUIT: Pushing the "q" key provides several options for saving files and exiting from the interactive scheduling mode.

If the EDIT option is chosen, the planner can either (1) insert or delete individual events in a schedule or (2) choose which alternative scheduling strategies the ERBS-TDRS CPS should use to build a schedule of events. If the first EDIT option is chosen, a type of spacecraft activity to edit is first selected. Next, an event of the selected type is manually requested by (1) typing the desired start and stop times for the event or (2) using the cursor to graphically locate the desired start and stop times for the event on the graphics interface. After an event is requested, the inference engine (TIE-1) determines whether the requested event is consistent with the ERBS scheduling constraints in the constraint knowledge base. If the inference engine returns a status of "OK", the color of the bar that represents the requested event on the graphics interface changes to blue to indicate that the event has been successfully scheduled. However, if the inference engine returns a status of "NOT-OK", the system sends a diagnostic message to the planner indicating why the selected event cannot be scheduled.

While in the EDIT mode the planner can invoke the ZOOM, SCROLL, or LIST feature without going back to the main menu by pushing the "z", "s", or "l" key. In addition, three other editing features are available directly from the EDIT mode by key selection, including:

OVERRIDE: Pushing the "o" key allows the planner to enter a password to override the constraints in the constraint knowledge base to schedule an event.

EXPAND: Pushing the "e" key while the cursor is resting on a scheduled event provides an expanded listing of information concerning that particular event.

DELETE: Pushing the "d" key while the cursor is resting on a scheduled event deletes the event.

The second main EDIT option allows the ERBS flight operations planner to interact with the ERBS-TDRS CPS and choose which alternative scheduling strategies the ERBS-TDRS CPS should use to build a schedule. In this mode the planner has the option of: (1) utilizing only the primary scheduling strategies in the strategies knowledge base (ignoring alternative scheduling strategies) to build a schedule, (2) utilizing all of the context-sensitive scheduling strategies (primary and alternative) in the strategies knowledge base to build a schedule, or (3) specifying with the "y" (yes) key or the "n" (no) key which alternative scheduling strategies the ERBS-TDRSS CPS should use to build a schedule. With the third option, the ERBS-TDRSS CPS asks the planner whether each specific alternative strategy can be used, waits for the planner's yes or no response, and then either tries the strategy or not depending on the planner's response. If a particular scheduling strategy fails, the ERBS-TDRS CPS provides a diagnostic message to explain why the event cannot be scheduled. As events are successfully scheduled, they are displayed graphically on the appropriate timeline.

5. GENERATING SCHEDULE REQUESTS

After a schedule of contacts has been built and saved as an output file, a "report-generation" program is invoked to read information in the output schedule file, along with information from the activities section of the strategies knowledge base, to prepare "Schedule Requests" for each separate ERBS-TDRSS contact period. The ERBS flight operations planner controls the report generation program through a configuration file by specifying: the date to be printed on each form, the name of the output schedule file to be read, the name of the report file to be created, the name of the strategies knowledge base to be used, the form file to be used, and the map file to be used. The map file specifies where data from the output schedule file and the strategies knowledge base are placed on the completed "Schedule Request" forms.

The requests for TDRSS service are then sent (via the Mission Planning Terminal) to the Network Control Center which has the responsibility of determining the usage of the Tracking and Data Relay Satellite System by all NASA and non-NASA users. If particular requests for TDRSS service are rejected, the ERBS-TDRSS CPS can be used to select and generate alternative ERBS-TDRSS contact requests to try to satisfy ERBS mission requirements.

6. CONCLUSIONS

As this paper is being completed, the ERBS-TDRSS CPS has been delivered to ERBS flight operations personnel for use but has not been fully evaluated. Early indications are that the ERBS-TDRSS CPS will be very useful in automating several planning/scheduling activities performed by the ERBS flight operations planner, including:

a. Reading ERBS-TDRSS Ground Track Orbit Prediction data and generating resource windows.

b. Applying context-sensitive primary and alternative scheduling strategies to generate requested events. If desired, the ERBS flight operations planner may interact with the ERBS-TDRSS CPS to select which alternative scheduling strategies the ERBS-TDRS CPS uses to generate requested events.

c. Checking requested events for consistency with the ERBS scheduling constraints before adding them to a schedule of events. With the use of a special password, the ERBS flight operations planner may manually overide the constraints in the constraint knowledge base to schedule an event.

d. Preparing "Schedule Request" forms for each separate ERBS-TDRSS contact in the proper format.

A valuable feature of the ERBS-TDRSS CPS is that its strategies and constraint knowledge bases can be easily edited to reflect changing operational requirements for the system that may occasionally arise. In addition, since the source code for the entire ERBS-TDRSS CPS (including the embedded inference engine and the user-friendly graphics interface) was developed by the NASA-GSFC/Code 514 contractor, extensive modifications or enhancements to the system can be made.

ACKNOWLEDGEMENTS

The authors would like to thank Patricia Lightfoot, William Macoughtry, and Carolyn Dent at NASA-GSFC/Code 514 and Ellen Stolarik at Bendix Field Engineering Corporation for their continual support and many contributions to the Interactive Experimenter Planning System task. This work was supported by NASA contract NAS5-27772.

REFERENCES

McLean, David R., "The Design and Application of a Transportable Inference Engine (TIE-1)", Proceedings of the 1986 Conference on Artificial Intelligence Applications, Goddard Space Flight Center, Greenbelt, MD, May 15, 1986.

User's Manual for the ERBS-TDRSS Contact Planning System, NASA-GSFC/Code 514, March 1987.

User's Manual for the Interactive Experimenter Planning System (IEPS) Version 3.0 Prototype Software, NASA-GSFC/Code 514, May 1986.

Software Maintenance Document for the IEPS Version 3.0 Scheduling System, NASA-GSFC/Code 514, March 1987.

McLean, David R. and Littlefield, Ronald G., "A Context-Sensitive Strategies Interpreter for Scheduling Alternative Events", submitted for presentation at the Third Annual Expert Systems in Government Conference, McLean, VA, Oct. 1987.

Earth Radiation Budget Experiment Program (Observations to Better Understand Climate), NASA-GSFC/Code 513.

The Mission Operations Planning Assistant

James G. Schuetzle

Ford Aerospace & Communications Corporation
Space Station Programs
4920 Niagara Road
College Park, Maryland 20740

ABSTRACT

The Mission Operations Planning Assistant (MOPA) is a knowledge-based system developed to support the planning and scheduling of instrument activities on the Upper Atmospheric Research Satellite (UARS). The MOPA system represents and maintains instrument plans at two levels of abstraction in order to keep plans comprehensible to both UARS Principal Investigators and Command Management personnel. The hierarchical representation of plans also allows MOPA to automatically create detailed instrument activity plans from which spacecraft command loads may be generated.

The MOPA system was developed on a Symbolics 3640 computer using the ZetaLisp and ART languages. MOPA's features include a textual and graphical interface for plan inspection and modification, recognition of instrument operational constraint violations during the planning process, and consistency maintenance between the different planning levels. This paper describes the current MOPA system.

INTRODUCTION

The UARS is a multi-instrument orbiting observatory scheduled to be launched by the Space Shuttle in 1991. The UARS will provide experimenters at remote locations with data on the temperature, composition, and dynamics of the earth's upper atmosphere.

Mission planning for a satellite such as the UARS is a complex and knowledge intensive process. There are ten instruments whose activities must be defined and coordinated daily for the estimated 18 month life of the mission. The mission planners need to be cognizant of the functions and capabilities of each instrument as well as the spacecraft itself. In addition, there are constraints and interdependencies among the instruments themselves and between the instruments and the spacecraft. Mission planners are also required to reason at different levels of abstraction during the process of translating high level descriptions of instrument activity into detailed command sequences.

BACKGROUND

The instrument planning scenario for the UARS is divided into three phases with each phase resulting in the formulation of a different plan. The three plans (in chronological order) are the Long Term Science Plan, the Daily Science Plan, and the Activity Plan.

The Long Term Science Plan is developed by the UARS instrument scientists (principal investigators) prior to the launch of the spacecraft. This plan describes how each of the instruments will be utilized in order to achieve the scientific objectives of the mission. After UARS launch and instrument checkout, the principal investigators will update the Long Term Science Plan to reflect changes in instrument performance or to include the study of new items of scientific interest. It is expected that this plan will change infrequently.

The Mission Planning group, based at GSFC, is responsible for developing a Daily Science Plan for each day of the UARS mission. This plan is developed using the operational strategy developed in the Long Term Science Plan. The Daily Science Plan specifies the operations of each instrument in terms of scientific objectives.

The final plan, the Activity Plan, is also developed daily by the Mission Planning Group. The Activity Plan specifies the operation of the instruments in terms of the detailed activities that each instrument will perform. These instrument activities at are at a level of detail suitable for the generation of spacecraft commands. This command generation is performed by the UARS Command Management System using the Activity Plan as input.

PROBLEM

The problem addressed by the MOPA system is to support the Mission Planning Group in the process of generating Daily Science and Activity Plans. In order to create these plans, the Mission Planning Group requires some of the knowledge possessed by the principal investigators. This knowledge falls into three general categories:

1. Instrument operations to achieve scientific objectives

   Each instrument can operate in different modes depending on the particular objective to be satisfied. For example, most of the instruments have several modes for data collection, calibration, and safing. The Mission Planning Group must know these modes and the conditions under which they should be selected in order to achieve the scientific objectives of the mission.

2. Instrument operational limitations and restrictions

   There are certain operating constraints placed upon the instruments. The improper operation of an instrument can result in damage to itself or another instrument, or in degradation of the data being captured by the instrument. The Mission Planning Group must be aware of these restrictions during the process of generating Daily Science and Activity Plans.

3. Translation of scientific objectives into detailed command sequences.

   The process of creating the Activity Plan from the Daily Science Plan requires that the Mission Planning Group decompose activities specified in terms of scientific objectives into the detailed activities which must be executed by the instrument to carry out the objective.

MOPA CAPABILITIES

The MOPA system provides many features to aid the mission planner in the creation of the various plans. This section provides an overview of MOPA features.

Generic Plans

In the MOPA system, the Principal Investigators knowledge of instrument operations to achieve scientific objectives is represented in the form of Generic Plans. MOPA Generic Plans attempt to capture much of the information present in the Long Term Science Plan. These plans are specified at such a level of abstraction as to be readily interpretable by the UARS Principal Investigators. The Generic plan specifies for each instrument, the conditions under which an instrument operation should be performed. Typically the conditions are expressed in terms of spacecraft orbital events such as sunrises or equator crossings. Additionally, these conditions may be expressed in terms of the operations of the spacecraft or other instruments. The instrument operations specified in the Generic plan identify abstract functions, usually scientific objectives, to be accomplished by the instrument. MOPA provides the mission planner with the ability to choose from a collection of previously created Generic Plans, selecting one that may be appropriate for the particular planning day.

Daily Science Plan generation

The MOPA system uses the selected Generic Plan together with orbital and other events scheduled for the day in order to automatically generate a Daily Science Plan. The Daily Science Plan is graphically represented by the MOPA interface in a timeline format. MOPA can also, at the option of the user, resolve conflicts between instrument operations in the plan. The user interface provides the planner with the ability to quickly view and modify the Daily Science Plan created by MOPA.

Activity Plan generation

Once the Daily Science Plan has been generated and reviewed, MOPA can automatically derive the Activity Plan from it. The Activity plan can then be displayed and modified through the user interface in either a graphical or textual manner. As with the Daily Science Plan, MOPA can resolve any conflicts between instrument operations.

Constraint checking

MOPA provides the capability to constraint check instrument operations specified in either the Daily Science or the Activity Plans. The MOPA system can represent and check a wide variety of operational constraints.

User Interface

The main purpose of the MOPA user interface is to provide the user with mechanisms to create, examine, and modify Daily Science and Activity Plans. MOPA has two different types of displays for these plans; A textual display and a graphical timeline display. MOPA reduces the amount of typing the user must perform by making extensive use of the mouse. Typical uses of the mouse are to select commands from menus, manipulate instrument activities and events, and scroll through displays.

Reverse Plan Maintenance

The Reverse Plan Maintenance feature of MOPA allows the user to make modifications to either the Activity Plan or the Daily Science Plan and "reflect" these changes back to the high level plan for the day, the Generic Plan. The "tailored" Generic Plan can then be saved for later use. This feature saves the user from having to edit the Generic Plan and then regenerate both the Daily Science and Activity Plans.

USER INTERFACE

In order to take advantage of the Symbolics windowing system features and object oriented design methodology, we have implemented the user interface portion of MOPA entirely in ZetaLisp. This design decision requires us to map ART schemata into ZetaLisp flavor objects which can then be used by the user interface subsystem. This mapping has been made almost transparent by using the flavors facility of ZetaLisp. We have defined a flavor called LART (Lisp to ART) from which all references to ART schemata from Lisp are made. Using the LART flavor, we can attach procedures (methods) to the schemata to perform such functions as accessing and updating attributes, and drawing activities and events on the screen.

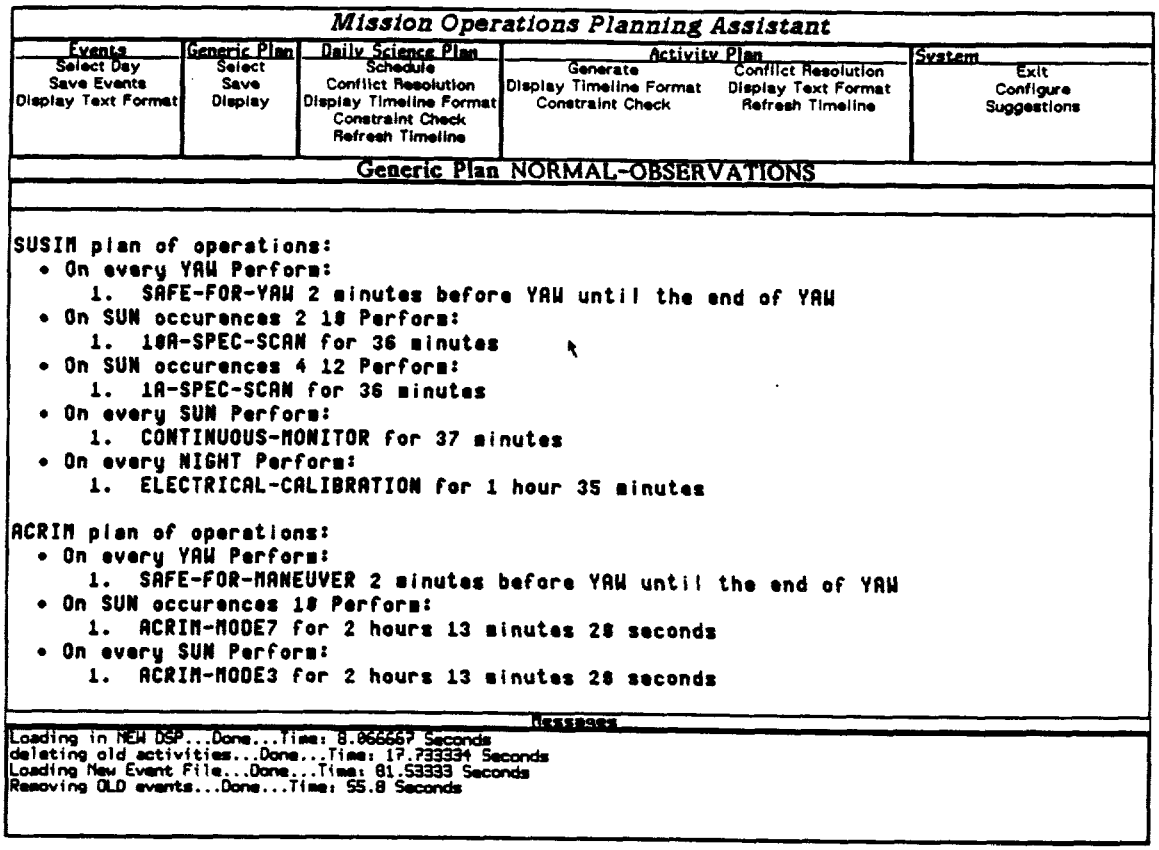


Figure 1

Figure 1 illustrates the MOPA display of a sample Generic Plan. The menus at the top of the screen allow the user to select different MOPA functions by pointing the mouse at the items within the menus. The menus are divided into functions pertaining to Events, Generic Plans, Daily Science Plans, and Activity plans. A System menu is available to modify certain system parameters and to enter suggestions or comments about experiences with the MOPA system.

At the center of the MOPA screen is a sample Generic Plan. Within the Generic Plan are entries for each UARS instrument which describe when certain instrument operations should be performed. This display may be vertically scrolled using the mouse cursor. At the bottom of the screen is a window area for informational messages. The messages within this display may also be scrolled using the mouse cursor.

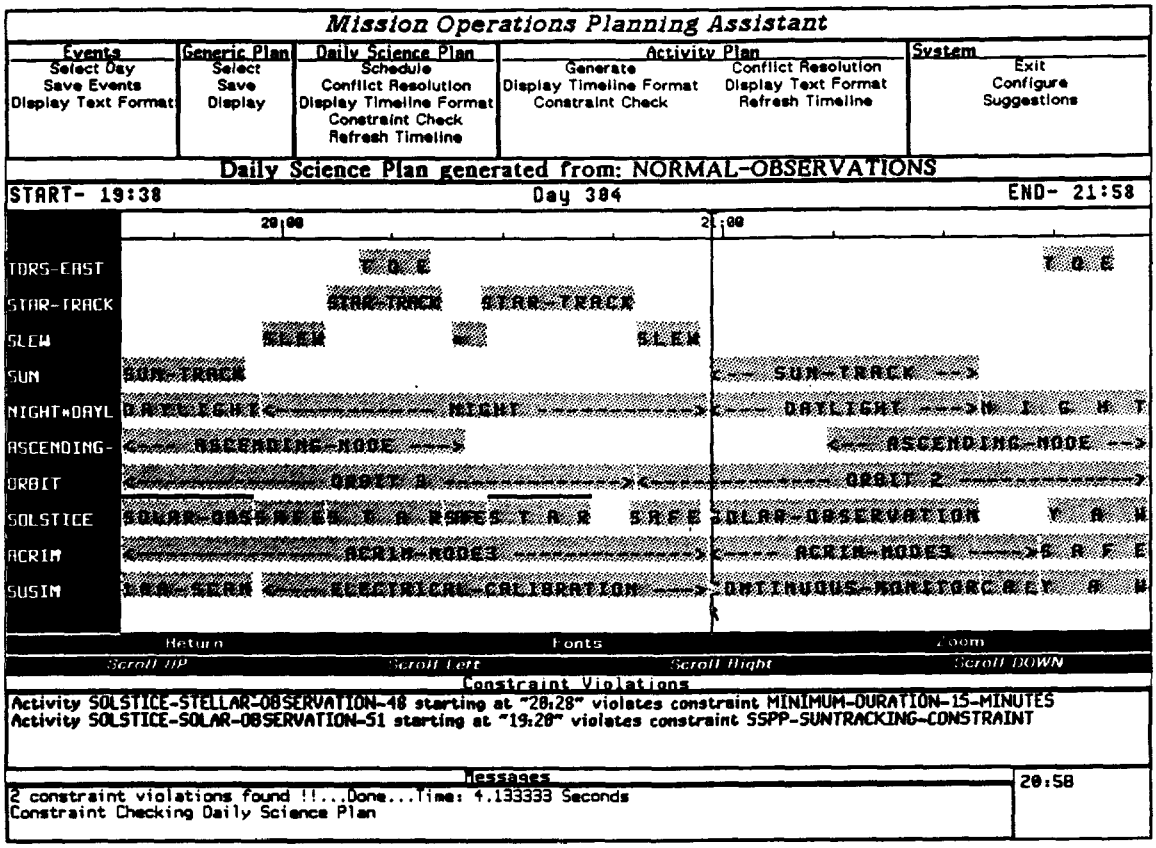


Figure 2

A timeline display of a Daily Science Plan is shown in figure 2. At the top of the display is a title which indicates the plan level and the name of the plan. On the next line are displayed the start time of the interval being displayed, the Julian date, and the interval end time. The interface provides both horizontal and vertical scrolling of the display. Horizontal scrolling can be accomplished in several ways according to user preference. The character font for the display may also be adjusted. Each timeline in the display can also be manipulated. The user can rearrange timelines, delete them from the display, or merge two timelines together. By pointing the mouse cursor over activities or events, the user can obtain a menu of several actions that may be performed; delete, edit, and replace. Constraint violation messages appear in the display below the timeline display. Activities with constraint violations are highlighted in the timeline display by drawing a thick line above them. The constraint violation messages have functions associated with them; the user may either display the constraint structure itself, or locate the activity with the violation.

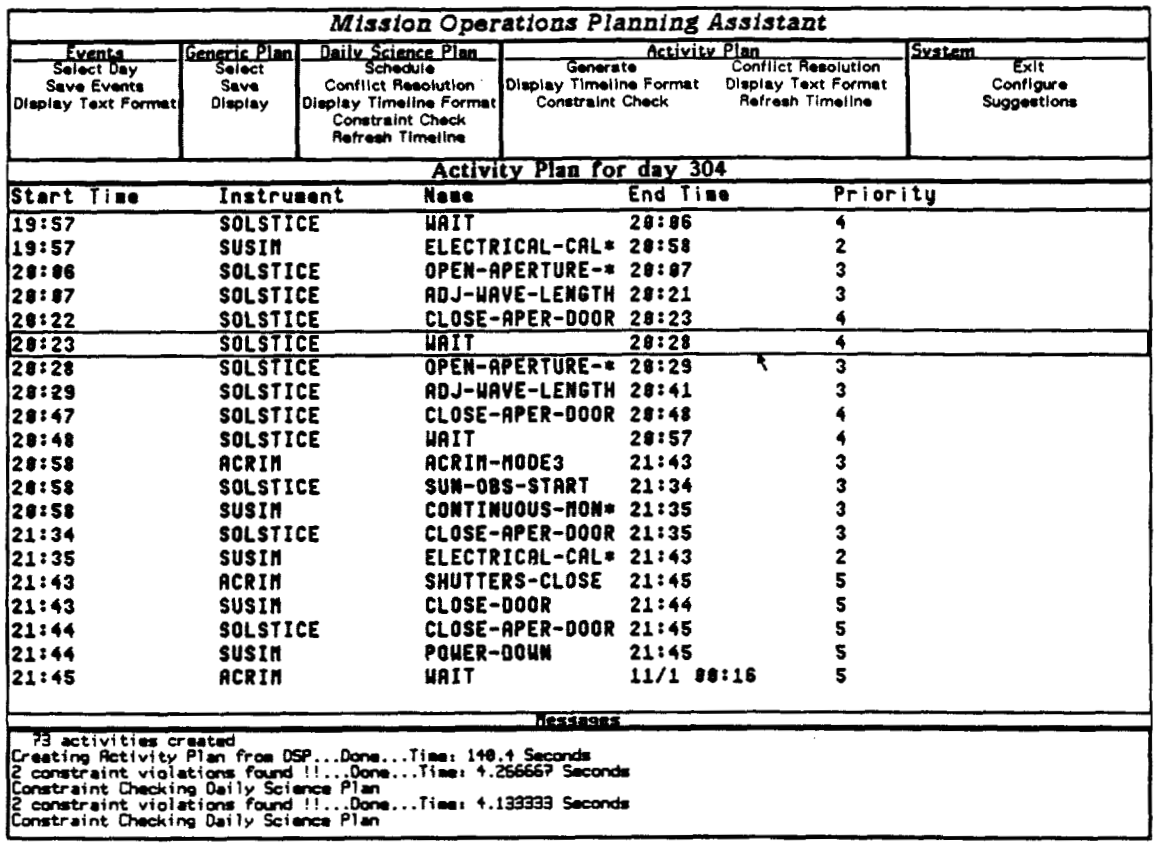


Figure 3

The Activity Plan can be displayed in a textual format as shown in figure 3. The attributes of the activities are displayed at the top of the display and their values for each activity are displayed below. Before the display appears, the user is presented with a menu of all activity attributes. The user may then select those attributes which he would like displayed as well as the order he would like them displayed in. In this way, the activities can be presented to the user sorted by any activity attribute, for example by start time or instrument. This display may be scrolled vertically using the mouse. Each activity entry in the display is mouse sensitive. The user can delete, add, or edit an activity simply by pointing the mouse over the activity.

KNOWLEDGE REPRESENTATION

There are several classes of entities which MOPA must reason about; Events, Activities, Instruments, Plans, and Constraints. We have chosen to represent all of these entities using ART's frame based representation facility referred to as schemata. In this section we discuss the representations of two important entities: instrument activities and constraints.

Activities

Central to the design of MOPA is the representation of instrument activities. Figure 4 illustrates a sample instance of an activity. All activity schemata have certain basic attributes associated with them. These attributes include the activity start time, end time, and duration. An activity is linked to the instrument which performs it via the activity-of relation. There are two relations of an activity which relate the activity to constraints. The has-pre-condition relation has as its value the constraint schema which limit the performance of the activity. The has-constraint-violation relation is used during the constraint checking process. The value of this relation is either NIL, meaning this activity violates no constraints, or the names of the constraint schema that this activity violated.

Activity Abstraction Hierarchies

Activities are represented at different levels of abstraction in the Daily Science plan and Activity plan. In the Daily Science Plan, the activities to be performed by an instrument are specified in terms of the scientific objective to be obtained. At the Activity Plan level, the activities are the detailed operations necessary to carry out the objective. MOPA's representation of activities allows for the construction of activity hierarchies to provide for both levels of specification.

For example, one function of a particular instrument might be to collect data about carbon dioxide level at some point in the earth's atmosphere. In order to make the collection of this type data, the instrument is powered on, a filter of the instrument must be set to a certain position, the instrument's microprocessor must cycle in this configuration for some duration, and then the instrument is powered off at completion. We might define a hierarchy of activities in which "Collect CO2 Data" could be an abstract activity composed of the primitive activities power-on, set-filter, cycle, and power-off. The planner could then

specify "Collect CO2 Data" as an activity to be performed by the instrument in the Generic or Daily Science plans.

Two relations within the activity schema facilitate the construction of activity abstraction hierarchies: has-sub-activity and sub-activity-of. The has-sub-activity relation links an activity to its detailed components, the sub-activity-of relation provides a link in the opposite direction. MOPA allows these activity hierarchies to be constructed in an arbitrarily complex manner, with as many levels of activities as are desired. Different abstraction hierarchies may also share common components if necessary.

The activity shown in Figure 4 has a link to three constraint schemata via the has-pre-condition relation. This activity is associated with two more detailed activities via the has-sub-activity relation.

Activity instance schema

```
(defschema solstice-solar-observation
  (is-a activity)
  (name solar-obs)
  (short-name sun)
  (has-pre-condition minimum-duration-10-minutes
                     maximum-duration-38-minutes
                     sspp-suntracking-constraint)
  (activity-of solstice)
  (duration 9000)
  (has-sub-activity (sun-obs-start
                     sun-obs-stop)))
```

Figure 4

Constraints

The activities of the UARS instruments are often restricted by operational limitations called constraints. In MOPA, we define a constraint as a condition which must be true in order for an activity to be performed. Constraints, represented as ART schemata, are linked to the activities which they constrain via the has-pre-condition relation. These constraints take a variety of forms. This section describes the classification of constraints used by MOPA and their representations.

The schema definition of a generic constraint is shown in Figure 5. Each constraint has a severity attribute associated with it. The value of this attribute is an indication of the severity of violating this constraint.

In the MOPA prototype the two possible values for this attribute are LOSS-OF-DATA and INSTRUMENT-DAMAGE. The result attribute indicates the specific result of violating the constraint. The description attribute of the constraint is a text representation of the error message given the user if the constraint is violated.

Generic Constraint schema

```
(defschema constraint
  (severity)
  (result)
  (constraint-type)
  (description))
```

Figure 5

Constraint Classification

We have identified two major classes of constraints: value constraints and relational constraints. The value type of constraint, shown in Figure 6, restricts the range of values that an attribute of an activity schema can have. This type of constraint is often used in MOPA to enforce maximum and minimum durations of an activity. The slot-name attribute specifies the attribute of the activity whose value is to be restricted. The restriction-value attribute identifies the constraining value of the attribute. The predicate-name attribute specifies a Lisp predicate to be invoked to compare the restriction-value to the value of the activity slot-name attribute.

Slot Value Constraint Class schema

```
(defschema slot-value-constraint
  (is-a constraint)
  (slot-name)
  (restriction-value)
  (constraint-type slot-value-constraint)
  (predicate-name))
```

Figure 6

The other general class of constraint, the relational constraint, is shown in Figure 7. This class of constraint specifies a relation which must hold between two activities. For example, the relational constraint allows us to specify the constraint that one activity must occur during another activity.

The predicate-name attribute of the relation-constraint schema has the same meaning it did for the value constraint. The schema-name attribute of the relation-constraint schema, points to an activity schema. The relation defined in the predicate-name attribute must be true between the schema-name attribute value, and the activity to which this constraint is attached.

Relational Constraint Class schema

```
(defschema relation-constraint
  (is-a constraint)
  (schema-name)
  (predicate-name))
```

Figure 7

An example of a relational constraint used in MOPA is the sspp-suntracking constraint shown in figure 8. This constraint is attached to activities such as Solstice-solar-observation to indicate that the SSPP must be in Sun tracking mode during these activities. In this case, the schema-name attribute of the sspp-suntracking-constraint schema has the value sun-track, which is the SSPP Sun Tracking event. The predicate-name value is overlaps, a predicate which determines if one activity or event overlaps another in time.

Relational Constraint instance schema

```
(defschema sspp-suntracking-constraint
  (instance-of relation-constraint
  (schema-name sun-track)
  (predicate-name overlaps)
  (severity loss-of-data)
  (result "Poor data quality")
  (description "This activity requires the SSPP
                to be in sun-tracking mode"))
```

Figure 8

DESIGN

The section discusses the design of two important processes in MOPA; The generation of Activity Plans from Daily Science Plans, and the constraint checking process. These two processes operate on the activity and constraint schemata representations described in the previous section.

Activity Plan Generation

The MOPA prototype system automatically generates an Activity Plan from a Daily Science Plan. This process involves a decomposition of abstract activities into their detailed component activities. The generation process contains two main components: Activity Decomposition and Activity Instantiation.

Activity Decomposition - The Activity Plan generation process begins with a collection of scheduled activities at the Daily Science Plan level. The activities specified at this level may have has-sub-activity links to other more detailed activities, or they may be "primitive" activities (i.e. They have no has-sub-activity links).

The generate process iterates through a list of all activities specified at the DSP level. Each activity is processed individually. If the activity is "primitive", then decomposition is unnecessary, and the activity is added to the list of activities at the Activity Plan level. If the activity is linked to more detailed activities via the has-sub-activity relation, the decomposition process performs a recursive depth-first search of the activity hierarchy. This search collects a list of activities found at the "bottom" or lowest level of the activity hierarchy. This list of activities will be included in the list of Activity Plan level activities after each new activity is instantiated.

Activity Instantiation - If the decomposition process is performed, a list of detailed activities is created. These detailed activities are the schema names of the generic activities. The generic activities provide default values for the activities which are actually instantiated (scheduled). The Activity Instantiation process creates instances of these generic activities, filling in specific values for attributes of the particular instance. The attributes whose values must be assigned are: start-time, duration, end-time, sub-activity-of, priority, and has-event.

Constraint Checking

There are three components involved in the checking process: Generic constraint processing, Constraint schema definitions, and Constraint evaluation functions. The constraint schema definitions were described in the previous section on Knowledge Representation.

Generic Constraint Processing - Constraint checking can be performed at either the Daily Science or the Activity Plan levels. In the MOPA user interface, these two options are selectable from the menus at the top of the MOPA displays. MOPA maintains separate lists of activities created at each of the two plan levels. When constraint checking is selected, MOPA examines only those activities created at the appropriate plan level.

The Generic Constraint processing functions iterate through activities examining their has-pre-condition attribute. If the value of this relation is not NIL, MOPA proceeds to evaluate each constraint attached to the activity. As described in the Knowledge Representation section, MOPA has two general classes of constraints: Value and Relational. MOPA checks the constraint type of the constraint and invokes one function to evaluate Value constraints, and another to evaluate Relational constraints. The arguments to these functions are the constrained activity and the constraint itself.

The evaluation of the Value constraint types is fairly straightforward. The predicate-name, slot-name, and restriction-value values are first retrieved from the constraint schema. Then, using the slot-name value, the appropriate attribute value from the activity schema is retrieved. Finally, the predicate-name function is invoked with the attribute value of the activity and the restriction-value as arguments. If the function returns the value NIL (false), then a constraint violation is detected and a function is invoked to attach the constraint schema to the has-constraint-violation attribute of the activity. A message is also sent to the Constraint Violations window.

The Relational constraint evaluation process is similar to the evaluation of the Value constraints. The difference between the two lies in the method of invoking the predicate-name function. The semantics of the Relational constraint type indicates that the specified relation must hold between the constrained activity, and at least one occurrence of the activity specified in the constraint. For example, the sspp-suntracking-constraint specifies that the solstice-solar-observation activity can only occur while the SSPP is in sun-tracking mode. In the schema

representing this constraint, the schema-name attribute has the value sun-track. The Relational constraint evaluation function finds all occurrences of the sun-tracking activity, and applies the predicate-name function to each activity. If the predicate-name function returns T for any of the occurrences, then the constraint is satisfied and no violation is detected. If no occurrence satisfies the constraint (i.e. returns T) then a constraint violation is detected and the constraint notification process begins.

CONCLUSION

The Mission Operations Planning Assistant has proven the applicability of the knowledge-based approach to mission planning. Several important concepts were demonstrated in the prototype; Activity abstraction hierarchies to facilitate multi-level planning , Reverse Plan maintenance, a general mechanism for the representation and evaluation of constraints, and contextual activity priorities.

MOPA has also demonstrated the effectiveness of a frame-based knowledge representation capability. In the early stages of the development of MOPA, we had implemented a large portion of the prototype using a rule-based approach. We discovered that this approach had several weaknesses: lack of structure, difficult to test and verify, difficult to maintain, and it hinders generic development. We believe the frame-based approach used in MOPA is key to developing maintainable and deliverable systems.

ACKNOWLEDGEMENTS

Thanks are due to NASA Goddard Space Flight Center for supporting this work; and to Robert Brekke, Edward Connell, Carolyn Dent, Larry Hull, Allan Jaworski, Patricia Lightfoot, William MaCoughtry, and Dorothy Perkins, for their support and guidance.

References

1. M. Fox, "Constraint Directed Search: A Case Study of Job-Shop Scheduling". Ph.D. Thesis.,Computer Science Dept, Carnegie Mellon University, Pittsburg, PA 15213, 1983.

2. W. Gevarter, "Artificial Intelligence". Noyes Publications, 1984.

3. I. Goldstein and B. Roberts. "NUDGE, A Knowledge-based Scheduling System". The Fifth Internation Joint Conference on Artificial Intelligence, IJCAI, 1977.

4. Inference Corporation, "ART Reference Manual Version 2.0". Inference Corporation, 1986.

5. J. King, "RPMS: Resource Planning and Management System". JAI PCC 1984.

6. J. Schuetzle and D. Zoch, "A Hierarchical Planning System for Mission Support". Proceedings of the IEEE Expert Systems in Government Symposium. 1986.

7. Symbolics Inc. "Reference Guide to Symbolics-Lisp". Symbolics Inc. 1985.

8. D. Zoch, "Maintaining Consistency Between Planning Hierarchies: Techniques and Applications". Proceedings of the NASA Conference on Artificial Intelligence Applications. 1987.

AUTOPLAN - A PC-BASED AUTOMATED MISSION PLANNING TOOL

Frank C. Paterra
Marc S. Allen

Computer Technology Associates, Inc.
Newport News, VA 23606

George F. Lawrence

Langley Research Center
Hampton, VA 23665

ABSTRACT

The Evolutionary Definition Office (EDO) at the Langley Research Center (LaRC) has the responsibility to analyze and evaluate alternative growth options of the Space Station and its utilization. Under contract to the EDO, Computer Technology Associates (CTA) has developed a PC-based automated mission and resource planning tool, AUTOPLAN. AUTOPLAN's input is a proposed profile of missions, including for each: start year, number of allowable slip periods, mission duration, and requirement profiles for one or more resources as a function of time. The user also inputs a corresponding availability profile for each resource over the whole time interval under study. Subject to the size of a given problem and microcomputer performance limitations, AUTOPLAN finds all integrated schedules which do not require more than the available resources.

AUTOPLAN is implemented in Arity compiled PROLOG, and executes on an IBM PC/AT with 640 KB memory. There is particular interest in small-scale planning and scheduling systems in the Space Station program because of the trend toward decentralizing these functions. The iterative resolution and recursion features of PROLOG greatly simplify the programming of this problem, and make it easy to customize or generalize the solution evaluation algorithm. The quantitative capabilities of the tool and several postprocessor interpretive aids presently under assessment are described, and a realistic sample application of the tool suite is presented.

1. Introduction

The Space Station Mission Requirements Data Base (MRDB) contains information on over 300 scientific, technology development, and commercial missions proposed for the Space Station. Each of these missions is characterized by requirements for supporting resources, including crew time for assembly, servicing, and operation, electrical power, thermal dissipation, and communications. The design of the flight segment must be able to allocate these resources among the various payloads installed at any given time. The implication of this allocation is a need for comprehensive resource scheduling and operations coordination.

With Phase B preliminary design studies complete, enough is known about the configuration and build-up of the Space Station to allow meaningful comparisons between the projected demand for resources and the availability of these resources over time. Recent studies have shown that many resource categories will be seriously oversubscribed from the outset of Station activation. A recent McDonnell Douglas Astronautics Company (MDAC) study for the EDO has shown that many of the more resource-intensive payloads will probably have to be operated intermittently if all customers are to be accommodated. Resource scheduling and operations coordination are thus developing into major aspects of integrated Station operations.

Because of the great complexity of the Station system, efficient distribution of available resources will have to be automated to a high degree. The competition of a large number of host and user subsystems, with widely differing requirements and operating priorities and options, for many dissimilar resources will not be adjudicable by manual methods. Worse, an important design objective supporting operational flexibility is replanning on as short a time scale as possible.

These factors suggest that the scheduling and coordination approach likely to be adopted will be a hierarchical one, where gross allocations will be made in a top-down fashion, and detailed schedules will be propagated upwards for integration into master schedules. Naturally, there will be a degree of iteration in this flow. A general approach of this type has the most promise for supporting a telescience operations concept and allowing maximum latitude in detailed planning and operations to end users.

One beneficial effect of a hierarchical approach to resource allocation and scheduling is that the problem is more tractable at each level than a simultaneous, global solution would be. This fits in well with a user accommodation concept emphasizing distributed operations; it also suggests

the need for automated planning tools suitable for end-users. AUTOPLAN can be viewed as a prototype for an automated scheduling tool which is capable of solving complex scheduling problems on a modest hardware configuration likely to be found at any end-user site.

2. AUTOPLAN Algorithm

The algorithm used by AUTOPLAN to search for successful mission sets must take into account the large search space that could be involved. In order to reduce that search space, AUTOPLAN searches for solutions in a recursive-descent, tree-like manner. Each mission is a node in this tree, and the number of branches at that node is the number of slips allowed for that mission. Successive missions on the list are then extension sub-branches. The central looping core of AUTOPLAN is a mixture of both iteration and recursion. A mission is appended to the solution set by iteratively reading and testing successive entries for that mission. If one of these tests is successful, the next mission on the mission list, and thus the next level (node) of the search tree, is tried through recursion. As AUTOPLAN moves down the search tree, it maintains a running sum of resource use of each resource for each time period and compares them against the corresponding resource envelopes; if the partial sum of any resource for any operating period becomes larger than the corresponding available resource, that search path is truncated. This allows AUTOPLAN to discard many failure paths without examining them because the tree is cut at that node and all subsequent branches are summarily removed from the search space. The result of this truncation is a significant improvement in performance for problems with sparse solutions.

Because AUTOPLAN maintains these running sums, and execution time will be reduced if branches close to the root are cut off, an analyst can speed up the searching process by arranging the missions in deceasing resource use order. In addition, since AUTOPLAN saves failure as well as solution sets, an analyst can order the missions by priority, and then examine the failure file for partial mission sets which were able to accommodate at least the high priority missions.

3. AUTOPLAN Implementation

AUTOPLAN was written in compiled Arity PROLOG. Because of the backtracking and unification features of PROLOG and its natural support for list processing and recursion, the tree-oriented search was easily implemented. During design and implementation, it was recognized that the searching algorithm should be optimized for speed. To achieve this,

interpretation and display of solutions and failures have been isolated from the searching algorithm. As a result, the raw output is not very readable without postprocessing. Also, there is no error checking of mission data during the solution search so all data must be entered through a preprocessor. To retain maximum flexibility, AUTOPLAN was developed with no resource or time period limits, other than the computing power of the host computer. Any number of resources and any time period granularity can be accommodated.

To run AUTOPLAN, an input data file is created from mission and resource data using a data set editor. As the program is executed, the input data file is read, and each mission with allowed slips is expanded. Expanding a mission consists of creating an additional entry for the mission for each time period it is allowed to slip. Once all missions have been read and expanded, AUTOPLAN asks the user whether a single solution or all solutions are desired, and then prompts for the type of runtime display. There are three types of runtime displays: Full Graphics, Resource Use Only, or Successes and Failures Only.

Full Graphics Display - This option displays a grid that highlights periods with excessive requirements, a dynamic list for each resource that shows the actual quantity being used in each period, and counts of successes and failures (Figure 2).

Resource Use Only - This option displays a list for each resource being examined and the counts of successes and failures.

Successes and Failures Only - This option displays only the success and failure counts tried so far.

Lastly, the user is prompted for the solution and failure file names, and the search begins. Full solutions and partial lists for failed permutations are saved in these files for input to the postprocessor. Once the last solution is found, the elapsed time is displayed and the program ends. The solutions and failures are viewed using postprocessing software.

4. The AUTOPOST Postprocessor

When AUTOPLAN identifies a solution or a failure, it writes it to the solution or failure file. The solution and failure records in their raw form are not very useful to a human analyst, because they are in a highly compacted, PROLOG-readable form. The postprocessor reads the solution and failure files produced by AUTOPLAN and presents the data to the analyst in a more intelligible form.

The present postprocessor package, AUTOPOST, offers three postprocessors:

Mission Schedules - This produces a chart, similar to Figure 3, graphically showing the operational interval of each mission in a solution set. A separate chart is displayed for each of the solution or failures sets.

Average Resource Use - This produces a histogram for each of the resources showing the average amount, averaged over all successful schedules, consumed in each period. The envelope of available resources is also shown on the report (Figure 4).

Most Efficient Solution - Because by definition all missions must be flown in each schedule solution, the integrated consumption of any resource for all solutions is the same. This postprocessor searches the solution file for the solution whose peak usage of a selected resource is the smallest. In the following example, solution 2 would be selected as the "most efficient" solution.

Solution 1	Solution 2
[10,11,12,9,6,10]	[10,8,11,9,10,10]

Because of the modularity of AUTOPOST's design, additional postprocessors can be added easily as their need is identified by Space Station analysts.

5. Case Study - Technology Development Attached Payloads

In order to conduct an illustrative, yet realistic, analysis with the available data, CTA selected 14 technology development missions, both U.S. and foreign, from the list of 33 attached payload missions grouped together in a recent MDAC evolution study. Technology development missions were selected because of their special interest to the LaRC Space Station Technology Office.

Although AUTOPLAN's full graphics display is capable of handling up to five resource categories at a time, and the central algorithm has no limitation at all, only power and IVA crew time for daily operations were analyzed in this study. Other interesting parameters available in the MRDB, such as physical volume and up- and downmass, need additional information before they can be applied as quantitative schedule constraints.

2.1 ASSUMED MISSION LIST

The mission subset chosen for study consists of Manned Base "attached payloads" not included in any of the other MDAC categories, further qualified by being either U.S. or foreign technology development missions. All SAAX missions, therefore, were excluded, as were all Japanese S-XXX and E-XXX missions.

Some of the 14 missions on the resulting list showed essentially continuous resource requirements after activation; others tended to operate for a year or so at a time, skipping one or more years between operational periods. In defining allowable slips for the missions on the list, the following rule was applied: for missions with continuous operation, allow no slips unless the first operational year is 1994, when resources are especially scarce; for missions beginning in 1994 or having embedded non-operational periods in the schedule, allow one or more slip years.

This principle does not represent any programmatic considerations, but it is conducive to optimum use of resources.

Although the missions in the list are baselined in the MRDB against a 1992 start date, their schedules are all mapped here onto a LaRC Critical Evaluation Task Force (CETF) 1994 resources timeline. This is equivalent to an a priori, uniform two year delay for all missions. Individual slips for selected missions are then applied to this modified schedule, as described above.

The resulting mission list, with assigned allowable slips, is as follows:

Mission	Allowed Slips	Mission	Allowed Slips
TDMX2441	0 years	TDMX2321	1 years
TDMX2011	0	TDMX2574	0
TDMX2132	0	T-007	0
T-001	1	TDMX2542	1
TDMX2061	1	T-008	0
TDMX2153	2	TDMX2541	2
TDMX2311	1	TDMX2543	0

The total number of possible schedule permutations for these input data is:

$$288 = 1^7 \times 2^5 \times 3^2$$

5.2 ASSUMED INDIVIDUAL MISSION RESOURCE REQUIREMENTS

This study considers only payload requirements for power and daily IVA crew time. Crew time requirements for setup, servicing, reconfiguration, and teardown are not addressed. The data used were extracted from the EDO MRDB (EDOM), using a partially implemented data extraction program.

In order to compute daily requirements for these two resources, it is necessary to interpret the contents of the data base. For power, a conservative algorithm is applied: if a given payload is operational on any day in a given year, it is assumed to draw resources every day of the year. Standby, normal operating, and peak demands are combined in proportion to their time fractions as tabulated in the data base. For crew time, a more liberal algorithm is applied: it is assumed that crew needs can be scheduled against one another within each year period in such a way as to minimize conflicts. That is, the mean daily crew time for a given payload over a year is computed by prorating the requirement per operational day by the number of days in that year that the payload was in operation. It should be noted that either algorithm, i.e., the conservative or liberal one, could be applied to any resource. Or, an average requirement could be computed from the two algorithms. The choice amounts to an emulation of the results of scheduling at a more detailed level.

Figure 1 presents the power and crew data extracted from the EDOM, as shown in a report produced by the EDOM Mission Analysis Tool (EMAT) software.

5.3 ASSUMED TOTAL USER RESOURCE ENVELOPES

The CETF briefing presents profiles for total user allocations of both power and IVA crew time; the resources provided to attached payloads must be a subset of this total user allocation, and technology development attached payloads will, in turn, be allowed a portion of this subset.

For power, a CETF graph shows approximately 20 KW available to all users until the solar dynamic generating system is installed in the last quarter of 1994. Then the user power resource increases to about 70 KW, from which it gradually declines because of mounting system requirements for the growing Station to about 60 KW in 1997. For the first year, 1994, the average total user power is then:

$$32.5\ \text{KW} = 0.75 \times 20\ \text{KW} + 0.25 \times 70\ \text{KW}$$

Note that this method of resource combining is only a compromise between conservative and liberal approximations, and does not strictly represent detailed scheduling within

the one year time period granularity. For example, this computation suggests that a solitary 25 KW device could be operated all year, whereas it actually could be operated for only the last three months. Similarly, it incorrectly suggests that a single 60 KW experiment could not be operated at all. This inaccuracy in accounting for scheduling within the finest time granularity can be reduced by using finer time divisions, but it will also be reduced for cases with less abrupt changes in resource availability or for situations where the most demanding resource sinks absorb a smaller fraction of the total available.

For 1995, the average total user power is 70 KW. It is assumed that there is a linear decrease for the next two years to 60 KW, after which, in the absence of additional information, total available user power is assumed constant. Further increases in system requirements might be offset by improved efficiency, addition of capacity or other augmentations. This leads to the following profile for total user power, expressed in KWhour/day:

Year	KW	KWhour/day
1994	32.5	780
1995	70	1680
1996	65	1560
1997-2003	60	1440

A similar argument is adopted for IVA crew time. From the CETF briefing, there is no permanent crew presence until the final third of 1994. From then through the first third of 1995, the graph indicates about 72 hours per week available to all users. From then until the end of the first third of 1996, the total weekly allocation is 128 hours. Finally, after that point, 244 hours per week of IVA crew time are available for sharing by all users. The necessary computation for 1994 is:

$$24.0 = 0.33 \times 72 \text{ hours}$$

For 1995:

$$109.3 = 0.67 \times 128 \text{ hours} + 0.33 \times 72 \text{ hours}$$

For 1996:

$$205.3 = 0.67 \times 244 \text{ hours} + 0.33 \times 128 \text{ hours}$$

For 1997 and later years, the answer is simply 244 hours per week of IVA crew time for all users. Converted to IVA crew hours per day for all users, this is:

Year	Hours/week	Hours/day
1994	24.0	3.4
1995	109.3	15.6
1996	205.2	29.3
1997-2003	244.0	34.9

5.4 ASSUMED TECHNOLOGY DEVELOPMENT RESOURCE ALLOCATION

As hypothesized above, the technology development attached payload user community can expect to be allocated only a fraction of the total resource pool available to all users. The precise fraction must be set on policy grounds. Other user groups include science and commercial users, and, within each of these groups, attached payloads must compete with laboratory equipment and servicing requirements. This case study assumes that the technology attached payload community will be allocated 20%. The resulting distributions for power and crew IVA time are listed in the following tables:

YEAR	POWER (KWhour/day)	CREW (IVA Manhour/day)
1994	156	0.7
1995	336	3.1
1996	312	5.9
1997-2003	288	7.0

It is evident that IVA crew time is an especially scarce resource, especially in the first two years of manned operations. The crew requirement analyzed here includes only regular periodic operations, and omits IVA needs for setup, configuration changes, servicing, and teardown. According to Figure 1, five of the missions on the list do not require any of this periodic crew activity.

5.5 RESULTS

Performance figures given in the discussion are based on execution on a PC/AT with 1.1 MB RAM; 512 KB of this RAM are configured as RAM-disk containing the input data set and PROLOG's dynamic data base. Additional, but small, performance improvement is possible by writing the output success and failure data sets also to RAM-disk.

For the resource availability assumptions used, the number of failures found is 95, with only 4 possible solutions out of 288 possible permutations. AUTOPOST displays of the solutions are shown in Figures 3 and 4. Figure 3 shows the feasible mission timeline for each solution, and Figure 4 shows the total resource consumption as a function of time,

averaged over all missions. This second plot, which also lists and displays the limiting resource envelope, is helpful in gaining a qualitative understanding of the critical resources and critical times. Execution time required was 2 minutes and 3 seconds.

Two additional cases were run to observe AUTOPLAN's behavior in discarding solutions for this data set as the envelopes were shrunken. The algorithm concluded that no solution was possible if only 14% of the total resources are allocated to this mission set. This was determined after examining 77 profiles of the 288 in 1 minute and 23 seconds.

In the extreme case of only 12% allocation, the first examined mission with a non-zero crew requirement, TDMX2132, required 0.5 manhours per day. This could not be provided by the total crew allocation. Since this mission was not allowed any slippage, AUTOPLAN terminated its search with this single failure in less than a second.

5.6 CASE STUDY CONCLUSIONS

The results presented for this illustrative example allow a number of conclusions to be drawn about the assumed scenario for scheduling the technology development attached payload missions and about AUTOPLAN and its use.

- The feasibility of an integrated schedule is critically dependent on the resources available. A change of even a few percent can mean the difference between many choices in schedule and no solutions at all.

- Separated from other mission groups as done here, the assumed model of the technology development community would need approximately 20% of total power and daily IVA crew time available to users. This does not include requirements for setup, servicing, and teardown. Of course, it is not necessary that the percentage allocation of different resources (e.g., power and crew) be the same.

- Crew requirements for setup, servicing, and teardown should also be included in the over-all crew requirement evaluation.

- Contingency margins should be subtracted out of allocated envelopes.

- Policy guidance should be available to help assign slip allowances, if realistic results are to be obtained.

- The fidelity of the results are only as accurate as the input resource budgets and input mission requirements;

determining these are the limiting factors for study accuracy.

- AUTOPLAN exhibited adequate performance for a realistic problem on a widely available equipment configuration. Although the processing power of the PC/AT does limit the size of problems which can be solved at once, it is adequate to support useful analyses at individual levels in a hierarchical allocation model.

- AUTOPLAN enables the mission analyst to perform schedule evaluations not possible by other means.

- Postprocessor functions are the key to making the results useful; these functions need not be limited to simple displays, but could include additional logical or arithmetic operations, since the solution data set fully characterizes all solutions. One candidate is a precedence filter which could select all solutions wherein certain missions are completed before initiation of others.

- The code is not restricted to problems based on ten, one-year time intervals; that is, it could evaluate daily schedules over a month, or hourly schedules over a day. However, a flexible data set editor is required to simplify input data set construction for input data other than standard MRDB timelines. Development of such an editor is planned as a follow-on activity.

6. Future Plans

Although AUTOPLAN is capable of analyzing data from any source, its use is presently restricted by limited support tools to data extracted from the EDOM reorganized version of the MRDB. Several postprocessor functions are also in place, as illustrated in the case study. Enhancements to the tool suite can be roughly divided into three groups: extensions to the algorithm itself, improvements in input data set construction, and addition of postprocessing functions.

The basic functionality of AUTOPLAN's search and evaluation algorithm is very general. Most enhancements to the main program are likely to provide additions to reporting of search by-product information for postprocessor use, or alternatives to the evaluation algorithm. For example, the simple add-and-compare test for schedule viability could be replaced by a more sophisticated test. In general, however, increased complexity in the core processing will degrade performance in searching the candidate solution space. Wherever possible, enhancements should be implemented through pre- or postprocessor functions.

One of the major limitations of the present package is difficulty getting data into AUTOPLAN. No software is currently available to help a user edit input data extracted from the EDOM: mission parameters must be used exactly as contained in the data base, even if the user has better information. Secondly, although AUTOPLAN is capable of solving entirely different problems, for example, scheduling individual hours during the day or individual days during the month, no tool is available to construct input data sets from scratch. It is expected that capabilities in both of these areas will be developed during a follow-on task.

Finally, a diversity of postprocessors will be needed to support the needs of different analysts. The postprocessors implemented so far are simply the most obviously useful. Since AUTOPOST operates on only known solutions, brute performance is not the over-riding concern that it is for the AUTOPLAN search algorithm. As a logic programming environment, PROLOG facilitates the construction of complex logical inferencing functions. New modules could be implemented and integrated easily into the AUTOPOST framework. Use of the AUTOPLAN/AUTOPOST package on real Space Station problems is expected to suggest numerous useful extensions.

This work was performed under contract NAS1-18247.

FIGURE 1 - EDOM POWER AND CREW REQUIREMENTS

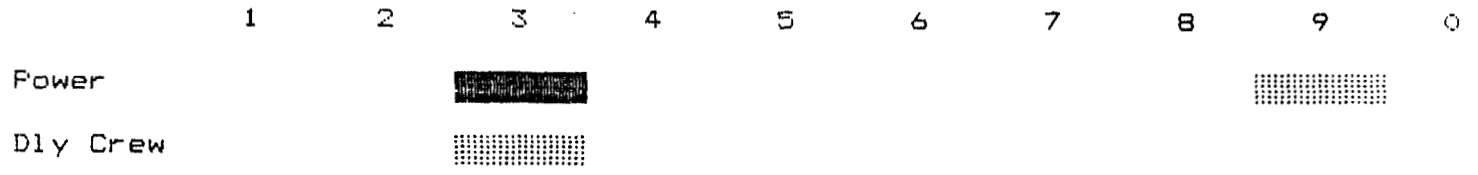
Table of power use (KWH/Opnl Day)

Mission	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001
TDMX2441	6.1	6.1	6.1	6.1	6.1	6.1	6.1	6.1	6.1	6.1
TDMX2011	11.2	11.2	11.2	11.2	11.2	11.2	11.2	11.2	11.2	11.2
TDMX2132	18.0	18.0	18.0	18.0	18.0	.0	.0	.0	.0	.0
T-001	24.0	24.0	24.0	24.0	24.0	24.0	24.0	24.0	24.0	.0
TDMX2061	6.0	6.0	.0	.0	.0	.0	.0	.0	.0	.0
TDMX2153	48.0	48.0	48.0	.0	.0	.0	.0	.0	.0	.0
TDMX2311	.0	24.0	24.0	.0	.0	.0	.0	.0	.0	.0
TDMX2321	.0	20.0	20.0	.0	20.0	.0	20.0	.0	.0	.0
TDMX2574	.0	.0	7.2	.0	.0	.0	.0	.0	.0	.0
T-007	.0	.0	.5	.0	.0	.0	.0	.0	.0	.0
TDMX2542	.0	.0	54.0	.0	54.0	.0	54.0	.0	54.0	.0
T-008	.0	.0	.0	.0	.0	7.2	7.2	.0	.0	.0
TDMX2541	.0	.0	170.0	.0	.0	170.0	.0	.0	170.0	.0
TDMX2543	.0	.0	.0	.0	.0	56.0	56.0	56.0	56.0	.0

Table of crew use (Avg hours/day)

Mission	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001
TDMX2441	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
TDMX2011	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
TDMX2132	.50	.50	.50	.50	.50	.00	.00	.00	.00	.00
T-001	.50	.50	.50	.50	.50	.50	.50	.50	.50	.00
TDMX2061	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
TDMX2153	.50	.50	.50	.00	.00	.00	.00	.00	.00	.00
TDMX2311	.00	.99	.99	.00	.00	.00	.00	.00	.00	.00
TDMX2321	.00	.10	.10	.00	.05	.00	.10	.00	.00	.00
TDMX2574	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
T-007	.00	.00	.11	.00	.00	.00	.00	.00	.00	.00
TDMX2542	.00	.00	.82	.00	.82	.00	.82	.00	.82	.00
T-008	.00	.00	.00	.00	.00	1.00	1.00	.00	.00	.00
TDMX2541	.00	.00	1.97	.00	.00	.66	.00	.00	.66	.00
TDMX2543	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00

FIGURE 2 - FULL GRAPHICS DISPLAY



[41.305,157.305,383.005,107.305,133.305,218.505,122.505,41.305,265.305,41.305]

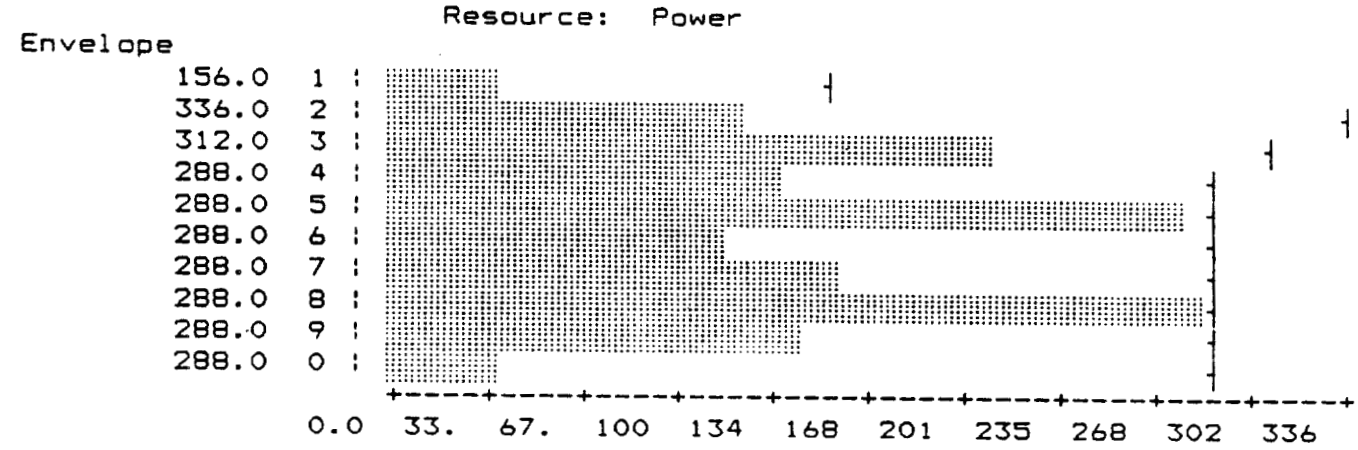
[0.5,2.585,5.49,1.5,1.871,2.158,2.421,0.5,1.98,0.5]

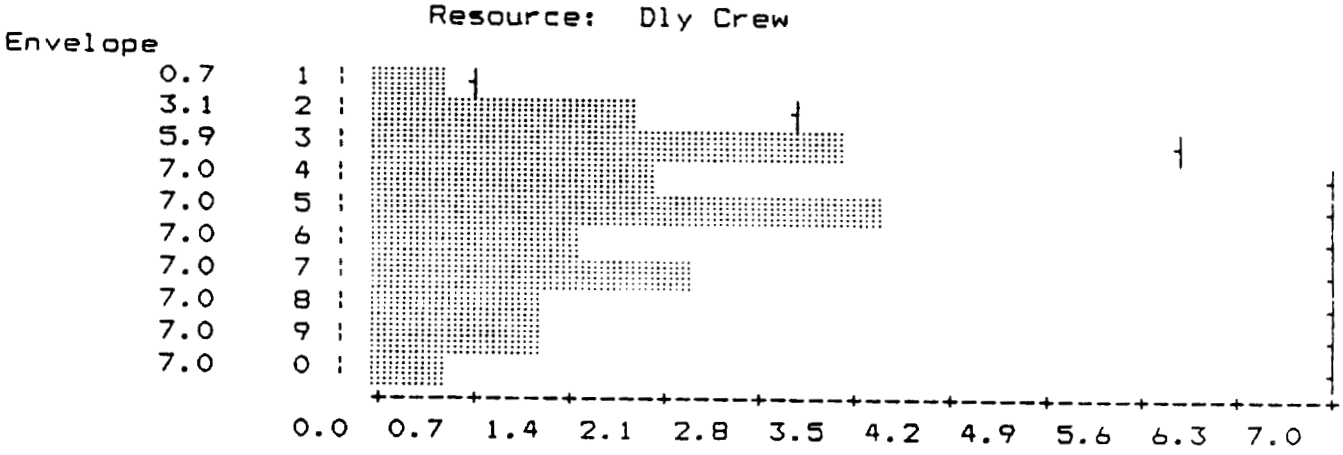
Failures: 4
Starting time - 11:56:29

FIGURE 3 - SOLUTION TIMELINES

```
              1       2       3      4       5      6       7      8       9       0
---------------------------------------------------------------------------------------
TDMX2543                                          *******************************
TDMX2541                                  *******                *******
T-008                                             ***************
TDMX2542                 *******        *******        *******        *******
T-007                    *******
TDMX2574                 *******
TDMX2321                 ***************        *******        *******
TDMX2311          ***************
TDMX2153          **********************
TDMX2061  ***************
T-001             *********************************************************************
TDMX2132  **************************************
TDMX2011  *****************************************************************************
TDMX2441  *****************************************************************************

---------------------------------------------------------------------------------------
TDMX2543                                          *******************************
TDMX2541                                  *******                *******
T-008                                             ***************
TDMX2542                 *******        *******        *******        *******
T-007                    *******
TDMX2574                 *******
TDMX2321                 ***************        *******        *******
TDMX2311                 ***************
TDMX2153          **********************
TDMX2061  ***************
T-001             *********************************************************************
TDMX2132  **************************************
TDMX2011  *****************************************************************************
TDMX2441  *****************************************************************************

---------------------------------------------------------------------------------------
TDMX2543                                          *******************************
TDMX2541                                  *******                *******
T-008                                             ***************
TDMX2542                 *******        *******        *******        *******
T-007                    *******
TDMX2574                 *******
TDMX2321                 ***************        *******        *******
TDMX2311          ***************
TDMX2153          **********************
TDMX2061          ***************
T-001             *********************************************************************
TDMX2132  **************************************
TDMX2011  *****************************************************************************
TDMX2441  *****************************************************************************
---------------------------------------------------------------------------------------
TDMX2543                                          *******************************
TDMX2541                                  *******                *******
T-008                                             ***************
TDMX2542                 *******        *******        *******        *******
T-007                    *******
TDMX2574                 *******
TDMX2321                 ***************        *******        *******
TDMX2311                 ***************
TDMX2153          **********************
TDMX2061          ***************
T-001             *********************************************************************
TDMX2132  **************************************
TDMX2011  *****************************************************************************
TDMX2441  *****************************************************************************

---------------------------------------------------------------------------------------
```

FIGURE 4 - AVERAGE TOTAL RESOURCE CONSUMPTION AND LIMITS





PLAN-IT: SCHEDULING ASSISTANT FOR SOLAR SYSTEM EXPLORATION

William C. Dias
ITT Federal Electric Corp.

Julia A. Henricks
Jet Propulsion Laboratory
California Institute of Technology

Jennifer C. Wong
Jet Propulsion Laboratory
California Institute of Technology

ABSTRACT

At the Jet Propulsion Laboratory (JPL), the scheduling of spacecraft activities is a complex endeavor for which streamlining is always being sought. Using the Comet Rendezvous Asteroid Flyby mission (CRAF - proposed for 1993 launch) as a development base, PLAN-IT (a frame-based expert scheduling system shell) has been adapted to assist in one phase of schedule generation. PLAN-IT CRAF automatic scheduling routines attempt to return a 'good-cut' schedule which can be adjusted by an expert with time-saving graphic manipulation tools. Work to date has led to a "GO" decision on technical factors for online capability development.

INTRODUCTION

The scheduling of spacecraft activities at the Jet Propulsion Laboratory (JPL) is a very involved process requiring many personnel over long time periods (1,2). The use of specialized automation tools has evolved and been encouraged over the years to increase the effectiveness and responsiveness to changes in the scheduling processes (3). Although the use of automation in the scheduling process reduces the workload of scheduling personnel, a main benefit is to gather more science data by allowing the scheduling of as many activities as possible while controlling mission risk.

Recent interest has increasingly focused on the potential of artificial intelligence (AI) to achieve those ends (4). Future spacecraft schedules will have a greater need for advanced scheduling automation tools because of enhanced spacecraft capabilities and the desire to automate tasks which are now labor intensive.

The scheduling of spacecraft activities may be divided into five phases (5):

1)	Request Generation	Collection and preprocessing of requirements
2)	Request Integration	Integration of requests into a timeline
3)	Activity Detail Design	Refinement of detail in schedule

4)	Sequence Generation	Translation of plan to a sequence of commands
5)	Command Generation	Translation of commands to program load

A scheduling program called PLAN-IT, short for Plan-Integrated Timelines, provides general interactive scheduling capabilities using AI techniques. It has proved to be applicable to spacecraft scheduling in the Request Integration Phase. The test case for the study of this applicability was the Comet Rendezvous Asteroid Flyby (CRAF) mission and is the subject of this paper.

BASIC, STRUCTURED, AND EXPERT SCHEDULING TECHNIQUES

Recent literature has pointed out advantages to "evolutionary delivery" or other staged delivery of complex, innovative software systems (6,7). This paper takes the view that construction of an Expert System for scheduling benefits by this treatment. A scheme for evolution of expert functionality is outlined in this section.

The art of scheduling encompasses many techniques and methods for generating a "good" schedule. Schedulers' techniques fall into three categories: Basic, Structured and Expert. One approach to development of an Expert System for a given scheduling application is to evolve from a simple to a more complex level of support. This can be done by increasing the level of intelligence being represented in the system, progressing from the Basic through the Structured and finally the Expert categories of support as outlined below. Attempts to develop Expert Systems from manual systems or those automated at only a Basic support level may incur excessive and unnecessary development risk. An Expert System should encompasses not only the Expert techniques, but also the Basic and Structured techniques as described below.

Basic

The Basic category of scheduling techniques emphasizes manual control of basic operations performed on individual activities. Basic techniques focus on a single activity and not on how this activity affects other activities. ***Knowledge*** of how a decision affects the schedule may be represented in the system and displayed graphically. That knowledge does not ***control*** execution of a command at the basic level, but may be used to report information to the operator. Using this knowledge, the operator controls execution. One of the effects of this is allowing systems to be partially functional in real environments with less completeness in the system internal knowledge.

The Basic category encompasses three functional operations: move an activity to another time frame, delete an activity, and add new activities to the schedule.

This category emphasizes manual execution which gives the user the flexibility to move activities at his discretion. The user scans the schedule and manually moves an activity to an area of lesser conflict. The strength of these techniques is best illustrated where it is desired to alter a schedule which has already been generated.

These Basic techniques are widely used in scheduling. Systems which address this category of techniques can be applied in a wide variety of scheduling situations. An expert scheduling system 'shell' should include these basic operations.

Structured

The Structured category incorporates more knowledge of the attributes of an activity. System internal knowledge controls execution of functions to a greater degree than with the Basic category. Examples of knowledge attributes include separation criteria or precedence relations among activities. The Structured category also operates coherently on groups of activities or combines the effects of several basic techniques.

The following examples of structured operations show why more knowledge about activities must be present in the system for correct execution:

1. Move activity A after activity B.
2. Adjust separation times between activities of type C to a maximum of 20 minutes and a minimum of 5 minutes.
3. Insert activity D between activities E and F, and shorten its duration accordingly.
4. Repeat activity G at the same time, every day of the week.
5. Reschedule activity H at time Z, and automatically reschedule activities logically dependent on activity H accordingly.

It can be seen that operations in the Structured category are more specific than those in the Basic category. There may be a subset of operations which are applicable to broad generic classes of scheduling. For instance, Operation 5 could be included in a generic scheduling tool devoted to handling problems with inter-activity temporal dependencies such as on a Pert Chart. The operations in the Structured category are usually very specific for a particular application.

Expert

The Expert category is characterized by operations which:

a) Encompass many activities
b) Include heuristics or 'rules of thumb' for simplifying scheduling problems
c) Generate "good" schedules by complex rules, such as:
    1. Rearranging the schedule to minimize total conflict.
    2. Rearranging the schedule to minimize the variance in work loads over time.
    3. Delete activities, as necessary until conflicts are eliminated.

There is no hard and fast division between Structured category techniques and Expert category techniques. Evolution from the Structured category to the Expert category is a low-risk means of

evolving towards an Expert System for a specific scheduling application.

PLAN-IT OVERVIEW

This section provides an overview description of PLAN-IT, the tool used to develop the scheduler for the CRAF test case. The term "developers" used in this context refers to the people customizing PLAN-IT for a particular application. The term "user" refers to the person who will use the customized version of PLAN-IT to generate schedules.

PLAN-IT is an interactive scheduling tool. PLAN-IT is written in Zetalisp on a Symbolics 3640 using version 6.1 of the operating system. PLAN-IT utilizes the frame features and object oriented programming capability provided by the Zetalisp FLAVOR system. The Symbolics has a high-resolution graphic screen and a three key mouse which is utilized by PLAN-IT to enhance the user interface.

The PLAN-IT screen (see Figure 1) graphically displays the activities which are scheduled and the resources the activities utilize. The PLAN-IT screen graphically represents the methods experts use to lay out a schedule. The upper portion of the PLAN-IT screen displays activities as a series of horizontal lines. Resources are represented as horizontal rectangles (thick rectangular bars) directly below the activity area. A white area in the resource line indicates that a resource is not allocated; gray implies utilization within the bounds of a given maximum; black indicates an interval of oversubscription or conflict.

Activities are internally represented with frames (8,9). Resources are internally represented as blackboards (8,10). Each resource line maintains internal lists of conflict and usage ratings for each interval of time. These interval bounds are determined as the point where usage differs. Any change (permanent or provisional) to an activity forces maintenance of the associated blackboards with an optional screen update.

PLAN-IT 'strategies' are blocks of code which, when invoked, assist the user in resolving or analyzing conflicts. These strategies are initiated from a pop-up menu.

Developers customizing PLAN-IT to a given application must have knowledge of PLAN-IT's internal functions and a background in Zetalisp. Software to interpret the input file and to display the desired output to the end user must be generated for each unique application. Each resource timeline is defined by utilizing a series of macros developed specifically by the original PLAN-IT designers. Code which defines and displays conflict on the resource blackboards must be written, again with help of PLAN-IT macros. The next step is the design and implementation of software strategies to assist the user in generating a "good" schedule. Since PLAN-IT is an ongoing development, changes to the core code (i.e., below the level of the macros), may be required in a specific application when conditions are not fully representable in the macros.

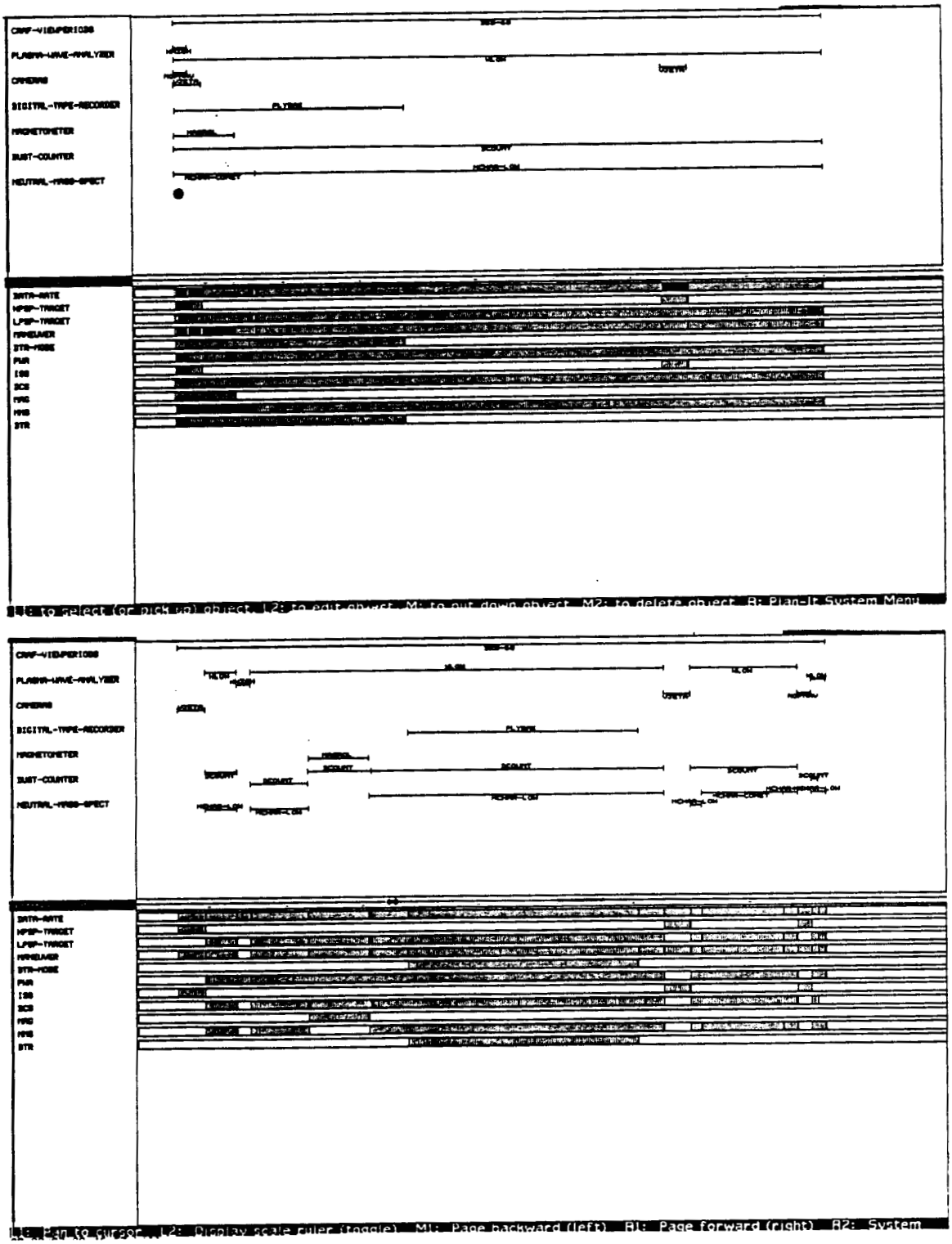


FIGURE 1 PLAN-IT SCREEN Conflicts are indicated by the solid-black bars. Top screen shows a single track as initially loaded by PLAN-IT. Bottom screen shows the same track after the conflicts have been removed.

CRAF PROBLEM OVERVIEW

The CRAF mission has a proposed launch date of 1993 (11). After a cruise of several years with several gravitational assists from close flyby of planetary bodies and an asteroid encounter, CRAF will encounter the comet Tempel 2 near the orbit of Jupiter in 1996. After conducting preliminary science activities, CRAF will accompany the comet past perihelion (closest approach to the sun) in 1999. The early encounter phase is expected to have power resource constraints due to the combined effects of spacecraft dependency on solar power while distant from the sun, battery storage, and a small nuclear power source.

The spacecraft instruments are housed on two separately movable scan platforms, on a stationary boom, and on the spacecraft body. The instruments on platforms are fixed to those platforms, thus it is the platform that is slewed to achieve the proper orientation for a requested observation, not the instrument itself. These instruments collect science, engineering and navigational data. This data is collected at specific data rates which must either be recorded on tape or communicated to the ground via antennas in the Deep Space Network (DSN).

Our test set consisted of nine hypothetical requests for science, navigation and engineering activities over a one week period near Tempel 2 perihelion. Each request represented multiple occurences of an activity, so the nine requests expanded to 77 activities. The activities had requirements and constraints on instrument usage, the platform pointing position, spacecraft orientation, data rate, and separation time. One request required use of the digital tape recorder (DTR) for record and playback. Ground rules for this demonstration stipulated that all activities were to be performed during the nine hypothetical CRAF DSN tracking viewperiods (See Figure 2). The data rate constraints in those tracks place limitations on the DSN link capabilities and tape playback speeds.

From the nine requests, the developers extracted the details implied by the requests and derived the definitions for conflict and constraint violation conditions. The following is a description of the extracted details.

First, of the nine requests, two were multi-phase. These two multi-phase requests were expanded into five single-plase requests for a total of twelve requests. A description of the twelve requests and their attributes are found in Figure 3.

Second, requests fell into two categories: short- and long-duration. The short-duration activities were performed one or more times per week. These short-duration activities had a higher priority than activities which were requested continuously throughout the one-week period.

In contrast, the long-duration activities were requested to be continuous. Portions of these activities could be suspended in favor of the short-duration activities.

DEEP SPACE STATION	ANTENNA DIAMETER (METERS)	DATA RATE (KBPS)	NUMBER OF TRACKS PER WEEK	DURATION OF EACH TRACK (HOURS)
14	70	115.2	1	8
15	34	40.7	3	8
43	70	115.2	1	8
45	34	40.7	2	8
63	70	115.2	1	8
65	34	40.7	1	8

FIGURE 2 DEEP SPACE NETWORK TRACKING ASSIGNMENTS FOR CRAF TEST BED

ACTIVITY	INSTRUMENT	PLATFORM	MANEUVER	TARGET	DATA-RATE (BPS)	DURATION (MIN)	PERFORMANCE CONSTRAINTS
VTARCAL-RT	ISS	HPSP	STAT	STAR-FIELD	115.2K	60	once
VTARCAL-RCD	ISS	HPSP	MNVR	CAL-PLATE	115.2K	120	once
VTARCAL-PB	---	---	STAT	---	100.8K	170	after VTARCAL
MAGROL	MAG	---	MNVR	---	400	45	once
NOPNAV	ISS	HPSP	STAT	STAR/COMET	115.2K	10	once/track
DCOUNT	DC	LPSP	STAT	COMET	44	cont	(1)
VJETA	ISS	HPSP	STAT	COMET-JET	115.2K	20	twice/track(2)
MCHAR-C	NMS	LPSP	STAT	COMET	1000	60	once/track
MCHAR-A	NMS	LPSP	STAT	ANTI-COMET	1000	60	once
MCHAR-L	NMS	LPSP	STAT	COMET	300	cont	---
WLOW	PWA	---	D-C	---	200	cont	---
WHIGH	PWA	---	D-C	---	115.2K	10	once/track

DEFINITIONS:

Cont - Continuous

DC - Dust Counter

D-C - Don't Care, doesn't matter if spacecraft is maneuvering or stationary

HPSP - High Precision Scan Platforme

ISS - Imaging Subsystem (cameras)

LPSP - Low Precision Scan Platform

MAG - Magnetometer

MNVR - Spacecraft required to manuever

NMS - Neutral Mass Spectrometer

PWA - Plasma Wave Analyzer

STAT - Spacecraft required to be stationary

(1) LPSP pointed at the comet 50% of time as well as at least once every two hours for 10 minutes

(2) Occurences within each track to be separated by at least 5 hours but not more than 7 hours

FIGURE 3 CRAF TEST BED ACTIVITY REQUIREMENTS

Third, from the twelve requests, definitions for conflict and constraint violation conditions were derived. Conflicts occurred when activities were scheduled in a time frame which required more resources than were available.

An example of an instrument usage conflict is two activities requiring the same instrument to be scheduled at the same time. An instrument can only accomodate one activity at a time.

An example of a constraint violation can occur with respect to spacecraft movement. An activity requiring the spacecraft to be stationary during its observation cannot be scheduled in the same time frame as an activity requiring the spacecraft to be maneuvered. A more detailed description of conflict and constraint definitions is included in the next section.

PLAN-IT CUSTOMIZATION FOR CRAF

This section discusses the CRAF test case with respect to PLAN-IT. The representations of the various data elements such as activities, resources, conflicts and constraints are discussed below. The different scheduling technique categories (Basic, Structured, Expert) added to the PLAN-IT repertoire, and the rationale for selecting these techniques are reviewed.

Personnel Background

Three engineers developed the PLAN-IT overlay code for the CRAF test case over a time span of seven months. The seven month implementation period included: 1) gaining experience in LISP programming, 2) familiarization with the PLAN-IT core code, 3) defining the CRAF-specific representation and processing requirements, and 4) coding of those requirements. The engineers combined work experience included spacecraft scheduling and programming for resource allocation problems. The original PLAN-IT designers were available for consultation throughout the seven months, as were CRAF spacecraft engineers. Development efforts culminated in a PLAN-IT CRAF proof-of-concept demonstration in October 1986, followed by informal CRAF project recognition of the product's overall potential.

Activity Representation

Activities were represented as frames with slots for each specific attribute of the activity. The attributes were a unique identifier, the start/stop times, the duration of the activity, maneuver status, platform targetting, data rate, and time windows. PLAN-IT created the activity frames as it parsed the input request data file.

Resource Conflict Representation

Resources were internally represented as blackboards. Each resource timeline contained time intervals, conflict and usage ratings. As activities were moved within the schedule, the resource blackboards updated themselves accordingly. Following is a description of each resource utilized by the CRAF activities and a description of the

conflict and constraint conditions that were applicable to each resource.

Instrument usage was represented on the **INSTRUMENT USAGE** lines (PWA, ISS, DCS, MAG, NMS resource lines in Figure 1). A unique resource line was assigned to each of the five instruments. The conflict representation associated with each of the instrument resource timelines was correspondingly simple - a single instrument usage could be requested by only one activity within any time frame. Graphically the resource timelines displayed white when the instrument is not used, gray when the instrument was requested for usage by one activity, and black when several activities requested the same instrument in the same time frame. Dedicated instrument use was assumed to be required for all CRAF activities, although in practice, scientists occasionally use the same data for multiple experiments.

The data rate resource was represented on the **DATA RATE** line. Each DSN antenna is characterized with the ability to support a spacecraft track within its maximum data rate capacity which is determined by equipment configuration and distance to the spacecraft. Figure 2 shows the data rate capacity for each test DSN track. All activities were scheduled within the given data rate envelopes. The DSN antennas can support several activities in the same time interval provided the sum of the activities' data rates does not exceed the maximum DSN link capacity. The DATA RATE resource line showed white when no data activity took place, gray when some bandwidth was used, and black when the maximum data rate was exceeded.

Targetted usage of instruments on the Low Precision Scan Platform (LPSP) was represented on the **LPSP TARGET** line. These instruments had very general targetting criteria. General targets were symbolized by the target name, for example, EARTH, STAR and COMET. Activities were in conflict when they disagreed on general pointing requirements.

The **LPSP TARGET** resource line showed gray when an instrument on the low precision scan platform had targetting requirements. Thus, the resource line showed white when none of the LPSP instruments was used or if an LPSP instrument was in use but untargetted. Black showed when conflicting targets were requested for the same time.

Usage of instruments on the High Precision Scan Platform (HPSP) was represented on the **HPSP TARGET** line. In contrast to the LPSP, the HPSP instruments (i.e., the cameras) had such exact targetting criteria that no two independently requested activities could conjointly utilize the same platform orientation. Representing exact targetting criteria (azimuth and elevation angles) would require a significant coding effort beyond the coding assistance provided by the PLAN-IT macros.

For the above reasons, the **HPSP TARGET** line showed a conflict whenever the high precision scan platform supported two different activities concurrently, regardless of how they were targetted. In this data set, the result was somewhat trivialized by the fact that only one instrument was present on the HPSP, unlike the actual plan for CRAF which has a complement of scientific and engineering instruments on the HPSP.

Maneuver status was represented on the **MANEUVER** resource line. Activities had one of the following maneuver states: YES maneuver the spacecraft, NO don't maneuver the spacecraft or DON'T CARE whether the spacecraft is manuevered or stationary. The assumptions made were:

(a) No two activities which required the spacecraft to maneuver could share the same maneuver
(b) No activity which required the spacecraft to be stationary could coexist with an activity which maneuvered, though several stationary activities could co-exist with each other
(c) Activities which don't care about maneuvering could coexist with any other activity.

Maneuver conflict code was written which stated that if two activities of the type (a) or (b) occurred, a conflict was displayed on the MANEUVER resource line. Portions of the maneuver resource line showed gray if one or more activities requiring a stationary spacecraft were concurrent, or if a single activity requiring a maneuver was scheduled. Any activity with LPSP or HPSP targetting requirements required, by default, a stationary spacecraft unless a maneuver was specifically requested (e.g., VTARCAL).

The assumptions detailed above were in accord with the test case. Realistically, the following qualifiers should be noted:

(a) Maneuvers, such as spacecraft roll maneuvers, could sometimes be shared by several activities.
(b) The degree of "hardness" of the stationary spacecraft requirement varies.

The **DTR-MODE** resource line was coded to show white when no activity was taking place on the tape recorder, light gray when recording was scheduled, dark gray when data was played back, and black when recording and playback were scheduled in conflict.

Resource Constraint Representation

The other major constraints included minimum and maximum separation times between activities, percentage of time pointed at specific targets, and separation time between targetted intervals of a given activity. These constraints were not represented on the resource timelines, so conflict was handled by coding special functions to run from a menu as discussed below. This meant that the code to detect these conditions could not be integrated with other interactive processing or other strategies through the PLAN-IT blackboard system.

Separation time constraints were handled by the PLAN-IT input preprocessor. It set up time windows during which activities could take place. Thus if activity-A was required to precede activity-B by 5 to 7 hours, and activity-A was placed by the preprocessor at 10:00 AM, it placed activity-B at 3:00 PM with an associated two-hour window ending at 5:00 PM. In this 'stationary' time window, if something moved activity-A, there was no mechanism to move the window for activity-B; that would be a 'dynamic' time window, which PLAN-IT could not support except at the expense of considerable CPU time. Although code for it could be written, iterative strategies would run much

longer. The following is a description of three constraints of this type.

Target Time Percentage Constraints were requirements to track a certain target with a certain instrument for a definite minimum percentage of the time. The underlying, unstated requirement was that the requester desired full time tracking, knowing it could not be achieved, but wanted to register the fact that when the activity was suspended, the need to resume that tracking increased with time.

Total Time Percentage Constraints were similar to Target Time Percentage Constraints, except they were untargetted. There was some overlap in coding these constraints.

Separation Time Constraints took the form of repetition requirements. Repetitive activities were required to be separated by either a minimum or a maximum amount of time or both. Again there was an overlap in coding with the constraints above.

CRAF Scheduling Categories

PLAN-IT did not contain the techniques described in the Basic, Structured and Expert categories. PLAN-IT did provide the structures for gathering the data required for implementation of the different scheduling categories. The three scheduling categories are described with respect to the PLAN-IT CRAF test case. PLAN-IT provided assistance in the development of the ***Basic Category*** techniques by providing a method by which the Basic techniques could be coded in LISP and interfaced with the existing core of code.

The Basic techniques developed for CRAF have not yet been incorporated into generic PLAN-IT. The program's good graphic and mouse interface, when combined with application code, enhances user friendliness.

The PLAN-IT CRAF test case emphasized features which gave the user full control while generating a schedule. The Basic category operations designed and coded for PLAN-IT CRAF included:

E1 - Move moused activity to moused time.
E2 - Delete moused activity.
E3 - Examine moused activity detail.

PLAN-IT facilitated development of the ***Structured Category*** techniques in the same manner as it did for the Basic category techniques. Broad classes of scheduling problems (such as a Pert chart application) have not been implemented in generic PLAN-IT. So, the developers wrote the software to accomodate the Structured techniques. The following is a description of some of the operations implemented for PLAN-IT CRAF in the Structured category.

DELETE CONFLICTING PART (A1) was used to delete any part of the moused on activity which is in conflict in any way. The strategy was to move short-duration activities around manually using E1 (to get them out of conflict with each other) then run A1 on each long-duration activity. This quickly creates a conflict free schedule for some tracks.

Some activities were in conflict solely because of target and maneuver conflicts. A1 could delete those activity portions, but there were cases where it would be better to relax the targetting constraint and preserve the activity itself in untargetted mode during an otherwise conflicting maneuver. For example it was better for the dust counter to be counting dust in a direction away from the comet than not counting at all. This scenario was accomplished by applying DECIMATE ACTIVITY (A2) to long-duration activities, which cut them up into ten separate descendant activities, followed by UNTARGET (A3) on the conflicting tenths, followed by RECONNECT CONTIGUOUS ACTIVITIES (A4) which would reconnect the 'decimated' activity. All these are directed by the mouse.

Another scenario was to DECIMATE all the long events, delete some of the tenths using E2, move things around manually using E1, then EXPAND EVENTS WITHOUT CONFLICT (A5) which would expand the decimated events after this manual 'shuffling' to the extent possible without instituting conflicts.

In the ***Expert Category*** different customizations of PLAN-IT have included complex 'strategies' which were attempts to code as much expertise as possible into the software. We tailored one of these, SHUFFLE TO REDUCE CONFLICT (S1), and found that it worked well to reduce conflict to an initial minimum. This minimum was a starting point for the rest of the scheduling operations.

The PLAN-IT initial allocator piles activities on top of each other at the beginning of their windows, regardless of conflict. Since many activities have windows a week long, this created an initial allocation with a lot of conflict at the beginning of the week. One heuristic we coded and tried with some success was DISTRIBUTE LOW-DENSITY EVENTS (S2). It collected events which occurred less than once per track and allocated them evenly among the tracks. Its only constraint was not to put activities in tracks which could not support the requested data rate.

The combination which seemed to give the best result was to execute S2 then S1 to return a "good cut" schedule.

Specialized Constraint Checking could not be practically represented in the blackboards, because their integration into the basic conflict definitions resulted in excessive execution time. The decision was made to have separate reporting code which was invoked by an operator call from the menu. The disadvantage was that operator decisions could cause increased conflict in these areas without the operator being the least aware of it till an explicit check was made later.

The TIME-CHECKER (C1) routine compared required separation time to actual separation time constraints for a supplied activity name, and displayed activities and time periods which violated requirements for minimum or maximum separation times.

POPUP-STATS (C2) showed a percentage of the time spent supporting an activity along with targetting percentages.

ANALYSIS OF THE CRAF DEMONSTRATION RESULTS

The CRAF demonstration was successful. Conflicts could be eliminated and constraints were reported within a reasonable time span, and in a user-natural way. The core of the interactive scheduling capability shows sufficient promise to be worth incorporating into an operational tool, and basis exists for further development towards an Expert System for the spacecraft scheduling problem.

It was our feeling that use of the mouse and menu techniques, as opposed to keyboard input, was necessary for the successful creation of a user-natural impression. Every time the scheduler takes his eyes off the screen to input keystrokes, he loses concentration on the problem. PLAN-IT made it possible to design an interface profitably emphasizing the mouse and menu.

It is unclear whether the final CRAF science instrumentation set will present problems of a qualitatively different nature from the test set. The general observation is that each instrument tends to present a qualitatively unique scheduling challenge.

The number and complexity of constraints imposed on the range of acceptable results would likely be far greater in a real situation. This is partly a function of the number of instruments and wider variety of requests, and partly of operational experience. A wide variety of conflict and constraint conditions need to be studied to characterize the tractable conditions.

FUTURE WORK

Although the PLAN-IT CRAF initial development met its objectives, additional work is implied for the future if the ultimate objective of operational support is to be met.

There is a requirement to represent steps within activities, with temporal and precedence interrelationships among the steps. It is generally agreed the test set representation was biased in favor of single-step, single-resource activities. However this is not considered a research issue, since other PLAN-IT implementations (12) support multi-step activities.

Dr. Boris Katz (Massachusetts Institute of Technology) demonstrated a natural language parser for possible use as a front end for the PLAN-IT scheduling process (13). Data originating from natural language was translated to Lisp forms which were fully compatible with the CRAF test bed PLAN-IT software. Data was input, displayed and updated at high speed using the Katz parsing system. We are optimistic such capability will have a place in spacecraft scheduling pending further definition of the role for natural language in that endeavor.

The power and energy management quandary inherent in being dependent on energy collected by solar panels is likely to require a heuristic solution peculiar to the CRAF spacecraft. The resource can be modelled as a continuous function dependent on distance from the sun.

Future work includes expansion in the Request Integration and the Activity Detail Design areas including tape recorder management. Heuristics and functions different from the test case will be required.

The current support for CRAF can be enhanced to include scheduling of activities outside tracking times.

CONCLUSIONS

The PLAN-IT CRAF effort was successful because of the efficacy of the tools afforded by PLAN-IT as a development base. PLAN-IT CRAF showed promise in its ability to assist interactively with integrating requests in a complex mission, using a subset of the mission resources over a subset of the conflict and constraint conditions.

With Project approval, capabilities to be added to the CRAF scheduling paradigm include power, energy and tape recorder management. Capabilities which require additional coding include activities outside of tracking times, and providing multi-step activity definitions.

Suggested improvements for the generic PLAN-IT core rose out of the CRAF effort: (1) provide Basic category commands, and (2) provide a language in which a higher-level 'applications' coder can design Structured and Expert Levels operations for his application.

ACKNOWLEDGEMENTS

The PLAN-IT CRAF demonstration described in this paper was performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration (NASA), and was funded through the Director's Discretionary Fund.

The PLAN-IT development was jointly funded by the NASA Office of Aeronautics and Space Technology (Code R) and the NASA Office of Space Science and Applications (Code E).

The authors wish to acknowledge the special assistance of CRAF Project representatives Sylvia Miller and Roy Kakuda for their help in comprehending the spacecraft and mission features. Thanks also to Donna Wolff, Supervisor of the Mission Planning Group, for providing and interpreting the test data.

The PLAN-IT CRAF demonstration would not have been possible without the technical assistance of Eric Biefeld, W. Curt Eggemeyer and Michael Hollander in interpreting the PLAN-IT core program.

The authors have been supported throughout the work by the efforts of the other members of the JPL Sequence Automation Research Group: Sven Grenander, Win Lombard, David Mittman, Stephen Peters, Mark Rokey, Louis Rubenstein, and John Sisino.

REFERENCES

(1) McLaughlin, W. I., and Wolff, D. M. "Voyager Flight Engineering: Preparing for Uranus". (AIAA-85-0287) AIAA 23rd Aerospace Sciences Meeting, January 1985.

(2) Linick, T. D. "Spacecraft Commanding for Unmanned Planetary Missions: The Uplink Process". Journal of the British Interplanetary Society, Vol 38, pp. 450-457, 1985.

(3) Planetary Exploration through the Year 2000 - A Core Program. Solar System Exploration Committee of the NASA Advisory Council. 1983.

(4) Grenander, S. U. "Toward the Fully Capable AI Space Mission Planner". Aerospace America, Aug 1985.

(5) Page, D. Space Flight Operations Center Sequencing Subsystem Level 4 Functional Requirements. (SSEQ0004-00-02). October 15, 1986.

(6) Gilb, T. "Evolutionary Delivery versus the 'Waterfall Model'". ACM SIGSOFT Software Engineering Notes, Vol 10, No 3, p. 49.

(7) Peterson, B. "The Spiral Model of Software Development". Data-Link, December 1985.

(8) Biefeld, E. "PLAN-IT: Knowledge Based Mission Sequencing". Proceedings of SPIE on Space Station Automation, Oct 1986, pp. 126-130.

(9) Barr, A. and Feigenbaum, E. A. The Handbook of Artificial Intelligence. Vol I. p. 156.

(10) Nii, H. P. "Blackboard Systems: the Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures." The AI Magazine Summer 1986.

(11) Miller, S. L., Bender, D. F., Stetson, D. S., and Myers, M. R. "Trajectory Options for the Comet Rendezvous Asteroid Flyby Mission". (AIAA-87-0641). AIAA 25th Aerospace Sciences Meeting, January 1987.

(12) Eggemeyer, W. C., and Bowling, A. "Deep Space Network Resource Scheduling Approach and Application". This publication.

(13) Katz, B. and Winston, P. H. "A Two-Way Natural Language Interface". Proceedings of the European Conference on Integrated Interactive Computing Systems. (ECICS 82). September 1982.

An Expert System That Performs A Satellite Stationkeeping Maneuver

M. Kate Lines-Browning and John L. Stone, Jr.
May 13, 1987

An Expert System That Performs A Satellite Stationkeeping Maneuver

M. Kate Lines-Browning and John L. Stone, Jr.

Abstract:

In this paper, we describe the development and capabilities of a prototype expert system that provides real-time spacecraft system analysis and command generation. At present, ESSOC (Expert System for Satellite Orbit Control) is capable of performing the stationkeeping maneuver for a geostationary satellite.

ESSOC guides the operator through the stationkeeping operation by recommending appropriate commands that reflect both the changing spacecraft condition and previous procedural action. Information regarding satellite status is stored in a knowledge base internal to the expert system. This knowledge base is continuously updated with processed spacecraft telemetry. Information on the procedural structure is encoded in production rules. The independence of the procedural rules from each other, and from the knowledge base, makes the system easy to maintain and expand.

Particular attention is directed to distinctive features of the ESSOC system and its development, namely, the structured methods of knowledge acquisition, and the design and performance-enhancing techniques that enable ESSOC to operate in a real-time environment.

1.0 Introduction

Certain properties of current satellite operation techniques indicate that significant benefit may be derived by introducing automation into the field of satellite operations. First, satellites are difficult to operate, requiring skilled teams that are both difficult to assemble and expensive to maintain. Second, errors on the part of the flight crew can be expensive to rectify or can even be irreversible. Automating satellite operations offers a number of distinct advantages:

1) Swift anomaly detection and response;
2) Identification of transient conditions;
3) Correct operational response to the aforementioned conditions; and
4) Capability to implement increasingly complex flight rules.

As proof of the concept that the use of expert systems is an efficient method of achieving the goals listed above we have developed ESSOC, a prototype expert system for satellite operations.

Rather than construct a system to completely handle all satellite operations, we limited the scope of ESSOC operation to a subset of the operations for a mission. Furthermore, once a prototype system was produced, the modular design inherent in ESSOC would enable us to expand the system over time.

We selected the stationkeeping maneuver for the TDRS-1 spacecraft as the domain for our development effort. The choice of spacecraft was predicated upon the availability of knowledge engineers familiar with the domain. The choice of the particular satellite operation to be automated was more arbitrary, but the stationkeeping maneuver met the following desirable criteria:

1) Need for swift response to problems;
2) Reasonable procedure duration (approximately 3 hours);
3) Manageable domain size/development complexity;

4) Universality of application to different satellites;
5) Critical need for correct commanding;
6) Greatest potential benefit to current TDRS operations.

Conveniently, the TSIM real-time TDRS Simulator was available to serve in the test bed for ESSOC, providing both a telemetry stream and a command response.

Using the techniques outlined herein, ESSOC prepares the satellite (in this case the TSIM real-time simulator) for the orbit adjustment by recommending and sending commands that route propellant to the appropriate thrusters for attitude control and firing the delta V thrusters. Attitude control modes are switched as necessary and the success or failure of each step of the procedure is verified continuously via telemetry. The user is notified of problems as problems are detected. ESSOC generates satellite commands in response to anomalies and displays them for user action.

2.0 The TDRS Spacecraft

The TDRS spacecraft (shown in Figure 1) is a 3-axis-controlled, bias-momentum-stabilized communications satellite. Launched in 1983 and stationed at 71° West longitude, TDRS-1 will be joined by at least two similar spacecraft yet to be launched. During normal on orbit operations, reaction wheels are used to control the attitude, while one pound (nominal) thrusters are used occasionally to remove accumulated wheel momentum. These same

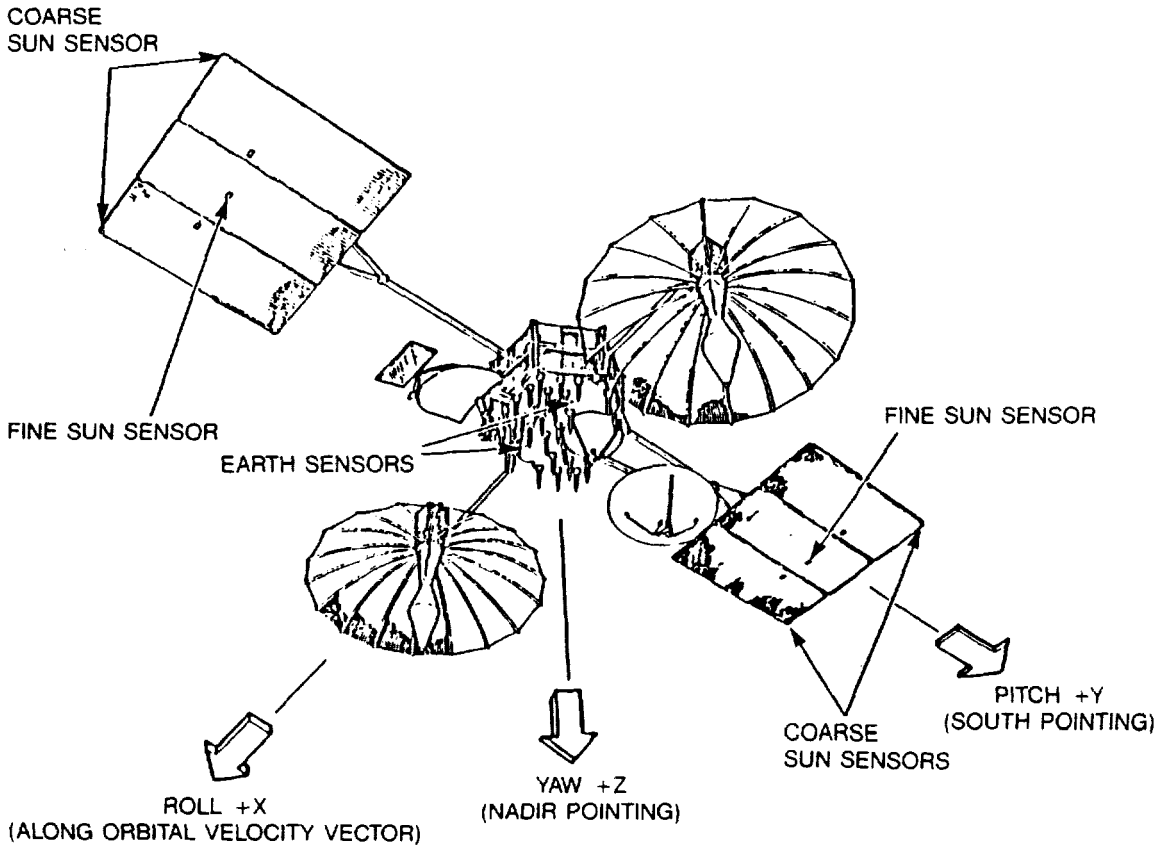


FIGURE 1. ON-ORBIT CONFIGURATION

thrusters are also used periodically to adjust the TDRS orbit to correct for perturbations, and to maintain the TDRS attitude and thrust vector during this correction procedure.

The thrusters use catalytic decomposition of hydrazine on a heated catalyst bed, which must have its temperature maintained through proper commanding. The earth sensors provide pointing error information provided the satellite is within five degrees of the proper attitude. The solar arrays provide electric power, and must be maintained in the correct, sun-pointing position by rotating them about the pitch axis as the satellite orbits. Various antennas included in the spacecraft payload may also be seen (Figure 1). Further details on the TDRS may be found in "TDRS Spacecraft Operations."[1]

3.0 The ESSOC Expert System

The ESSOC expert system resides on a Symbolics 3675 LISP machine. It receives processed telemetry data via Ethernet from the ESSOC front end processor which resides on a VAX 11/785. The telemetry data is provided by a real-time spacecraft simulator that also resides on the VAX. In this section, we will discuss each portion of this configuration separately. A discussion of the link between the two machines is found in section 3.3.5.

3.1 ESSOC Development Environment

The Symbolics 3675 LISP machine is a stand-alone, single-user LISP workstation. Our system is equipped with both a black-and-white console and a color graphics monitor. The programming environment supports multi-windowing, multi-tasking, incremental development of programs, and optimized LISP programming. We used LISP to implement the expert system functions that dealt with the color graphics, procedure timing, networking, and arithmetically intensive functions used by procedural rules. Most of the expert system, however, was developed using the expert system development shell, ART (Automated Reasoning Tool), which greatly expedited expert system development. ART provides an inference engine and mechanisms for representing frames (schemata), rules (backward and forward chaining), and inheritance relations.

Telemetry data for the expert system is provided by a real-time, high-fidelity simulator of the TDRS spacecraft (TSIM) that resides on the VAX 11/785. Because the simulator is able to model response to commanding in telemetry, the simulator provides a telemetry stream (1000 bits per second) functionally identical to that of the spacecraft. Hence, in designing the expert system, we were able to consider the simulator indistinguishable from the satellite.

ESSOC's front end processor, which resides on the VAX, is responsible for processing the raw telemetry data from the simulator into a specified format and placing it into a processed telemetry buffer on the VAX. The conversion of data is performed in two steps. First, the front end processor breaks the data from the simulator down into complete telemetry frames and stores these frames in a buffer on the VAX. Whenever the processed telemetry buffer becomes empty, the second step of the processing is performed. The second step of the conversion process changes this raw data into engineering units, performs trend determination, and labels the data values with ASCII tags. The front end processor places the resulting data into the processed telemetry buffer which holds up to 64 frames (32768 bits) of telemetry.[2]

3.2 Expert System Development

Using the "rapid prototyping" method of software development, a working prototype of the ESSOC expert system was developed within six months. The first three months of the project were spent in an intensive knowledge acquisition phase. The information collected at this time was used to select a scheme for representing knowledge in the expert system that would allow the system to operate in real time and to be expanded. After deciding on the general design of the system, the information gained from the domain experts (in this case, spacecraft engineers) was organized and converted into code. The expert system prototype generated from the initial data was evaluated by the domain expert and suggestions for improvements made by the spacecraft engineers were incorporated into the system. The development cycle then repeats: the spacecraft engineers are interviewed by the knowledge engineers to obtain more information about the problem domain, this knowledge is organized and encoded, and the resultant system is evaluated by the experts. With each iteration of the development cycle, the system becomes more refined and complete.

From our initial interviews with the spacecraft engineers, it was clear that there were two basic types of knowledge about the problem domain that were needed by the system: procedural knowledge and structural knowledge about the spacecraft. Hence, we drew the methods to organize our data from two distinct software design methodologies: Object Oriented Design (OOD) and Structured Analysis Design Technique (SADT).[3] In implementing the expert system, we encoded the structural knowledge in frames, and we encoded the procedural knowledge in forward-chaining rules. The ESSOC expert system therefore, may be described as a hybrid-frame/rule-based system.

3.3 Expert System Structure/Operation

The ESSOC expert system is composed of three main portions: 1) the rule base; 2) the knowledge base (ESSOC's internal representation of the satellite); and 3) the user interface. In addition, ESSOC requires processing of the satellite data by custom software. In the following sections, we discuss each of the parts of the expert system in more detail and the flow of data throughout the expert system and its test bed.

3.3.1 Rule Base

The rule base of the expert system contains all the procedural knowledge of the system (i.e., procedures for detecting and correcting anomalies as well as the procedure for the delta V itself). The procedural knowledge gained from interviewing the engineers was divided into specific phases which were then subdivided into discrete activities. As prescribed by the SADT methodology, each of these activities was then analyzed by identifying the inputs, outputs, and constraints associated with each of the activities.[4] In composing rules, the left (IF) side of the production rule contained the input conditions and the constraint conditions, whereas the right (THEN) side of the rule contained the items in the output portion of the activity description. Examples of an activity decomposition and the resultant ART rule are shown in Figure 2.

Because we anticipated that there would be a large number of rules in the system, and because ART's inference engine considers every rule for matching in every inference cycle, a strategy to speed the matching of rule patterns against the data base was employed. The rules were partitioned into functionally related sets called rulesets. Rulesets may be designated as active or inactive based on the relevance of the function of this ruleset to

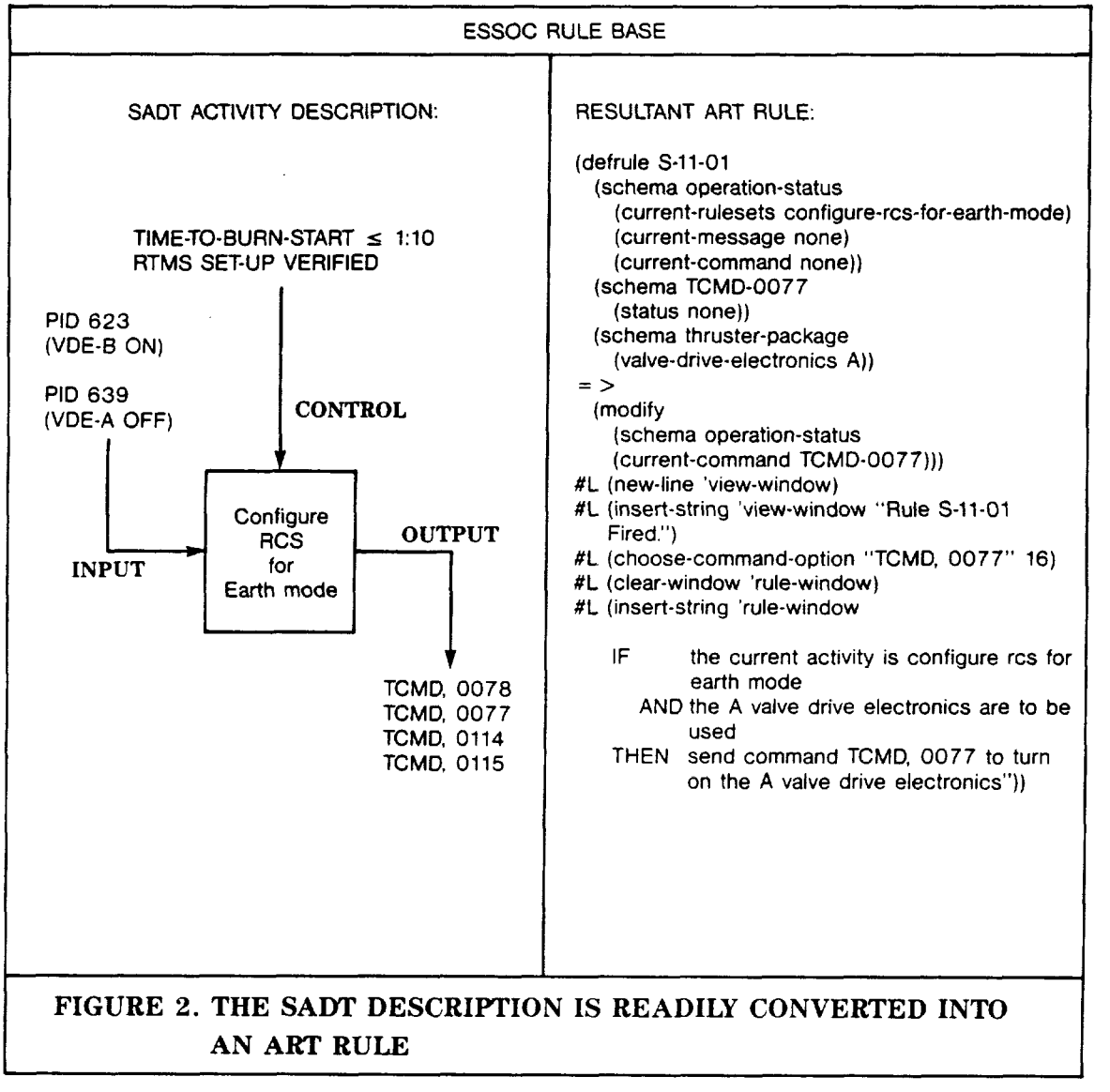


FIGURE 2. THE SADT DESCRIPTION IS READILY CONVERTED INTO AN ART RULE

the current status of the maneuver. The status of a ruleset (active or inactive) is dynamically determined by a set of metarules that respond to specific telemetry, timing, and sequencing conditions. The first condition for matching a rule is that the rule be a part of a ruleset which is active. Since only a few of the rulesets are active at any one time in the maneuver, the time that the system spends pattern matching is greatly reduced. The list of rulesets which are active is stored in a data structure in the knowledge base.

In addition to the metarules, there are two other general categories of rulesets: phase-specific and phase-independent rules. Phase-specific rules perform the delta V procedure. For example, there is one ruleset that enables the catalyst bed heaters, and another that opens the propellant valves, etc. There are a total of 22 phase-specific rulesets. Phase-independent rulesets are those that perform monitoring functions. There are a total of six phase-independent rulesets. For a more detailed listing of these rulesets, see the paper "An Expert System for Satellite Orbit Control."[3]

As an example of the operation of a monitoring (phase-independent) ruleset, we discuss the Rhold monitor found in ESSOC. This monitor is active for a considerable period

during the delta V procedure, during which several of the phase-dependent steps execute. At intervals that are unknown in advance, the Rhold monitor interrupts the normal procedure to recommend commands. This operation is detailed as follows.

Shortly following launch, a failure rendered 13 of the 24 TDRS hydrazine thrusters unusable,[5] further complicating control of the spacecraft. The failed thrusters are shown (in black) on the diagram of the spacecraft in Figure 3. In particular, the lack of an operating negative roll thruster required an alternative method to provide negative roll torque for attitude control. The workaround developed requires firing a pair of yaw thruster pulses that cancel in yaw but have a fractional negative roll torque. The command sequence for performing the pair firing is called "Rholdn," where n is a number from one to seven denoting the number of thruster pulse pairs. These command sequences must be performed to provide negative roll control authority whenever thrusters are used for attitude control, such as during the delta V procedure.

Currently, the attitude control system specialist instructs the satellite controller to command the spacecraft by observing the earth sensor roll error, and issuing corrective satellite commands based upon his/her intuition and experience. Incorporated in ESSOC is a roll axis controller, the Rhold monitor. The block diagram for this monitor is shown in Figure 4. While unremarkable in design (further details on automatic controllers of this type may be found in the book *Automatic Control Systems*[6]), the ability to use a real-time controller in an expert system illustrates the performance margin and flexibility found in ESSOC.

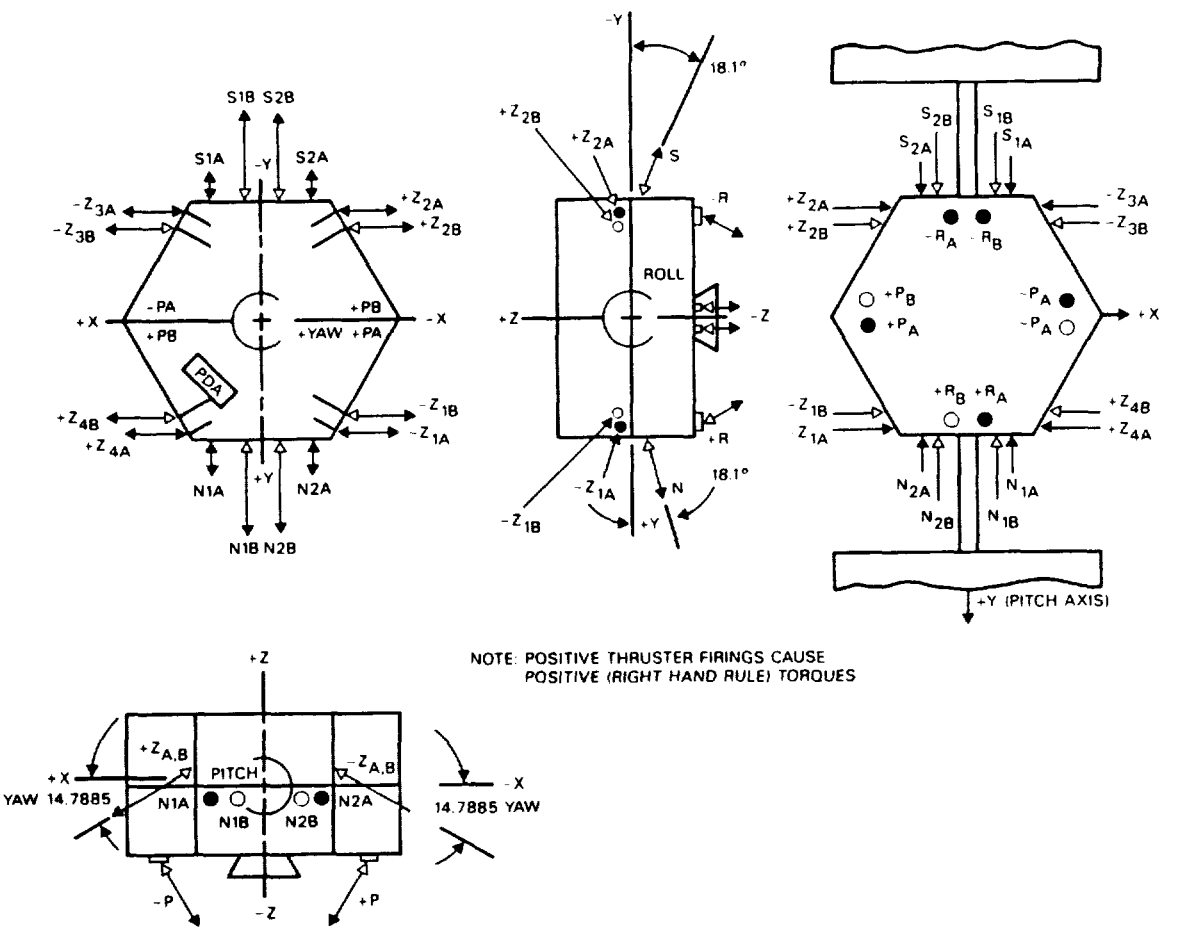


FIGURE 3. DUAL THRUSTER MODULE LOCATIONS

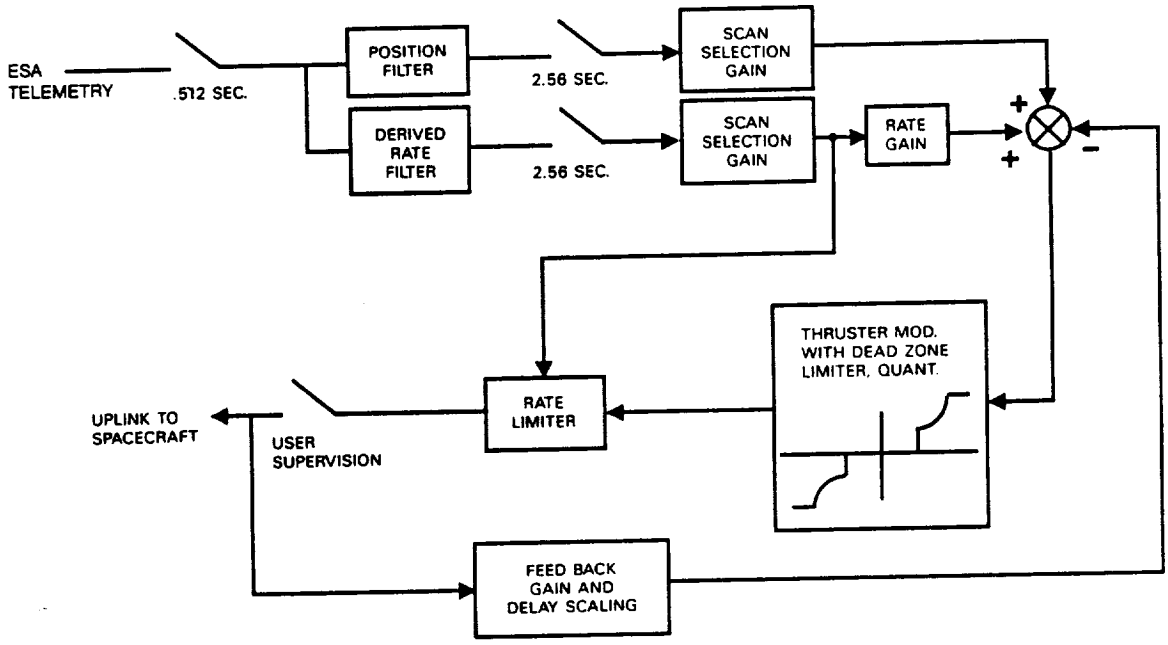


FIGURE 4. RHOLD MONITOR CONTROL LAW BLOCK DIAGRAM

In ESSOC, the controller itself is implemented in LISP on the Symbolics with the exception of the data filters, which are described in Section 3.3.5. Previous implementations of this type of controller by CONTEL, were done in FORTRAN as a closed (no operator control over commanding) loop.[7]

When the controller determines that commanding is required, a condition is set in the expert system knowledge base that causes a rule firing which displays the proper command for user action. As detailed in the user interface section, a monitor-generated rule preempts any procedural commands in requesting user action.

3.3.2 The ESSOC Knowledge Base

As prescribed by the OOD method, features of the spacecraft were identified as objects; the operations which act upon these objects were identified and the object's attributes and values were listed.[8] With the structural knowledge organized in this way, the conversion of the knowledge into ART's schemas (ART's frame-based knowledge representation) was a very natural process. Figure 5 shows an example of a schema which represents a spacecraft object in the knowledge base. ART's relational network allows the system to represent not only parts of the spacecraft, but also the relationships between spacecraft parts. Values from the telemetry data are stored in the knowledge base schemas and updated from incoming telemetry data. Thus, at any given moment the state of the knowledge base reflects the current status of the satellite; this portion of the knowledge base may be regarded as a satellite simulator internal to the expert system.

In addition to information about the spacecraft, certain control structures are also present in the knowledge base. One important control structure that we have already discussed is the schema that keeps track of the rulesets which are active. Another important control device is ESSOC's clock. The clock schema stores the current Greenwich Mean Time and additional needed timing information such as the time until thruster burn start, duration

An ESSOC Control Structure: the ESSOC Clock

```
(defschema operation-time      ;time of delta v operation
  (gmt "330:00:00:00")         ;(ddd:hh:mm:ss)
  (duration 100.0)             ;deci-seconds
  (new-time no)                ;old/no/yes/new
  (new-duration no)            ;no/yes
  (universal-time 0))          ;seconds since 1/1/1900
```

A portion of the ESSOC Satellite Representation: the th-Z3B Thruster

```
(defschema th-Z3B)
  (instance-of thruster)
  (thruster-id th-Z3B)
  (status dis)                 ;pid 680 (en/dis)
  (health working)             ;(working/failed)
  (duty-cycle 0)               ;(%)
  (temp 0)
  (redundant-thruster-id th-Z3B)
  (cat-bed-heater-id cat-bed-htr-Z3B)
  (propellant-valve-id prop-valve-Z3B)     ; prop-valve-xxx
  (thermistor-id thermistor-Z3B))
```

FIGURE 5. ESSOC KNOWLEDGE BASE

of thruster firing and whether this information has been changed while configuring for the delta V maneuver. ESSOC's clock is updated from the system clock and may be accessed easily by any of the rules.

3.3.3 ESSOC User Interface

The ESSOC system is equipped with two monitors: a black-and-white monitor and a high-resolution color monitor. The black-and-white console is an interactive user interface that functions as the command terminal. The color monitor is used for generating graphics that augment, rather than replace, displays that are currently available to the satellite controller.

ESSOC displays its recommendations and messages to the user in a specific window on the black-and-white screen. The operator may send or cancel a command, or confirm a message recommended by ESSOC by selecting the appropriate option from the command menu with the mouse. From this user interface, the expert system may query the user for information not found in the telemetry, and may request confirmation that certain procedures have been accomplished before proceding with the delta V. High-priority commands are displayed on pop-up menus that cover the command window, forcing the operator to respond before continuing with the procedure. In separate windows, ESSOC displays a history of recommended commands, a history of the commands that have been sent and a brief justification of the currently recommended command or message. In addition to the command interface present on the black-and-white screen, additional windows give helpful information to the operator concerning the Greenwich Mean Time, length of time prior to thruster firing, and the current phase of the delta V operation. The operator may also send a satellite commands at will from this screen.

The ESSOC color graphics are displayed on a high-resolution (1280 X 1064) 24-bit color graphics screen. While not strictly necessary for ESSOC operation (in contrast to the monochrome display), information on this display is provided to aid the user in his/her decision making.

To date, two real-time displays have been implemented. The first depicts the current configuration of the TDRS Attitude Control Subsystem. The values displayed on the screen are obtained from the spacecraft telemetry found in the ESSOC database. The second depicts the orientation of the spacecraft with respect to the earth. Attitude position limits are indicated by rectangles about the center of the earth (nadir). Earth sensor fields of view are indicated by animated rectangles that are repositioned to reflect spacecraft attitude motion. The coordinate transformations and graphics for the display are coded in LISP; the data are drawn from the expert system's knowledge base.

3.3.4 ESSOC's Inferencing Cycle

The ESSOC expert system's inferencing cycle is a slight modification of the standard match-select-fire inferencing cycle; the ESSOC cycle consists of four distinct steps rather than three. The operational cycle is as follows: first, the expert system examines the information in its database and determines which, if any, of the operational rules are matched; second, the system selects one of the rules that have been matched; third, ESSOC executes one of these rules; fourth, the system reads any data present in the data buffer on the Symbolics into the knowledge base. The pattern-matching and the rule selection, conflict resolution algorithms are provided by the expert system tool ART, whereas the data-polling and parsing functions were custom-made for this application and implemented in LISP. Note that the expert system does not wait for data; if no telemetry data are present in the buffer, the expert system continues inferencing.

Commands and messages are generated and displayed to the user during the third step of the cycle, as a result of rule execution.

3.3.5 ESSOC's Data Flow

Because ESSOC is data driven, it is important to discuss the methods by which data are generated and placed into the system's data structures. As previously discussed, the system and its knowledge base reside on a Symbolics 3675 while the source of data (the simulator TSIM) and the data preprocessor reside on a VAX 11/785 (see Figure 6).

Expert systems are CPU-intensive, often requiring 80 to 90 percent of the CPU's processing power. Because the processing of the telemetry data involves trend determination and conversion to engineering units, which are arithmetically intensive, and because performing of processing on the Symbolics would significantly interfere with the expert system's use of the processor, we decided to perform all the telemetry data processing on the VAX. This design decision greatly enhanced ESSOC's real-time performance. The following paragraphs describe the flow of data in the system, the operational cycle of ESSOC and the way in which the two are integrated.

The flow of information between the expert system and the simulator is bidirectional; spacecraft telemetry data are transmitted from the simulator to the expert system and commands are transmited from the expert system to the simulator.

The transmission of commands to the simulator from ESSOC is totally under the operator's control; commands may be sent at any time the expert system is in operation. Whenever the operator sends a command, the command is transmitted to the VAX. The system is designed so that the simulator will accept the commands coming over the link and will model a response in telemetry.[2]

In contrast, the transmission of telemetry to the Symbolics is controlled by software on the Symbolics. The simulator generates data continuously. To prevent data loss, data is buffered on both the VAX and the Symbolics side of the link. The ESSOC front end proces-

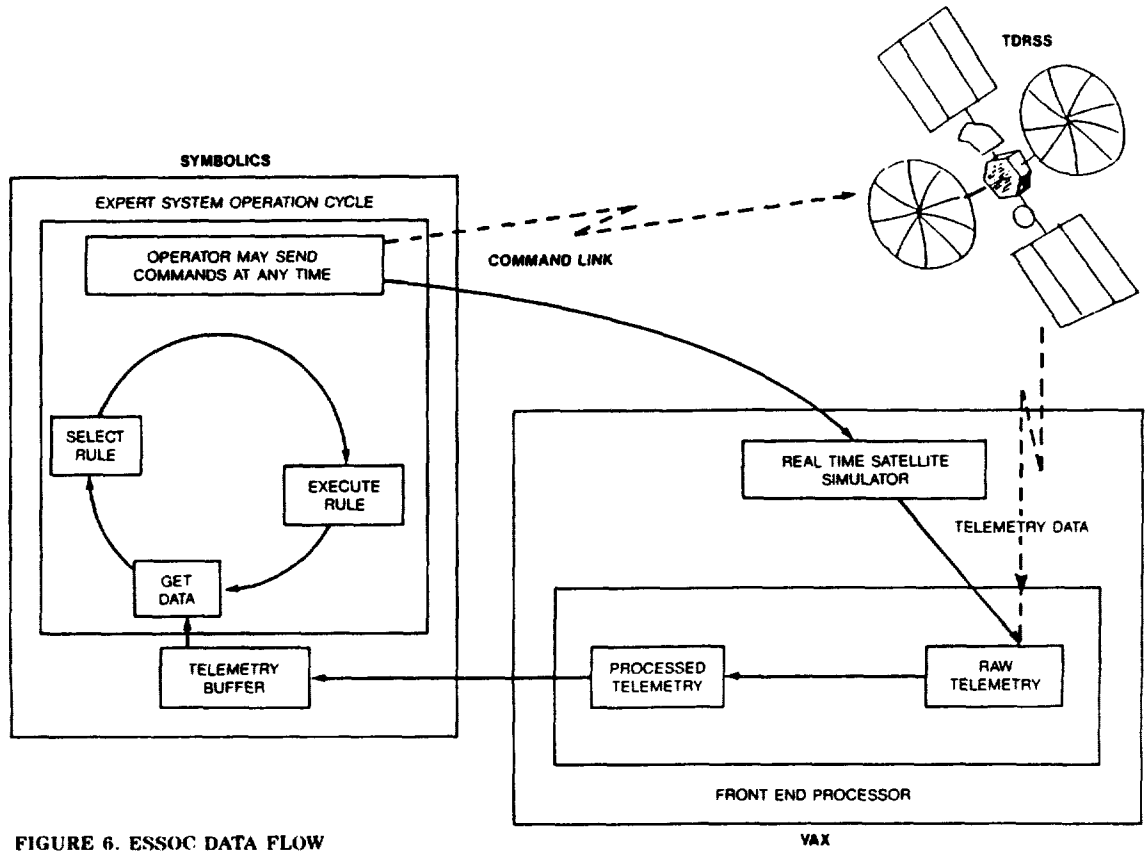


FIGURE 6. ESSOC DATA FLOW

sor converts the raw data into blocks of processed telemetry data.

The ESSOC front end preprocessor receives the binary bit stream and performs real-time conversion of the telemetry data to engineering units. Following the conversion, certain parameters are further processed through sliding average filters and derived rate filters. The data are then associated into object-value-attribute pairs and sent to the output buffer as ASCII strings. The derived parameters are sampled each 0.512 seconds, averaged or fit over 5.12 seconds, and are placed in the processed telemetry buffer at 2.56 second intervals. The front end processor is implemented in FORTRAN on a VAX 11/785, co-resident with the spacecraft simulator.

When the data buffer on the Symbolics side of the link is empty, one frame of processed telemetry is sent across the network in a block and stored in the buffer. At the proper time in its inferencing cycle, the expert system places this buffered information into its knowledge base. At this point, the data buffer on the Symbolics side is empty and another data block is requested from the processed telemetry buffer. Data transmission across the Ethernet occurs asynchronously with respect to the ESSOC expert system operational cycle. It is important to note that the system continues to inference and monitor telemetry data even when the system has requested user input. Because the system continues to check data and inference, anomalous conditions will not go undetected during the time that ESSOC is awaiting a user response.

4.0 Testing

In testing ESSOC, we first tested each individual rule by presetting its conditions in the knowledge base and then determining if the rule fired and if the proper actions were taken.

After the individual rules of the system were verified, the complete rule base was tested by transmitting known data generated by an off-line TDRS-1 simulator to the expert system through the expert system's TSIM-VAX link. Scenarios were constructed that generated data to test the function of each ruleset. Because we had complete control over the data in the scenario, we could determine whether the rules were firing at the correct time and under the proper circumstances. Using an off-line simulator rather than telemetry tapes allowed us to control, as well as generate, anomalous data with which to test the monitoring rule sets.

Future testing of ESSOC will be accomplished by linking the system to the simulator TSIM, although at the time of this writing, this link has not been tested.

5.0 Conclusions and Discussion

ESSOC has demonstrated that expert systems technology has promise for supplementing current communications spacecraft control and monitoring methods. By using an expert system to perform the TDRS-1 delta V procedure, the probability of incorrect commanding can be greatly reduced. Changing spacecraft conditions are detected as they occur, and the proper response made immediately. Virtually any failure mechanism that can be identified in advance may be entered in the knowledge base during the development and operational phases.

While some of the above capabilities may be achieved through other methods (e.g., rule-based expert systems, conventional programming techniques), the hybrid rule/frame-based expert system is faster and far easier to maintain. Since the rulesets are functionally independent, additional rules may be added easily to either expand capability or correct faults.

No discussion of ESSOC's capabilities would be complete without mentioning some of its limitations. The types of support that an expert system (or any other system) may provide for a given fault during a process are: 1) anomaly detection; 2) attaining safe configuration; 3) performing corrective action/identifying workarounds; and 4) cancelling the process. At present, ESSOC provides only anomaly detection, the ability to cancel the procedure, and a limited capacity for workarounds. Performing corrective action/identifying workarounds would theoretically appear to be the most desirable capability to develop in an expert system, but in a majority of cases it is more desirable in practice to attain a safe configuration. A number of reasons supporting this conclusion are listed below.

1) It is simpler to identify a type of problem than it is to correct a specific one. Each problem type has a procedure that corrects a family of problems and places the spacecraft into a safe configuration. Because there is one procedure per problem type rather than one procedure per problem, the number of corrective procedures that the experts must define is reduced.

2) It is more cost effective. While the chance of a particular failure scenario occurring is quite small, considerable expense is required to develop, implement, and test each of a large number of explicit failure scenarios. Conversely, simply configuring the satellite for a safe haven, prior to corrective action by specialist input, prevents this excessive level of expenditure.

3) It is safer. Any of a number of events, which may require distinct recovery procedures, may produce similar symptoms in the telemetry. Thus, the actual failure mode may not have been foreseen at the time of expert system development, leading to incorrect

response and unnecessary risk.

While the above discussion suggests that safe haven anomaly recovery is in general preferable to explicit recovery, there are exceptions. For certain failure modes, the failure causality and corrective action is relatively simple, and consequently, the costs of not performing these actions is great. Likewise, certain failure modes can be identified as having a greater probability of occurrence than others which warrants an increased level of expert system response capability. For these cases, it is preferable to implement a full recovery procedure.

While ESSOC is used for performing the delta V procedure, the techniques used by ESSOC can be generally applied to spacecraft control. A system like ESSOC could be expanded to handle many tasks in satellite operations. The ultimate goal would be to enlarge the expert system so that it could perform all phases of satellite control.

With slight modification to the system, the user can be eliminated from the control loop entirely; the expert system would request user interaction only as unforeseen circumstances occur. Under these circumstances, manpower needs would be greatly reduced.

Acknowledgements:

The funding for development of ESSOC was provided by Contel Federal System's internal research and development fund. The authors thank the TDRS Program Office for their support during the development of ESSOC and acknowledge the dedicated efforts of the other members of the ESSOC development team: Terry Rubin, Ted Russell, and Jim Young.

1. "TDRS Spacecraft Operations," volumes 1-5, TRW Defense Systems Group Document TMD 253, 1985.
2. M.K. Lines-Browning and J.L. Stone, Jr. "An Expert System for Conducting a Satellite Stationkeeping Maneuver." Submitted for publication to the 1987 MILCOM Conference.
3. F. Rook and J. Odibiyi. "An Expert System for Satellite Orbit Control." Expert Systems in Government Conference, Tyson's Corner, VA, 1986.
4. D. Ross and K. Schoman. "Structured Analysis for Requirements Definition." *IEEE Transactions on Software Engineering*, January, 1979.
5. "TDRS F-1 Launch to Synchronous Orbit Spacecraft Subsystem Performance" 31 March 1984. Space Communications Company Document W200A122.
6. B.C. Kuo, *Automatic Control Systems*, Third Edition, Prentice-Hall, 1975.
7. "Gyro-less Acquisition Operations Manual," Space Communications Company Document TMD Z53, 1984.
8. G. Booch. *Software Engineering with Ada*. Menlo Park, California: Benjamin/Cummings Publishing Company, 1983.

DEEP SPACE NETWORK RESOURCE SCHEDULING APPROACH AND APPLICATION

William C. Eggemeyer, Alan Bowling
Mission Profile and Sequencing Section
Jet Propulsion Laboratory
California Institute of Technology

ABSTRACT

Deep Space Network (DSN) resource scheduling is the process of distributing ground-based facilities to track multiple spacecraft. The Jet Propulsion Laboratory has carried out extensive research to find ways of automating this process in an effort to reduce time and manpower costs. This paper presents a resource-scheduling system entitled Plan-It with a description of its design philosophy. Plan-It's current on-line usage and limitations in scheduling the resources of the DSN are discussed, along with potential enhancements for DSN application.

INTRODUCTION

Scheduling is believed to be one of the most difficult issues artificial intelligence (AI) has attempted to resolve. This paper addresses the how and why of AI structures and techniques which were used in resolving the DSN Resource Allocation scheduling problem. Finally, the results, which caused a factor of six speed-up in the schedule generation process, will be discussed.

This paper encompasses three main topics. The first part of the paper describes the constraints and requirements of the DSN Resource Allocation scheduling problem, followed by a description of the design philosophy behind the AI scheduling system Plan-It, providing the conceptual background for this approach. The remaining portion of the paper will discuss Plan-It integration and application to DSN Resource Allocation scheduling, along with what has been learned from the task.

DSN RESOURCE ALLOCATION PROBLEM DESCRIPTION

The Deep Space Network is a worldwide system of tracking antennas, consisting of three ground stations spaced 120 degrees in longitude from each other. The stations are located in Canberra, Australia; Madrid, Spain; and Goldstone, California. As the earth rotates, this geographical arrangement of stations ensures that a spacecraft will be visible to at least one ground station at any time. Each station has a minimum of three antennas, two 34-meter dishes (one with receiver only, the other with a transmitter) and a 64-meter dish antenna.

Scheduling DSN support for tracking spacecraft is a very difficult problem, involving many dynamic factors that influence or even change a scheduler's strategy from month to month. The schedule is based on a set of constraints consisting of viewperiods, project requests, and DSN system requirements.

Viewperiods are time intervals in which radio dishes have line of sight to their targets. This line of sight is required to monitor the signal from a particular spacecraft or to uplink commands. When the DSN antenna is used for radar imagery of a planet, the planet must be viewable by the antenna. These time intervals may also be referred to as time windows.

Project demands upon the system fall into two major categories. The first consists of viewperiod-dependent requirements that a project levies upon the DSN. Flight projects usually submit a document containing these time-specific tracking requests for the spacecraft and the minimum antenna tracking requirements for the project. For instance, a project may require ten continuous hours of coverage in duration once a day on a 64-meter antenna. This request not only implies multiple usage of an antenna resource, but also implies viewperiod restrictions on where the activities may be placed in the schedule. Some non-spacecraft requests are also viewperiod dependent, if the project wishes to track a planet or a quasar.

The second category of project requests are known as non-viewperiod-dependent requests, and deal with non-time dependent observations, such as certain classes of radio astronomy. These may be in the same format as that of the first category, but contain no target-timing restriction. Both categories of requests have two types of requests: generic and specific. The generic request indicates multiple activities occurring in the schedule, with some time-dependence relationship between them. The specific request specfies a specific date, time, antenna, and duration which a project requires for antenna coverage.

The DSN also imposes many constraints on the system in the form of station maintenance requirements. Each antenna requires a certain amount of maintenance, usually eight hours a week. This maintenance activity is further constrained by not allowing personnel to cross workshift boundaries at the station. There are also times when the station is unmanned, so no requested activity may be scheduled during such time. Other DSN activities

may be antenna upgrades, antenna calibration, and special activities.

In addition to the activities and constraints listed above, the scheduler must observe certain scheduling techniques which may further constrain the schedule. For example, no two antennas may simultaneously track the same spacecraft for more than 30 minutes, unless simultaneous tracking was specifically requested by the project. This limitation/restriction is used to maximize use of scarce antenna time.

Another potential problem the scheduler must address is viewperiod overlap among two spacecraft, causing a conflict in their tracking requests. This conflict forces the scheduler to work out some kind of compromise, such as juggling the projects' requests between other radio antennas or stations, or arranging some time-sharing schedule on an antenna between the two projects. These are just two of the many different strategies available to the scheduler in resolving this conflict.

The amount of constraints and the number of spacecraft requiring tracking yield an incredible number of solutions to a schedule for a particular situation. The scheduler's job is to find the solution which best optimizes antenna usage, meeting at least the minimum tracking requirements of each project.

The preceding text has described but a few of the basic factors a scheduler must consider in establishing a basic DSN schedule. However, there are many other special requests and situations which may change this situation. For example, when a project has a planetary encounter, all of that project's requests become specific requests, which now provide for continuous spacecraft tracking for most of the encounter period. Two or more antennas may be used in tracking the spacecraft simultaneously for hours at a time. These types of constantly fluctuating constraints make the DSN scheduling problem a unique one for which the search for a better solution still continues.

A significant contributing factor to the problem with the DSN Resource allocation plan is that as spacecraft get farther and farther away from earth, a larger diameter antenna is required to pick up the signal from the spacecraft. And since there are few antennas capable of picking up deep space signals, there is a great deal of competition between the projects for the large antenna resources.

As the number of projects requiring support increased over the years, and more special events occurred closer together, each requiring more and more support, it became increasingly more difficult to produce a realistic schedule in a reasonable amount of time. To overcome this difficulty the DSN Resource Allocation Group was formed to develop an automated process to reduce preparation time and enhance reliability of the schedule. The proposed process is split into two parts. The first part consists of the Computer-Aided Resource Allocation and Planning system (CARPA), which provides an initial version of the schedule after all the constraints and requests have been entered into the system. This was a batch-mode scheduling system that uses a dynamic priority bin-packing technique. The second part consists of

manually refining the plan to fit a particular situation.

Each part of the process, however, has its own special problems. A major problem in the refining process is that the conditions for which a schedule was produced may drastically change as the timeframe to implement the schedule approaches. This sometimes requires massive changes to a schedule, which must be made quickly and accurately. Excessive delays can cause further problems in the DSN schedule because the delays may impact a project's future inputs in planning communications with its respective spacecraft. Hence a need exists to further automate the process.

DSN RESOURCE ALLOCATION PLAN-IT OPERATION

Plan-It was developed to address the final refinement or tweaking portion of the scheduling process. The DSN scheduling problem was addressed by the resource-scheduling system Plan-It operating in several conceptual modes, from the most primitive to almost fully automatic. Another requirement Plan-It had to meet was the ability to interface with CARPA.

The figures on the following page show some of the capabilities a user can invoke in Plan-It on a typical DSN schedule. Figure 1 shows a menu of statistics, giving the user a quantitative measure of how the radio dishes are being used. Figure 2 shows the user mousing on a black conflict area to gather further information about that particular conflict. Menu interaction is the main user interface to the program. Every operation is mouse driven. The menus are accessed successively through a tree structure applicable to a particular task category, such as editing, data i/o, strategy implementation and modification, and graphical display control. The functions selected from the menus direct the tool to do different tasks. The major selectable functions are graphical manipulation, data i/o, schedule manipulation and verification.

The graphical display and user's ability to manipulate it maximize the bandwidth of information that passes between the person and the program. Each user has his own way of wanting to see how activities lay out in the schedule. To satisfy this need, Plan-It enables the user to dynamically reorder requests and resource lines on the screen. This further enhances the user-natural interface so a person can intuitively resolve conflict and opportunity patterns seen on the screen. Further capabilities the user possesses with Plan-It are redefining the relative sizes of the activity-plotting pane and the resource line pane. In order not to overwhelm the user with an abundance of scheduling data, the Plan-It screen consists of two major panes acting as small view windows on a much larger scratchpad of the schedule timeline. If the user wishes to concentrate only on a few resources but see more of the activity layout, he changes the relative proportions between the two windows. The final graphical manipulation tool the user has at his disposal is the ability to change the frequency that Plan-It updates its windows. By default, Plan-It will update its windows whenever any action occurs. If the user does not wish to see all of the intermediate action taking place during a task execution, he can change the

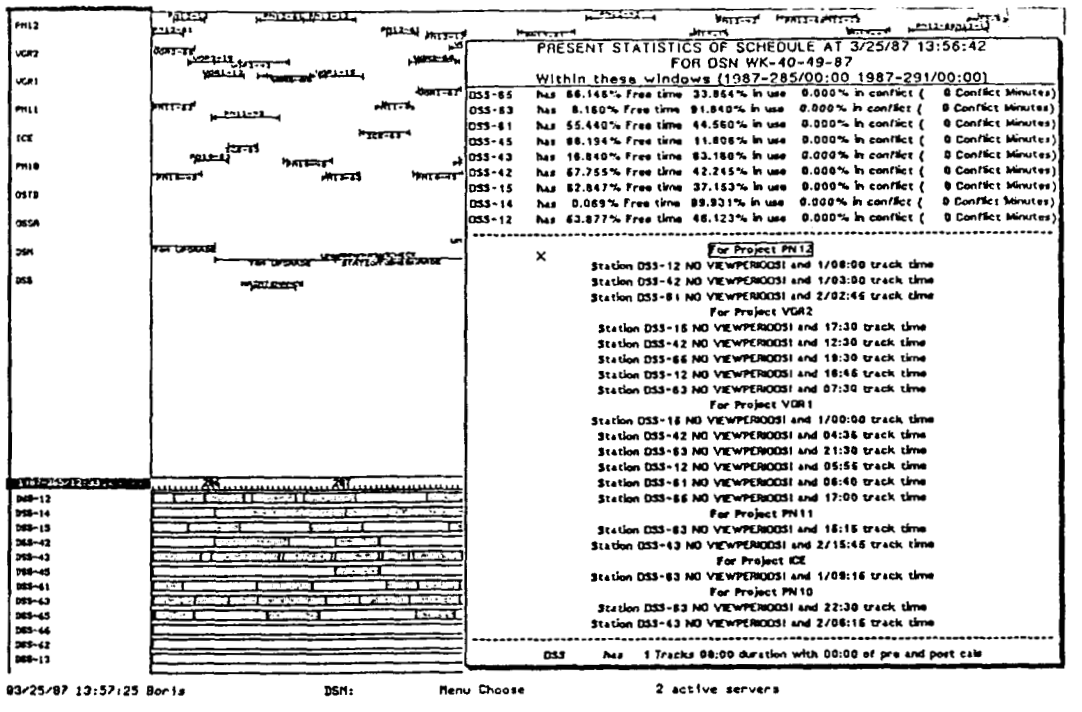


Figure 1. DSN Plan-IT Display with Statistics Menu

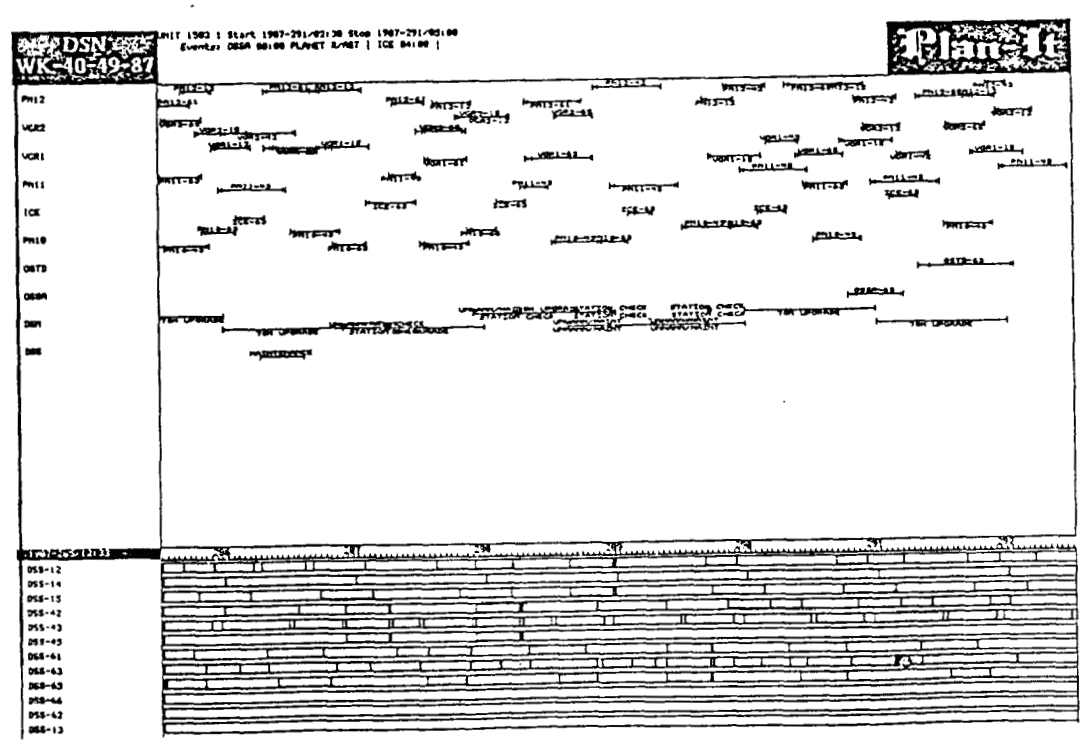


Figure 2. DSN Plan-It Display. Top pane shows information on the conflict area the user had moused on.

frequency of update to occur at whatever time interval he desires.

Via the menu interface, the user loads in the scheduling problem and data files. There are several different types of files Plan-It would accept for DSN scheduling, ranging from Plan-It's initialization files, viewperiod or targetting files, and the schedule file output from CARPA. The user's task is to iterate on the input requests or partially generated schedule and to finalize the schedule. During any point of the operation of Plan-It, the user can request via a menu to either save the present state of the schedule or view statistics of the resource usage. The statistics gives the user another quantitative means of measuring his progress toward completion of the schedule, rather than the graphical view that is always present in Plan-It. The saved schedule file can be later loaded in to resume scheduling from that point.

DSN RESOURCE ALLOCATION SCHEDULE GENERATION PROCESS

CARPA generates the initial schedule. After CARPA completes this phase of the scheduling process, the CARPA schedule file is transferred to Plan-It for further refinements. In many instances CARPA adequately resolves the initial schedule with some lower priority requests deleted. Once Plan-It receives the CARPA file, the deleted requests are brought back into the schedule. The resource lines representing the radio antennas utilized by the requests graphically depict the conflict areas. The user may mouse on the conflict area to obtain further information on the exact time frame and requests or activities contributing to the conflict. Seeing the conflicts and opportunities motivates the user to either edit or invoke specific heuristics to further resolve the schedule. This display representation can be seen in the figures under the menus. After many cycles of iteration between the user and Plan-It, the schedule will finally be completed.

This mode of operation shows that the Plan-It scheduling process is totally user-controlled. As the user edits the schedule, he supplies the intuition and motivation to apply and guide the supplied Plan-It heuristics, called strategies. The "user-natural" graphical interface of the program allows the user to see conflicts and opportunities as they arise from previous actions in Plan-It, whether initiated by him or the strategies. Upon viewing the results, the user can edit directly or invoke other strategies. This is the circular-action cycle that the user cooperating with the Plan-It employs to produce a schedule.

The most utilitized concept in the Plan-It system is the strategies. Strategies act as a library of simple DSN scheduling heuristics for use by the Resource Allocation Team. These strategies may be scoped by user-imposed constraints or modifications, specified by strategy-modification menus. For example, in DSN scheduling there is an activity-expansion strategy, the purpose of which is to expand any activity or request to its maximum allowable duration without causing conflicts. The fact that the user does not have to be precise on the amount of expansion or which activities to expand demonstrates the robustness of these strategies. Also, the strategy may be

modified by the user to expand about the middle of the activity or expand forward or even backwards. The user may further scope the strategy to take action only on non-conflicting activities of a certain class of projects that only use specific radio dishes within particular intervals of time. This broad flexibility of modifying the strategy further enhances the user interaction in a more satisfying scheduling process.

WHAT'S BEEN LEARNED

One thing learned from watching the Resource Allocation Team schedule the DSN is that a scheduler tends to avoid resolving the schedule in a chronological order. This jumping about to different time frames on the schedule during the scheduling process is a result of changing perspective or focus level. People look for opportunities and quick fixes. Initially, during the early phases of schedule development, the user lays out the requests in the schedule at their preferred locations and applies global strategies. This defines the general layout of the schedule. This action may produce conflicts throughout the schedule, but the user usually is not concerned with them until later, unless by changing his focus level he can quickly resolve a conflict that may appear during that process. As the user goes through the Plan-It action cycle, the types or pattern of conflicts shown cause the user to localize his focus level to the particular conflict or opportunity at hand. At this point, he may edit the specific activity, causing the conflict or invoking a strategy on the conflict itself to resolve it. Both the Plan-It strategies and the user monitor their performance from the resource lines. Presently, only the user is knowledgeable enough to change focus level and choose the order of invoking the strategies.

The last and most important feature emphasized in Plan-It's creation is cooperation with people. Approaching a complicated scheduling problem in a top-down, time-ordered programmatic manner does not work. Knowledge of the problem domain must be gathered from seeing how people deal with it. In the DSN scheduling domain, resource contention and tracking opportunities play a major role in determining how a person allocates his time and effort in resolving the schedule. Presently, Plan-It views the scheduling problem solely from one basic perspective: the resource lines. This single viewpoint forces the Plan-It strategies to be more algorithmic rather than intuitive driven, thus limiting the scheduling-resolving capabilities of Plan-It. But because the person actually sees what Plan-It sees, he can supply the conflict pattern and opportunity recognition, changing perspective and focus control as needed to resolve a scheduling problem.

CONCLUSION

Originally the Resource Allocation Team generated schedules manually. This manual operation was reduced in part by CARPA. However, even with an initial computer-generated schedule, the Resource Allocation team was

barely able to keep pace with the realtime generation of schedules, taking nearly a month to generate one month's schedule. The close interaction between Plan-It and the scheduler resulted in a rapid turnaround time for producing schedules. It is now possible to generate schedules for an entire year within two months. Plan-It's user-natural concept and graphical display increase the user's scheduling prowess by enabling him to readily see the results of the actions he performs in the schedule itself. But in spite of this improvement in scheduling performance, additional research is needed to address the issue of incorporating the user's intuitive abilities into Plan-It.

ACKNOWLEDGEMENTS

The Plan-It DSN implementation described in this paper was performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration (NASA).

The Plan-It development was jointly funded by the NASA Office of Aeronautics and Space Technology (CODE R) and the NASA Office of Space Science and Applications (CODE E).

The authors wish to acknowledge the special assistance of the Resource Allocation Team for their help.

The authors have been supported throughout the work by the efforts of the other members of the JPL Sequence Automation Research Group: Bill Dias, Sven Grenander, Win Lombard, David Mittman, Stephen Peters, Mark Rokey, Louis Rubenstein, John Sisino, and Jenny Wong.

REFERENCES

1) S. U. Grenander, 1985, "Toward the Fully Capable AI Space Mission Planner", Aerospace America, Vol. 23, No. 8

2) D. Woods, 1986, "Cognitive Technologies: The Design of Human-Machine Cognitive Systems", AI Magazine, Vol. 6, No. 4

3) E. Biefeld, 1986, "Plan-It: Knowledge-Based Mission Sequencing", Proceedings of SPIE on Space Station Automation, Oct. 1986, pp. 126-130.

4) K. A. Bahrami, E. Biefeld, L. Costello, J.W. Klein, "Space Power System Scheduling Using an Expert System", 21st Intersociety Energy Conversion Engineering Conference, San Diego, CA, Aug. 25-29, 1986

6) M. Rokey, S. U. Grenander, 1986, "Artificial Intelligence Planning Applications for Space Exploration and Space Robotics", AAAIC Conference Proceedings

7) W. Dias, J. Henricks, J. Wong, 1987, "Plan-It: Scheduling Assistant for Solar System Exploration". This publication

Space Station Platform Management System (PMS) Replanning Using Resource Envelopes

by
Joy Lee Bush, Anna Critchfield, and Audrey Loomis
Computer Sciences Corporation, System Sciences Division

Abstract

One of the responsibilities of the Space Station Platform Management System (PMS) is to maintain constraint-free, short-term plans for platform and free-flyer activities. Both the replanning function and the associated constraint-checking function are viewed as potentially requiring expert system assistance.

This paper describes the PMS Resource Envelope Scheduling System (PRESS) expert system, which is currently under development. PRESS capabilities will include the following:

- o Plan, replan, and perform constraint checking using resource envelopes resembling those required for telescience
- o Initialize itself using the results of a previous run
- o Infer the replanning needs associated with a change in resource availability
- o Allow the user to determine the level of interaction (including an advisory capability) with the system during execution.
- o Generate both a graphic timeline and a report as output

PRESS is being developed on an IBM PC/AT using TeKnowledge, Inc.'s M.1 expert system shell. PRESS activity definitions and constraints are based on those defined for the Cosmic Background Explorer (COBE) mission scheduled for launch in early 1989.

I. Background

The development of the Platform Management System (PMS) Resource Envelope Scheduling System (PRESS) is part of an on-going task contracted to Computer Sciences Corporation by the Mission Operations Division (code 511) and the Data Systems Technology Division (code 520) of NASA at Goddard Space Flight Center. PRESS is a prototype expert system being developed as part of an effort to study the feasibility of using expert system technology in the Space Station environment. Thus PRESS attempts to implement some of the functions that have been defined for the PMS, and to use the "resource envelope" concept that has been developed for Space Station applications but is not yet fully defined.

II. PMS Functionality

The PMS (as defined in the PMS Definition Document, October 1986) is a software system that provides operational management services among payloads and platform systems. It consists of an automated on board segment, the Platform Management

Application (PMA), and a ground segment, the Platform Management Ground Application (PMGA). Seven functions have been defined for the PMS:

(1) Short-Term Plan Management - modify, as necessary, a short-term plan in response to requests from operators, customers, subsystems and payloads;
(2) Schedule Execution - execute the short term plan by coordinating instructions to payloads and systems;
(3) Operations Monitoring and Activity Logging - track and store data for anomaly investigation and billing;
(4) Intersystem/Payload Testing - execute testing as desired by platform operators;
(5) Conflict Recognition and Resolution - recognize and prevent conflicting activities;
(6) Fault Handling - supervise fault management and reconfiguration for payloads and systems; and
(7) Transaction Checking - control messages to on board destinations.

III. PRESS as a Subset of PMS Functionality

The goals of PRESS are: to research the feasibility of expert system applications in providing PMS functionality; to use (and therefore help define) the resource envelope concept as a basis for automatic scheduling; to implement COBE functions as a proof-of-concept; and to evaluate the suitability of the system development environment for expansion of this prototype or more complex development/delivery efforts.

PRESS addresses two of the PMS functions. The first of these is the Short-Term Plan Management function. This function involves the PMGA receiving a plan from the Platform Support Center, and uplinking appropriate portions to the PMA. The PMGA and the PMA may receive plan changes requested by operators, customers, subsystems, and payloads. The second function, Conflict Recognition and Resolution, involves the monitoring of resource usage, allocation, and margins. Conflicts for resource usage are to be resolved on a priority basis. This function will be used in deciding whether a given request may be scheduled. The PMA and the PMGA are required to modify the short-term schedule while maintaining a conflict-free plan that does not exceed the platform's resource capabilities or compromise its safety.

Another PMS issue that will be explored via PRESS is how autonomous the scheduling/rescheduling functions can be made for eventual on board usage. The process of PRESS development will help to identify specific areas where human intervention is critical. Functions that can be safely migrated to the on board system will be identified.

PRESS will implement the short-term plan management function and the conflict recognition and resolution function using specific constraints from the Cosmic Background Explorer (COBE) spacecraft. PRESS will perform both initial scheduling and replanning to accommodate updates; will constantly update resource usage with each schedule modification; and will check constraints and limitations such as safety margins via a table lookup. PRESS is designed with provision for as much flexibility as possible, so that the demonstration of the rapid prototype can elicit feedback regarding which approaches appear the most useful. The prototype under development involves a smaller set of resources and constraints (primarily uplink and downlink time slots) than will be encompassed by the PMS, but the application will demonstrate basic functionality in

serving as a proof-of-concept for using the expert system approach in PMS functions. Figure 1 depicts the placement of an expert system that would include the PRESS functions in the PMS.

IV. PRESS Development Status

A final PRESS prototype system is planned for September 1987, with a rapid prototype demonstration scheduled for early May 1987. Table 1 lists the functions of the final prototype and indicates whether they will be implemented in the rapid prototype. Briefly, the rapid prototype will create a new schedule, and will accept scheduling requests either from a file or manual input. The rapid prototype expects some operator interaction and provides the advisory capability to suggest alternative time slots. It generates output in the form of a graphic timeline and a printed report. Feedback following the rapid prototype demonstration will be critical in determining the direction of further PRESS development. Section VI discusses PRESS functionality in detail.

V. PRESS Terminology

An "activity " is the item being scheduled. It may be anything from a complex scientific experiment to a single use of a communications link. An activity may thus consist of more than one event, with each event represented as one envelope. An activity is represented to PRESS in the form of one or more "resource envelopes".

Each "resource envelope" is equivalent to an event and represents a time period, one or more resources whose use is required, and a usage level (if appropriate). It is assumed (for scheduling purposes) that, within an envelope, resource usage is stable. Since "resource envelope" represents the requests for scheduling that will come to PRESS, we will refer to them as "request envelopes", in order to avoid confusion with the representation of the resources themselves.

Examples of "resources" are power, an instrument, a communications link, etc.

The "schedule" or "schedule timeline" is produced by the system to show what activities have been scheduled within a given time period. PRESS output shows the activities plotted against the resource used over time, so that the schedule timeline is actually a set of parallel timelines, one for each resource, with usage periods identified with an activity identifier.

VI. PRESS System Description

PRESS will accept user input interactively or from a stored file, and will be capable of creating a new schedule or modifying an existing schedule. PRESS will generate as its final product a schedule timeline, available both graphically and in a printed report. Some of the PRESS functions, described below have yet to be implemented; others exist in the current form of the rapid prototype. The discussion corresponds to the "and-or" graph of PRESS functions shown in Figure 2, with some key issues expanded in section VII.

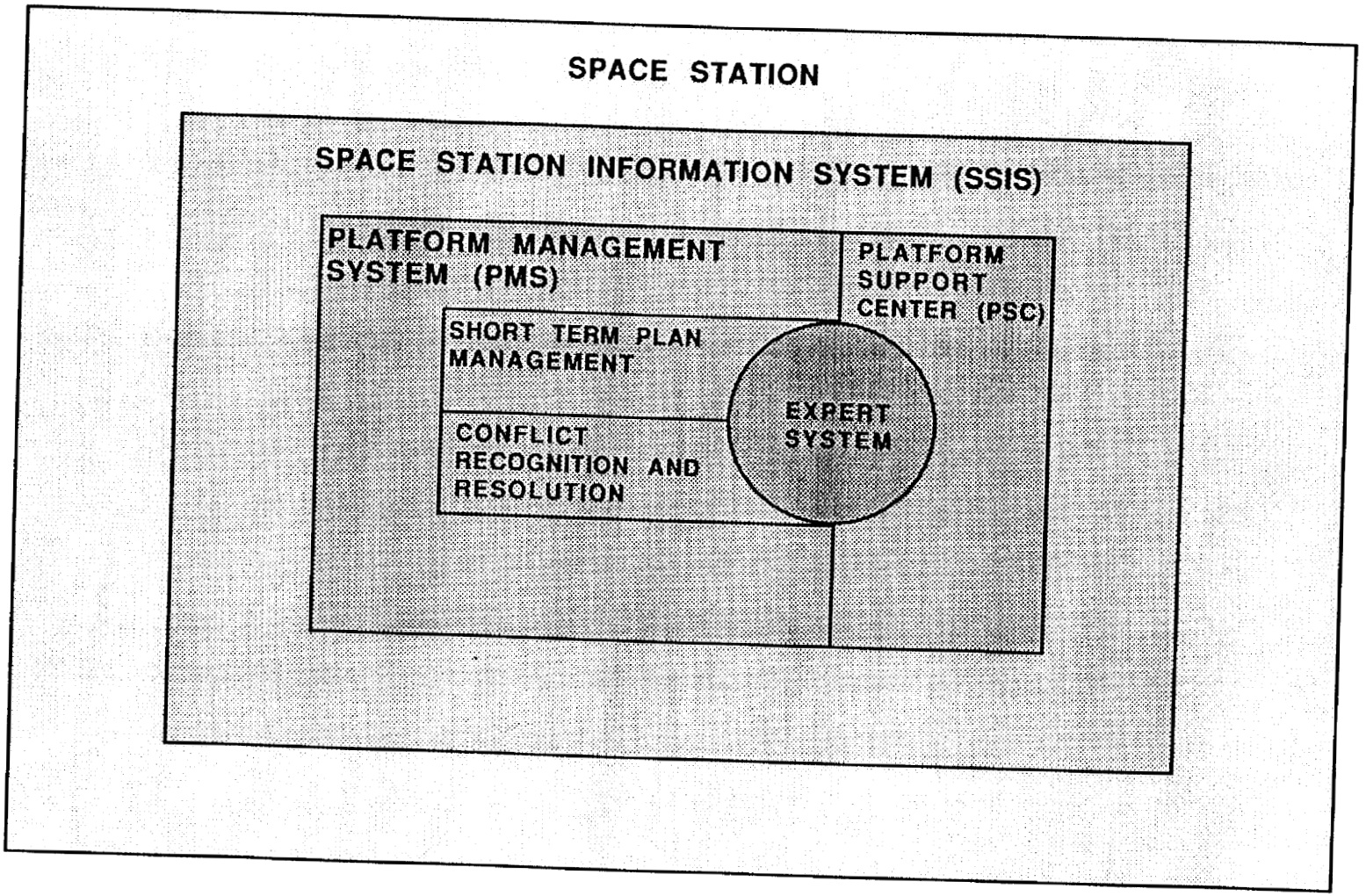


Figure 1. Placement of Expert System in PMS

FUNCTIONS OF FINAL PROTOTYPE	RAPID PROTOTYPE
perform initial planning	yes
-- process requests by priorities	yes
perform replanning	adds only
accept schedule modification requests as resource envelopes	yes
accept schedule modification requests as resource availability changes	no
resolve schedule conflicts based on assigned envelope priorities	no
perform event conflict checking via table look-up	no
accept multiple envelope requests	no
change scope of user interaction at operator's discretion	requires interaction
-- provide advice identifying modifications to the resource envelope requests that would permit successful scheduling	yes
-- provide capability for operator to cease processing before completion	yes
perform input error checking	
-- on query responses	built-in only
-- on time format	some
generate new schedule timeline output	no
-- graphic representation	yes
-- report tables	yes
	yes

Table 1. PRESS Functions

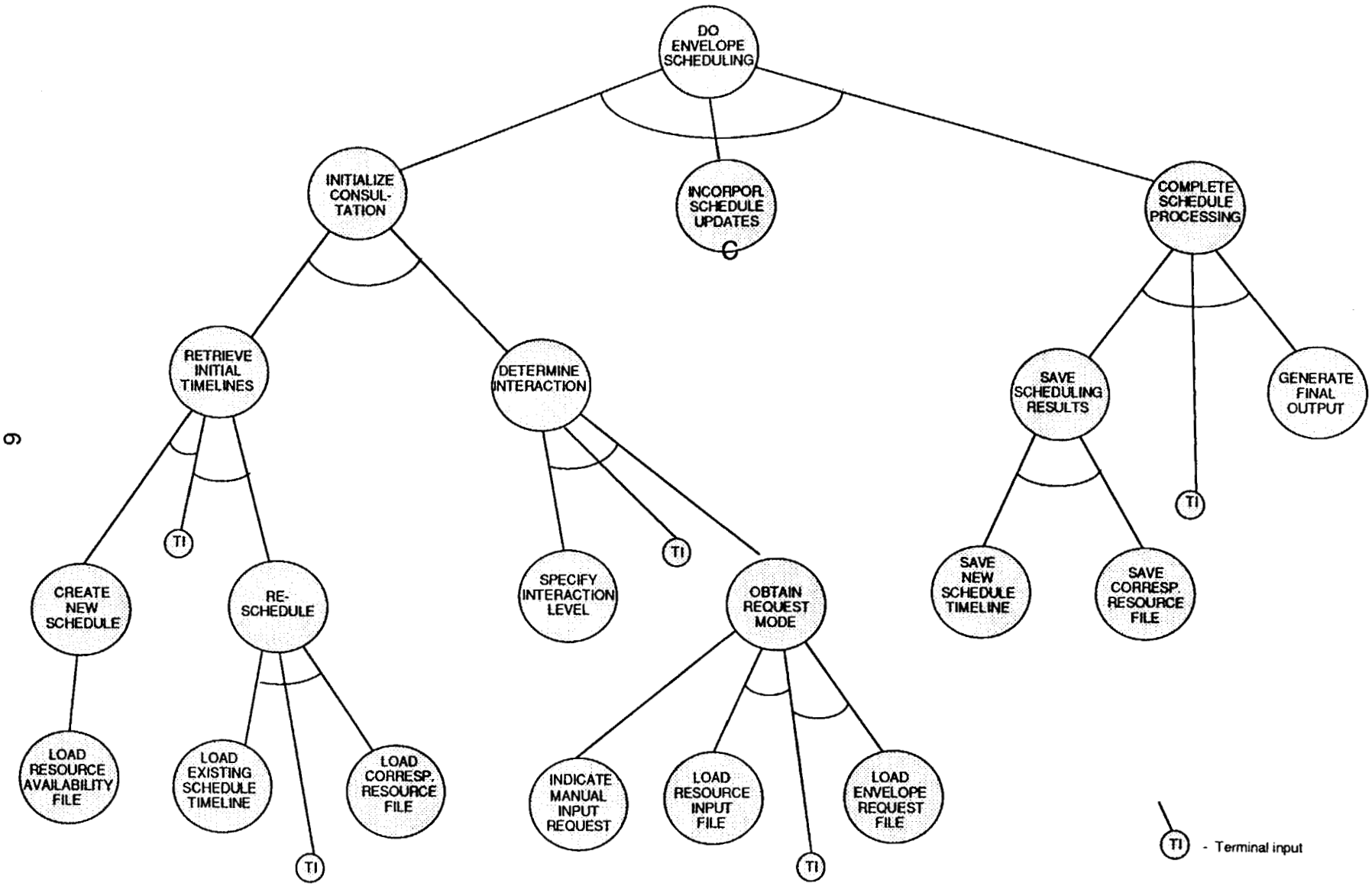


Figure 2. And-Or Graph of PRESS Functions (1 of 3)

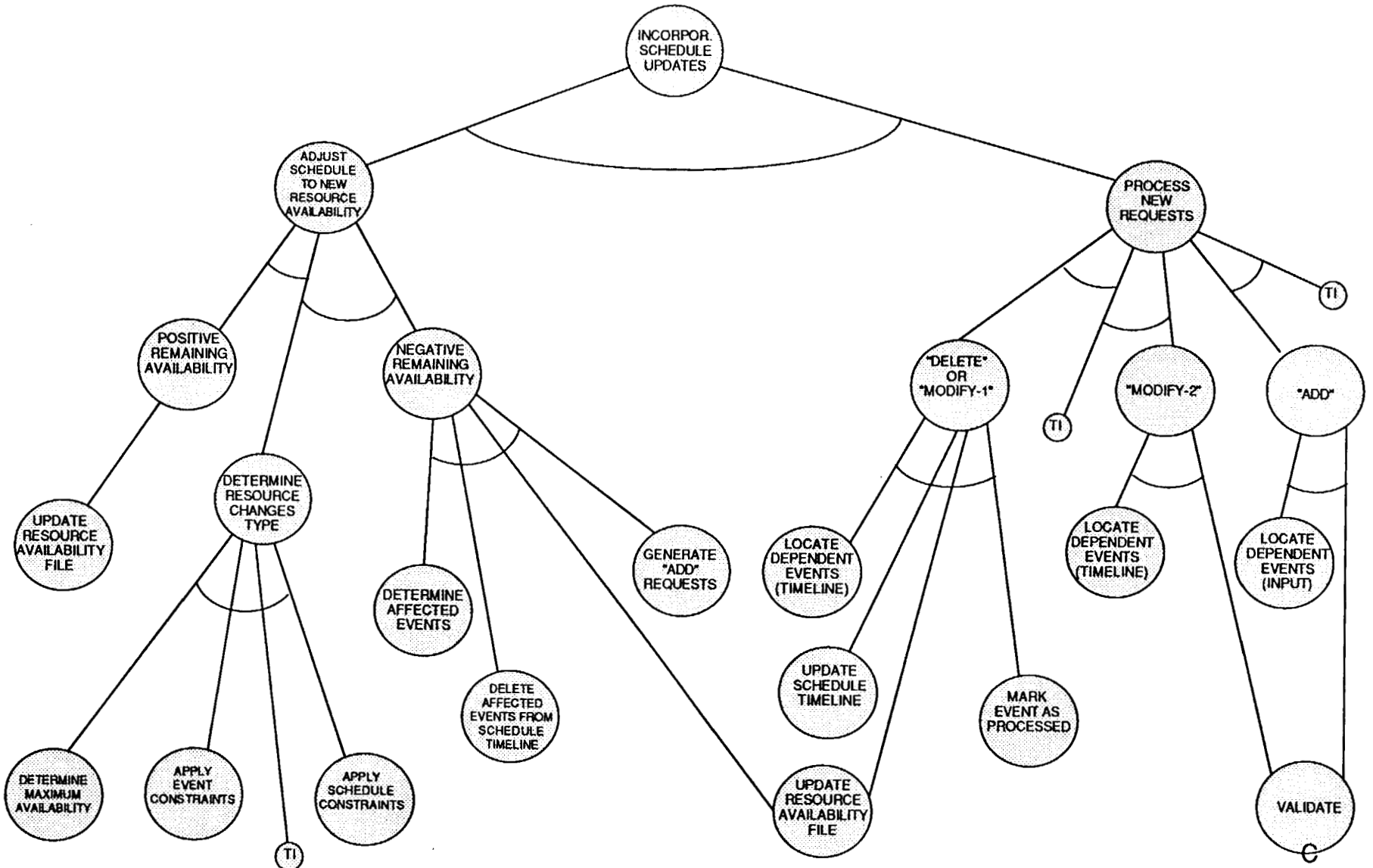


Figure 2. And-Or Graph of PRESS Functions (2 of 3)

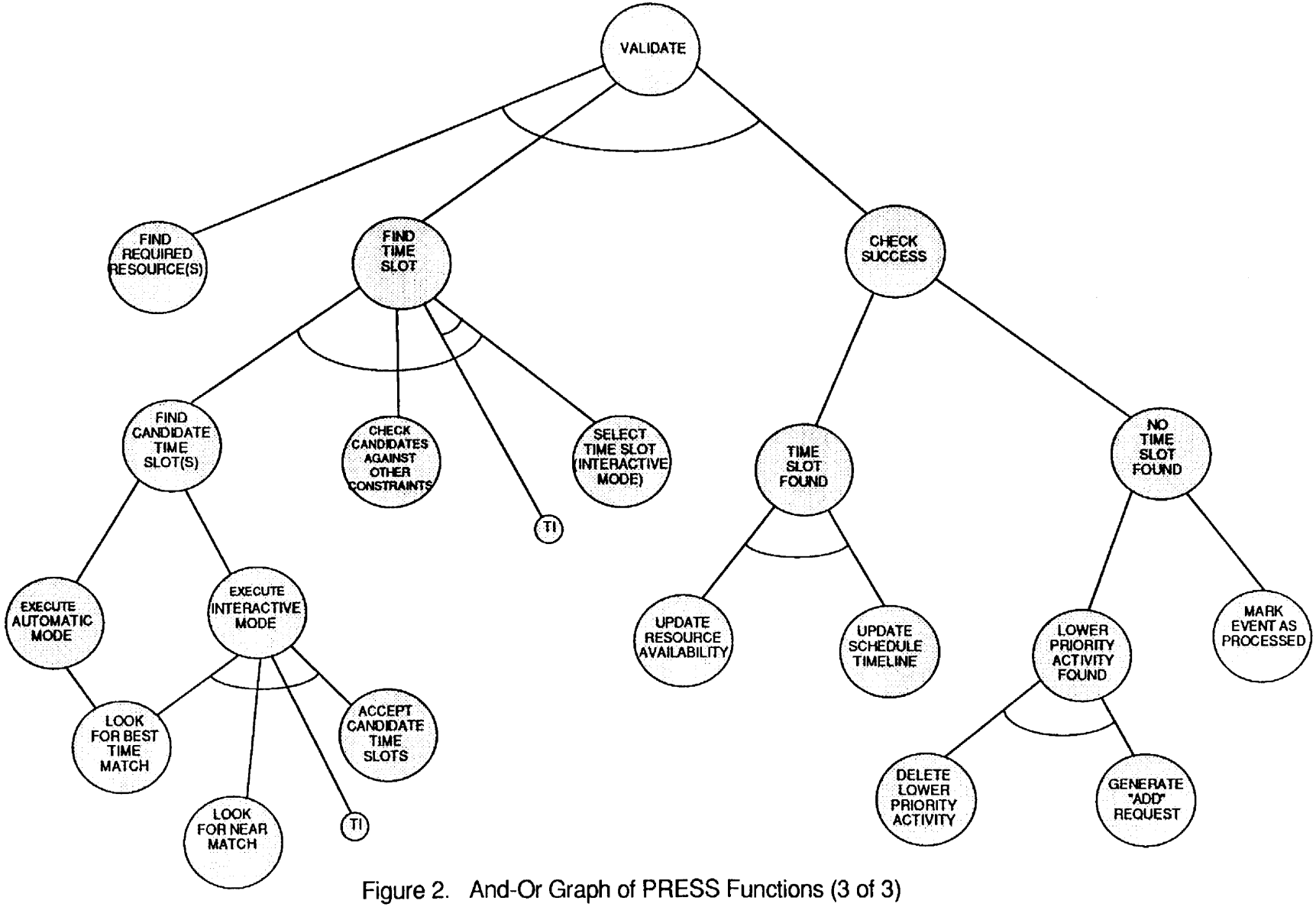


Figure 2. And-Or Graph of PRESS Functions (3 of 3)

Initialize Scheduling Consultation

Initialization will be performed once for each consultation session. The user must input information regarding the data input "mode", the location of the initial timeline, and the required level of interaction. These modes will be active through the entire consultation and cannot be changed within the same session.

PRESS will load initial files based on user-provided input including the start and end times for the requested schedule; whether the request is for the creation of a new schedule or for the replanning of an existing schedule; and the designation of the schedule immediately preceding the current schedule chronologically. For an initial planning effort, the schedule timeline will be empty, and resource availability information will be loaded from the default Resource Availability file. For replanning (modifying an existing schedule), the user-specified files, including the existing Schedule Timeline, and the corresponding Resource Availability file, will be loaded.

The user will determine the mode of system execution, including the level of operator interaction (automatic or nonautomatic) and the request mode (schedule plan/replan or change in resource availability). If the nonautomatic mode is selected, the user can specify a number of requests (N), so that, after the automatic processing of every N requests, control returns to the operator. This allows the examination or graceful termination of the system before processing of all requests is completed. Also in nonautomatic mode, an advisory feature is available which allows PRESS to widen the requested time windows to present some alternatives for an otherwise unschedulable request. This is discussed in more detail below. The user may also select manual request input, which requires the user to input request envelopes manually on a one-by-one basis. Automatic mode executes scheduling without operator interaction and without the advisory capability.

Incorporate Schedule Updates

Requests for changes in resource availability are satisfied by processing modifications to the Resource Availability file. Requests for activity scheduling or rescheduling are satisfied by processing the requests for addition, deletion, or modification of activities in the form of request envelopes. The output of this function will be the modified Schedule Timeline and the Resource Availability file.

PRESS will accept input identifying resource availability changes such as equipment failures. This type of input will include times, if applicable, and the corresponding, changed resource limit(s). Two special processing steps are required. First, the available resource level must be updated by inserting the new limits. Second, all scheduled activities drawing on the changed resource must be reexamined; they may not still be schedulable.

PRESS will compute new resource availability as a function of the following components:

$$R = F(O, N, S, C)$$

where R - new remaining resource availability (can be positive or negative)
O - currently available level of resource

N - requested changes of resource availability
S - scheduled activities drawing on the resource
C - other constraints applied to the resource.

A positive or zero value of the remaining resource availability means that the existing Schedule Timeline is still valid and to complete the consultation, it is only necessary to update the Resource Availability file. Negative value of the remaining resource availability means that the existing Schedule Timeline is no longer valid. To resolve the negative resource availability level, PRESS will identify the affected activities and delete them from the existing Schedule Timeline, creating a positive resource availability level. The changes will be reflected in the Resource Availability file, and requests to add those activities back to the schedule will be generated; they can thus be rescheduled in priority order up to the limit of the available resources.

PRESS will permit users to request the deletion or modification of a previously scheduled activity, possibly involving a series of envelopes. PRESS will automatically locate all envelopes affected by the changes and update the Schedule Timeline and the corresponding Resource Availability file. Should the modification involve an increased use of resources, PRESS will locate dependent events on the Schedule Timeline and will try to reschedule all involved events. The new resource requirements will be validated in the same way as adding a new request.

In adding a single request envelope or a series of request envelopes comprising a single activity, the system will first check the list of requests for any related envelopes. All the related envelopes will then be validated together.

Validation

PRESS will begin the validation process by locating the requested resource(s) in the Resource Availability file. When these resources are found, PRESS will try to schedule an activity exercising the maximum duration scheduling and conflict handling concepts (these will be discussed in detail in Section VII). Once a time slot has been found, the legality of the activity at that time will be checked against other constraints, such as incompatibility with any already scheduled activities or orbital events. If no conflicts are encountered, the request is flagged as successfully scheduled. Otherwise it is passed to the conflict resolution function.

When running in fully interactive mode, PRESS will execute an advisory feature which suggests to the user, in cases where no match is found within the initial (i.e. envelope-specified) time range, what "compromises" can be made in moving the start time or shortening the duration in order to successfully schedule the request. If the user agrees to consider the suggested (altered) times, they are added to a list. When the system has exhausted possible compromises, the user is presented with the list of candidate time slots and asked to choose one or none. If the user selects none, the request is not scheduled; otherwise, the selected times are used, and the request is flagged as successfully scheduled.

Figure 3 depicts the situations where this advisory feature can be applied. In Case 1, PRESS has located an available resource within the requested duration window, but not

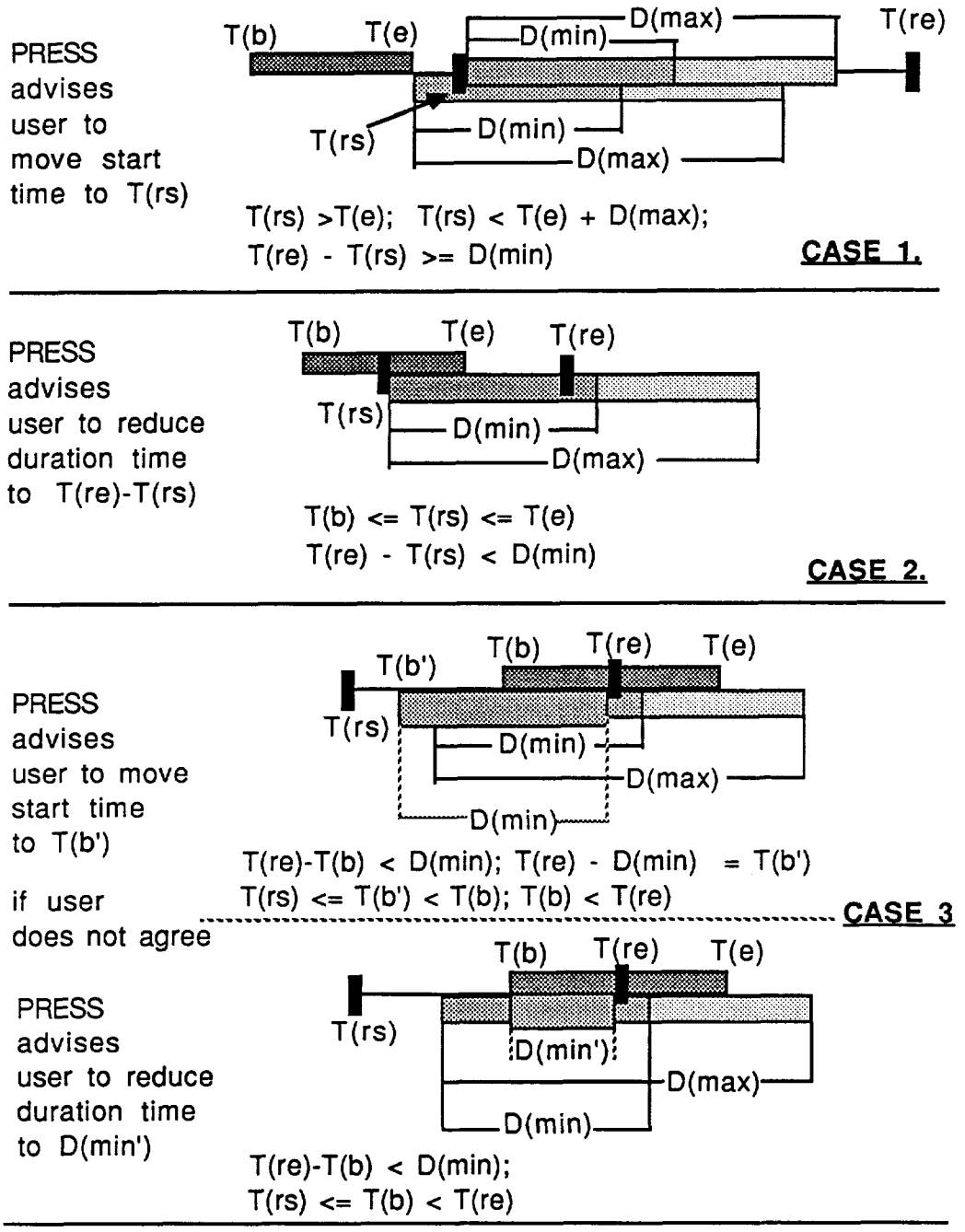


Figure 3. PRESS Advisory Feature Applications

within the requested start time window. PRESS therefore computes the earliest available start time, which in this case will be the resource availablity start time, and advises the user that if the start time may be moved, then a candidate scheduling slot has been found. Case 2 illustrates a possible compromise in altering duration. Case 3 illustrates the situation in which the possibility of either altering the start time or shortening the duration may be considered.

If the scheduling attempt is successful, the updates are reflected in the Schedule Timeline and the Resource Availability file. If the attempt fails, "delete" and "add" requests will be generated for the lowest priority conflicting activity, and PRESS will choose the next request for processing. Because "delete" requests have the highest priority, the requested deletion will be processed before another attempt is made to schedule the activity. This process will iterate until either the current activity is scheduled or the current activity becomes the lowest priority activity and is deleted itself. In that case, the request is marked for no further processing and is included in the final report to the user as an unscheduled request.

The final Schedule Timeline will be saved and reported to the user at the end of the run. The user will have the option of reentering the scheduling process before exiting from PRESS. The Schedule Timeline and the Resource Availability file will be stored along with continuity information regarding how the time period for the current schedule fits with other scheduled runs. Graphics representing the Schedule Timeline will be displayed along with an option to display the resource timeline and continuity information. The user will be provided with information on any requests that could not be satisfied and will be given an opportunity to place any replanning requests for the scheduled time. A file containing a printable report will be generated at the user's request.

VII. Key Issues Addressed by PRESS

Request Envelope Types

Activity scheduling requests will be typed according to the following: "delete" a currently scheduled event; "modify" a currently scheduled event; and "add" an event not yet scheduled. "Modify" requests are divided into "modify-1", a request that involves no increase of any resource utilization and a decrease in the use of at least one resource, and "modify-2", a request involving either the same or an increased level of resource utilization.

Priority

A two-step scheme is envisioned for PRESS. Its intent is to minimize internal rescheduling/backtracking by ensuring that PRESS is always doing the most important job of which it is aware. Priority conflicts will occur primarily when a high-priority activity must be added to a previously prepared schedule.

The first criterion for determining the priority of an input request is the activity request type. Before assigning priority, PRESS must examine "modify" requests to classify them as either "modify-1" or "modify-2". "Delete" and "modify-1" requests, processed in any

order, are PRESS top-priority actions, because they return resources to the system. "Add" and "modify-2" requests are considered when there are no outstanding "delete" and "modify-1" requests. Their priorities are assumed to be supplied externally. The chosen request is marked after being processed by the system, whether successfully scheduled or not, so that it will be ignored when the next request is chosen.

Maximum Duration Scheduling

Each envelope submitted for scheduling will have an associated start time window (earliest and latest start time) and duration window (minimum and maximum duration). First, PRESS will attempt to schedule the earliest possible start time within the start time window and the maximum duration. If this attempt fails, PRESS will try to find the earliest start time and the maximum duration allowed by the resources' availability, still within the requested time window. Figure 4 illustrates possible ways in which an optimum time slot may be found.

This strategy tends to minimize backtracking by assuring that an envelope is never considered unschedulable because the wrong subset of acceptable time was chosen for the envelope. The strategy also biases scheduling in favor of high-priority activities, by assigning them the maximum amount of available time. (This scheduling strategy could be refined by giving the conflict resolution function awareness that a low priority activity might be scheduled by decreasing the duration of a conflicting high priority activity).

Multiple Envelopes

One of the key issues yet to be handled is the scheduling of multiple-envelope requests (i.e. activities composed of more than one event, and thus represented by more than one envelope). Scheduling a multiple-envelope activity imposes additional constraints on the system due to the need to schedule all of the activity component events, in the proper sequence. The scheduled times of the multiple-envelopes may involve mutual interdependencies (e.g., no time gaps permitted between envelopes, or specified time gaps required between envelopes). The system must be able to recognize an envelope as part of a series representing a single activity, both on input for scheduling and on making any schedule adjustments. When, after scheduling, any envelope in a series is adjusted, all other envelopes must also be reconsidered. We have not yet finalized an approach to handling this issue, but we make the following preliminary assumptions: requested time windows in a multiple envelope activity must contain no gaps, and the envelopes must be scheduled back-to-back chronologically. The approach we will try initially will involve the generation of "delete" and "add" requests for the entire series. We believe, particularly for this issue, that it is crucial to emphasize the use of frequent rapid prototyping and expert input in order to better define and refine the optimal correlation between data representation, user interface, and the PRESS knowledge base.

Constraint Checking

Envelopes active at the same time may conflict in one of two ways: they may

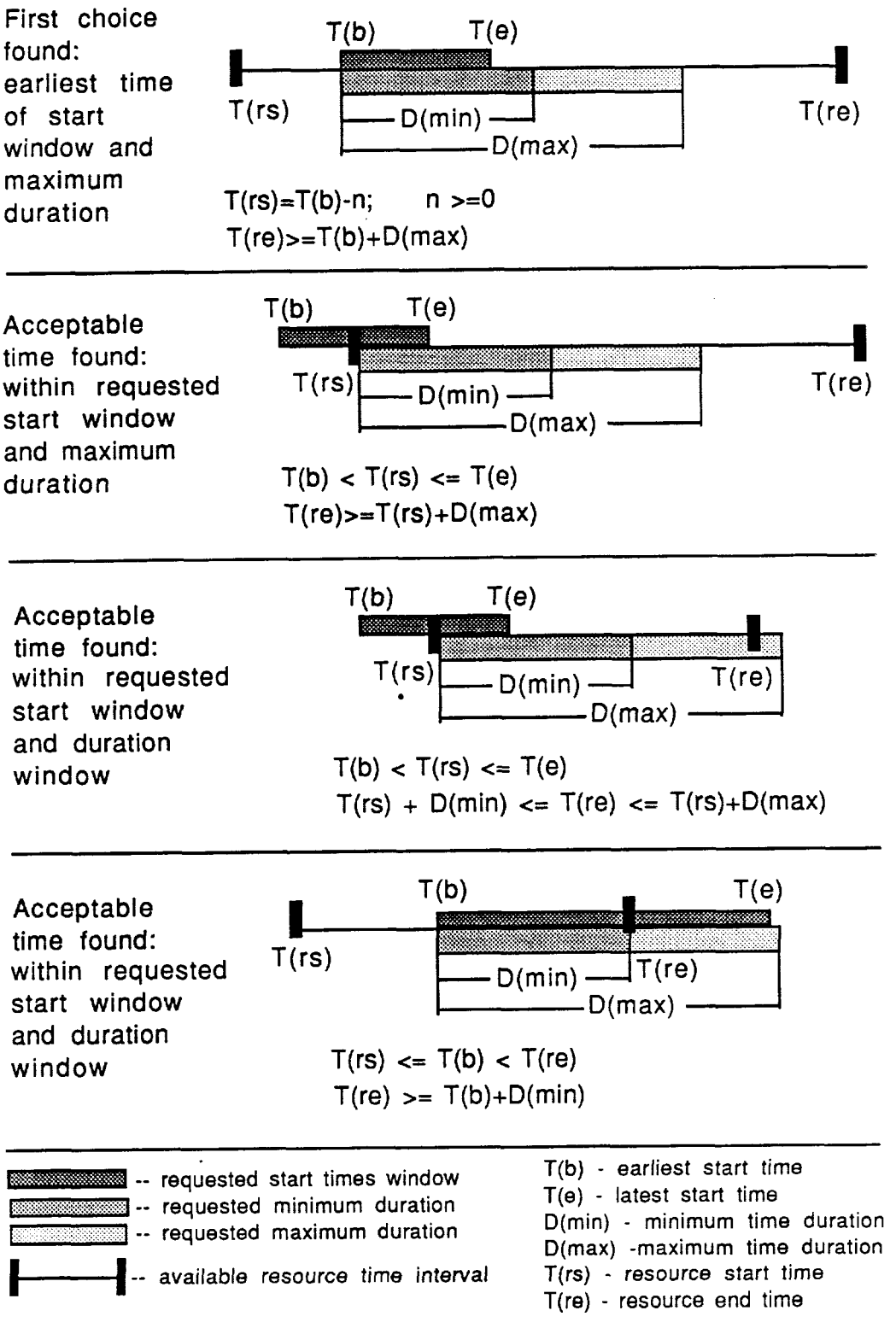


Figure 4. Optimum Time Slots with Maximum Duration Approach

oversubscribe available resources or they may require incompatible operating conditions. The first type of conflict is handled via the maximum duration scheduling approach. Operating condition constraint violations (e.g., incompatible experiments) will be checked, possibly with a combination of table lookup and rules in the knowledge base. We recognize that contextual information is critical for performance of this function.

VIII. PRESS Input and Output

PRESS will expect scheduling requests in the form of request (resource) envelopes, where each envelope represents an event, and one or more events comprise an activity. Figure 5(a) depicts the request envelope definition as it is input by the user and with the additional fields added for internal use by PRESS. The resource envelope includes a list of resource/usage-level pairs. The user will have the choice of file or manual input for scheduling requests. If the former is selected, PRESS will accept an ASCII file consisting of envelopes in list form with the required fields separated by commas. If manual input is chosen, the user must type in the envelope, when prompted by the system.

Other input required by PRESS includes a file defining the available resources and, for the replanning function, files containing the existing Schedule Timeline and the corresponding Resource Availability file.

PRESS represents each resource in list form. Figure 5(b) depicts the fields contained in a resource definition. PRESS dynamically creates and destroys resource representation lists. For an initial plan, PRESS loads a default Resource Availability file. For a replan, PRESS loads the Resource Availability file corresponding to the Schedule Timeline being considered.

PRESS outputs the result of the scheduling activity in the form of a graphic timeline and a printed report. PRESS generates the timeline dynamically, using information from an output file containing the scheduled request envelopes, annotated with the times actually scheduled. This same file is used to generate the printed report. The Resource Availability file is output containing the resource representation lists that match the scheduled activities.

IX. PRESS Development Environment

M.1 by TeKnowledge was chosen because it appeared to have many of the capabilities desired at the start of this task. These include the capability of easy interface with a language outside of the system domain, in this case, C; both forward and backward chaining capability; flexible control structures; provision for use of meta-knowledge; data representation which include a list structure; and built-in explanation facilities. In addition, the system is already into its third release and is well supported.

We are unable to provide an accurate evaluation of the M.1 shell at this point in development, but a full evaluation of the shell for the PRESS application will be part of our final report.

The M.1 software runs on an IBM PC/AT with a Video 7 EGA graphics board and an NEC Multisync color monitor. The operating system is IBM PC DOS, version 3.10.

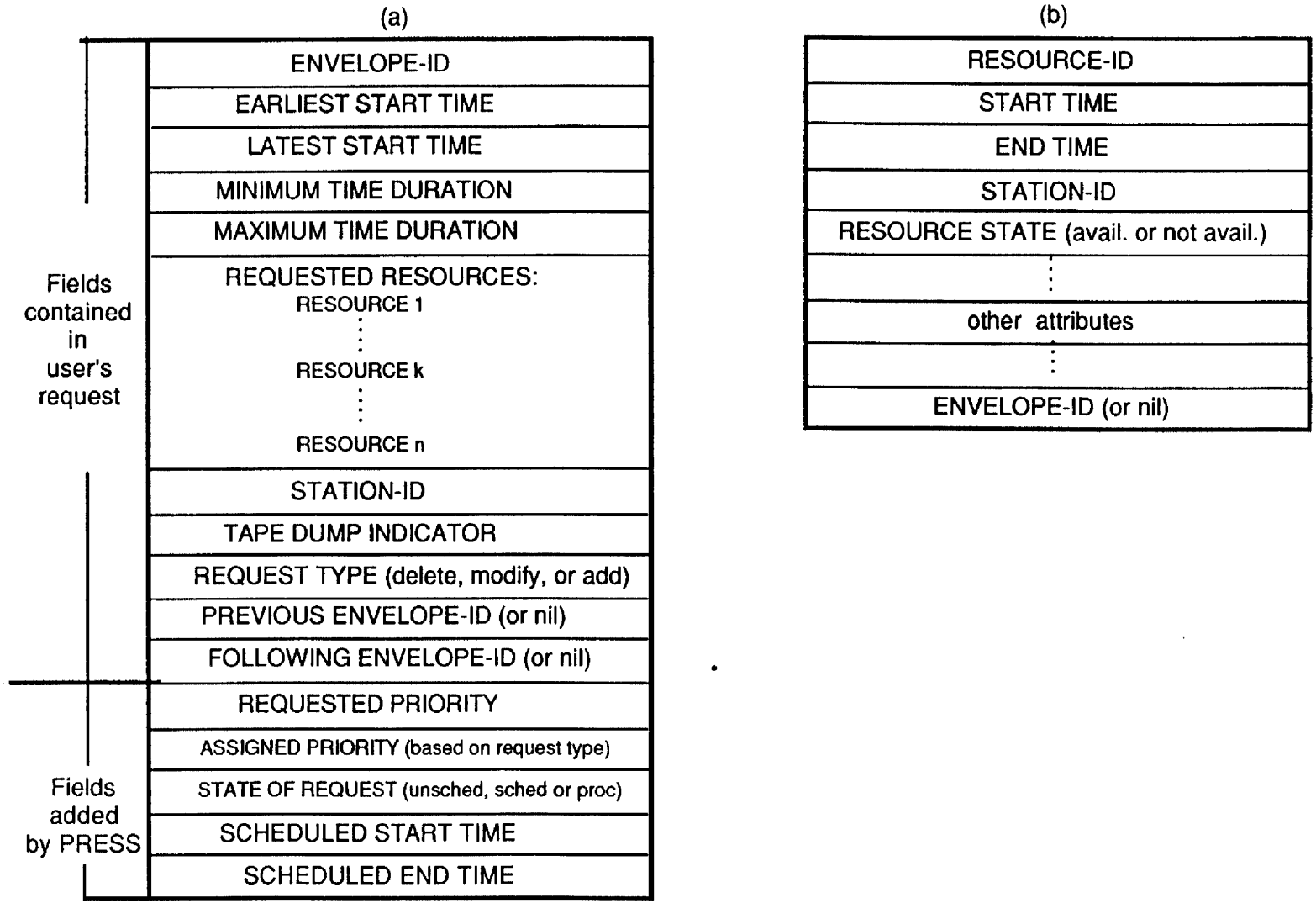


Figure 5.
Request Envelope and Resource Availability Representation

X. Comments

Several areas requiring particular attention have come up during the course of this project. The question of the nature of the user interface remains, especially in terms of the most convenient and useful input and output. Some new information on this issue is expected at the time of the rapid prototype demonstration, but the most informed feedback will arise only when PRESS is tested by operators in an approximate real-life situation. Another side of the user interface question is the level of human interaction required. Since PMS aims include the eventual migration of software from ground to on-board, we are attempting to give this issue some attention. Our initial response has been to make the system as flexible as possible by allowing the level of human interaction to be specified at the start of a consultation. Currently, if the fully automatic mode is chosen, PRESS will be unable to extend any additional flexibility in time-slot scheduling beyond that provided in the original request envelope. PRESS may help to identify types of requests to which automatic scheduling option may be applied. This function requires additional knowledge allowing the system to choose, on its own, from a list of candidate schedule opportunities.

Two of the most difficult technical issues to be solved by PRESS are multiple-envelope handling and constraint checking. These issues have been only superficially discussed here because our approach to solving them has not yet been fully defined. Both of these areas will require further definition by area experts, and it is hoped that the PRESS prototype will help to elicit that information.

Acknowledgements

The authors wish to acknowledge Ed Lewis (GSFC code 520), Dolly Perkins (GSFC code 520), Steve Tompkins (GSFC code 511), Steve Wadding (GSFC code 511), and Larry Zeigenfuss (GSFC code 511) for their encouragement and helpful discussions. This work was funded by the National Aeronautics and Space Administration, Goddard Space Flight Center for the Mission Operations Division (GSFC code 510) and the Data Systems Technology Division (GSFC code 520) under task assignment 319000.

References

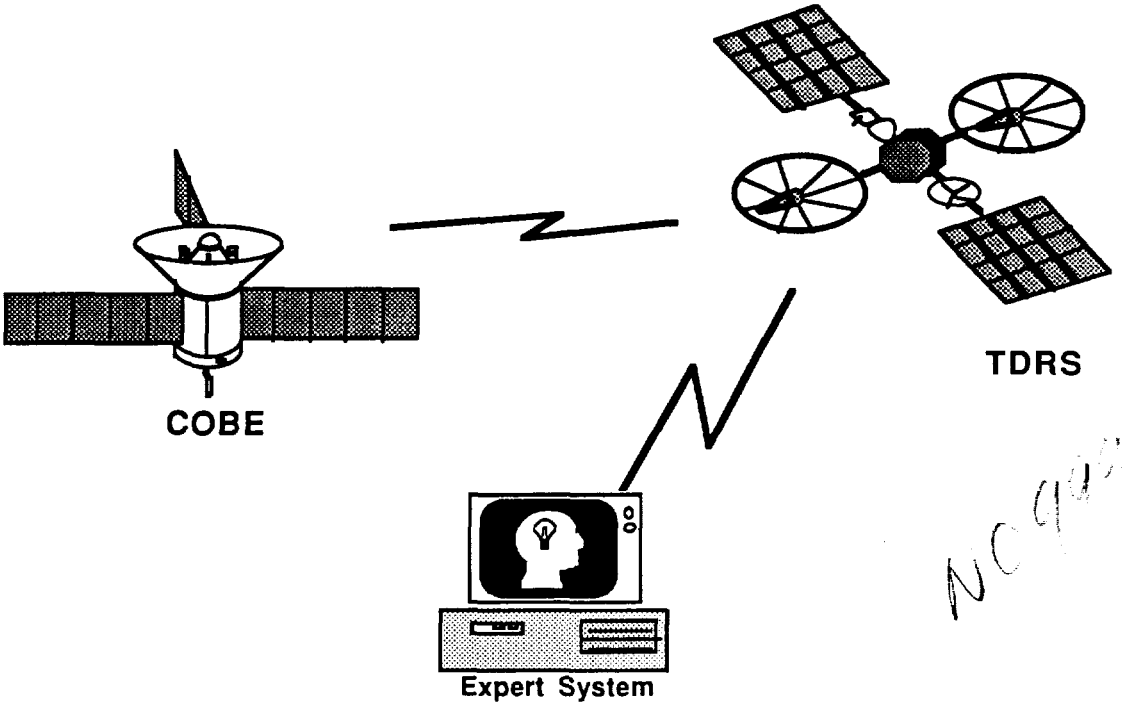
"The Platform Management System Definition Document", GSFC, October, 1986
"Expert System Feasibility Study Report", Computer Sciences Corporation, September, 1986
"The PRESS Functional Definition" (initial input), Computer Sciences Corporation, February, 1987

Section B

Fault Isolation/Diagnosis Expert Systems

CLEAR

Communications Link Expert Assistance Resource



Larry G. Hull
Project Manager

Peter M. Hughes
Knowledge Engineer

National Aeronautics and Space Administration
Mail Code 522.1
Goddard Space Flight Center
Greenbelt, Maryland 20771

ABSTRACT

Communications Link Expert Assistance Resource (CLEAR) is a real time, fault diagnosis expert system for the Cosmic Background Explorer (COBE) Mission Operations Room (MOR). The CLEAR expert system is an operational prototype which assists the MOR operator/analyst by isolating and diagnosing faults in the spacecraft communication link with the Tracking and Data Relay Satellite (TDRS) during periods of realtime data acquisition .

This paper discusses the mission domain, user requirements, hardware configuration, expert system concept, tool selection, development approach, and system design. Development approach and system implementation are emphasized. Also discussed are system architecture, tool selection, operation, and future plans.

INTRODUCTION

The Communications Link Expert System Resource (CLEAR) is an expert system to be implemented and demonstrated for the Cosmic Background Explorer (COBE) Payload Operations Control Center (POCC) as a joint project between the Mission Operations Division and the Data Systems Technology Division within the Mission Operations and Data Systems Directorate at Goddard Space Flight Center. The purpose of this joint project is to design and implement an expert system to monitor and isolate faults of the COBE and Tracking and Data Relay Satellite (TDRS) communications link and to provide advice to correct these faults.

BACKGROUND

NASA's Tracking and Data Relay Satellite System (TDRSS) has placed the responsibility of configuring, monitoring and troubleshooting many types of spacecraft communications upon the analysts at the consoles in the realtime environment of the POCC. The result is a complex task which, if not handled quickly and properly, can result in poor utilization of TDRSS services, inefficient spacecraft operations and potential hazards to spacecraft health and safety.

Operating the spacecraft communications links with the TDRS requires realtime evaluation of a mission oriented subset of more than 100 configuration and performance parameters and requires knowledge of both TDRSS services and spacecraft communications systems. This evaluation of realtime data must be correlated with an understanding of these services and systems both to isolate problems and to select appropriate courses of action to resolve identified problems.

At present, extensive training and communication of actual experience are used to develop MOR analyst capabilities to the fullest. Regardless, the size of the task places a large burden on the operator. It is the objective of automation, specifically utilization of an expert system acting as an advisor, to produce a more reliable, more efficient and less error prone system of operations.

Spacecraft communication links with the Tracking and Data Relay Satellite and TDRSS services are used routinely and by many missions. This gives the CLEAR system a very high utility, particularly if only minor modifications are needed to allow other missions to use the system.

The rationale for choosing the COBE spacecraft communication links as the domain was timeliness. Ground system preparation for the mission including acquisition of MOR equipment was just starting at the time of the decision to develop the expert system. This equipment included computer workstations of sufficient power to support an expert system and the designers felt that one workstation would normally be available during operational periods.

MISSION DOMAIN

The COBE, a single observatory mission in the area of astrophysics and specifically cosmology, will be launched in early 1989 and placed in a 900-km altitude, circular Sun-synchronous terminator (twilight) orbit. The COBE will use the TDRSS single access (SSA) S-band service for nominal on-orbit tracking, command, and telemetry support.

The COBE ground system will acquire data via both the Ground Network (GN) and the Space Network (SN). The GN will provide the interface to the COBE POCC and the Sensor Data Processing Facility (SPDF)

for the COBE science dump data. The SN will provide realtime data acquisition, command interface and tracking.

The Flight Operations Team (FOT), operating from the COBE POCC, will perform and participate in mission planning, realtime telemetry evaluation, off-line in-depth analysis, COBE-unique data base inputs, and assist in descrepancy and enhancement reporting, software testing, experiment-scientist interfaces, command management and generation, orbit and attitude-data coordination and data accountability.

COBE realtime operations will consist of four or five 20-minute TDRSS SSA forward and return events per day and will be used for uplinking stored commands, ranging, time management, and observatory safety and health monitoring. The CLEAR will support these periods of realtime operations.

The COBE POCC will be located in the Multi-Satellite Operations Control Center (MSOCC). A dedicated COBE Mission Operations Room (MOR) will be provided. Telemetry data from TDRSS or the ground receiving station will enter MSOCC via Nascom circuits. The data will enter the Telemetry and Command (TAC) computer and processed in the Application Processor (AP) computer for display at keyboard CRT's in the MOR.

The COBE Project Flight Operations Team performs the function of controlling the COBE satellite. MOR facility requirements include console facilities for three operations positions. One of the three positions will be used for CLEAR.

USER REQUIREMENTS

The following are the functional and performance requirements specifications for the Communications Link Expert Assistance Resource. The information herein was extracted by the CLEAR knowledge engineers working with available TDRSS and COBE documentation and with the domain expert.

To carry out its task, the CLEAR system will perform the following five functions:

Data Conversion,
Configuration Checking,
Communication Link Monitoring,
Fault Diagnosis, and
Event Logging.

CLEAR is to have no effect upon the COBE POCC processing systems and is to be transparent to other MSOCC systems. The CLEAR system will be a strictly passive component of the system supporting COBE realtime operations.

CLEAR is to be transportable within the MOR The system will run on any Engineering Analysis Workstation (EAW) in the COBE POCC without hardware modification and with the same operating system level software, e.g. communication package, graphics routines and device drivers, used by other application programs.

CLEAR is to use a standard communication package to be developed for POCC workstation applications. The data furnished by the AP will be ordered (positional not keyword) ASCII text (alphanumeric not binary). The system will extract the TDRSS performance data Operations Data Messages (ODM) and spacecraft status parameters from the communication buffer and convert them to the internal format required by the expert system.

COBE and TDRSS configuration parameters

for the scheduled event are to be set at initialization of the expert system. The CLEAR system will allow the operator to input values for configuration parameters, current date and clock time. The system will log the configuration parameters for the event. These parameters will be utilized by CLEAR to check for the start of the event and for the correct event configuration and will notify the analyst of any discrepancy.

The analyst is to be able to correct the data if the error lies in the CLEAR configuration parameters rather than in the COBE or TDRSS configuration. After the corrected data has been entered, the expert system can be reset and restarted.

CLEAR is to be driven by ODM and status data sent by the AP. The system will monitor the input data (communication buffer arrival) frequency and will warn the analyst if input is not received within the expected (4 to 5 second) interval. The expert system will monitor the communications link parameter values to detect and isolate faults.

CLEAR is to diagnose the faults identified during an event. The system will determine possible sources or causes of a fault, rank multiple possibilities in order of probability and present the result to the analyst. The system will also recommend course(s) of action that might be taken by the analyst. If requested, the system will explain the reasoning used to diagnose a fault and recommend a course of action.

CLEAR is to log all expert system activity for post event analysis. The system will time tag all identified faults and will record the inferences, the diagnoses, the recommendations offered and the actions taken as (and if) indicated by the analyst. The system will provide non-realtime utilities to print a formatted copy of the log, to trace and analyze the activity of the expert system during the event and to extract statistics for evaluation of system performance.

CLEAR is to operate in realtime with a performance requirement derived from the expected 3 to 7 second communication buffer (input data) arrival frequency. The expert system will convert input data, check parameter values and perform inferences within this time interval. Event logging, operator dialog and explanations are not realtime events subject to the performance requirement.

HARDWARE CONFIGURATION

The computer on which the CLEAR will run will be any one of the three workstations being used for console operations in the COBE MOR. These workstations are IBM AT compatible personal computers that attach to the POCC communications network as workstations which receive a composite of operator display screens. The configuration of these AT compatible personal computers is as follows:

- Kandl AT running at 8 mHz clock speed,
- Dolen DC-4 (to be upgraded to DC-8, when available) Video Board supporting RS-170A,
- 30 MByte hard disk,
- 1.2 MByte and 360 KByte floppy disk drives,
- 1.5 MByte AST Advantage Memory Board,
- 4 RS-232 communications ports,
- 1 parallel port, and
- Color display compatible of 640 x 480 pixels in 8 colors.

This configuration will support the Intelligent Systems Corporation (ISC) video format for compatibility with MSOCC and the POCC display systems.

The Kandl AT is configured with 512 KBytes of internal RAM with an additional 1.5 MBytes of Extended Memory RAM available on the AST advantage card. Although there is a 640 KByte addressable memory limit of PC-DOS 3.1, the 1.5 MByte AST Advantage Memory Card can be configured as a RAM disk and used for storing data buffers, or as a repository for other executable code that could be swapped into the DOS memory area on demand.

The Kandl AT will contain a 30 MByte hard (fixed) disk for storage use. The operating system and other support programs are predicted to occupy approximately 10-15 MBytes of disk storage. This leaves ample storage for the entire CLEAR application and supporting files which are predicted to be less than one MByte of disk storage. The amount of memory required by the CLEAR Event Log is not included in this figure as it is presently not known how much storage will be required.

EXPERT SYSTEM CONCEPT

The CLEAR expert system is to assist the analyst in the MOR in operating the COBE spacecraft communication links with the TDRS. The following functions are to be performed by the system:

- monitor the spacecraft communication data,
- isolate suspected faults for analysis,
- diagnose actual faults,
- determine the set of alternative actions,
- rank and display the possible responses for the analyst,
- explain the diagnosis, the selection of alternatives and the ranking of possible responses, and
- activate the operator selected response (future enhancement not part of initial prototype).

Two very significant requirements are the "realtime" response required of the expert system and the mandatory requirement that the effect of the CLEAR on the operational system be either nil or minimal. These two requirements considered together have generated the concept of the CLEAR expert system prototype attached to the operational system as if it were an operator's workstation display. This approach, rather than that an embedded or inline system, is expected to reduce the effect of the prototype expert system on the operational system to the minimum and to meet the response requirement.

TOOL SELECTION

The realtime response required of the CLEAR system translates into a performance requirement for the expert system. The data driven and diagnostic nature of the expert system place interface and inference logic requirements on the tool selected to build the application. Further selection criteria come from the hardware and software compatibility requirements. The CLEAR must run on the POCC workstation which is an IBM AT compatible personal computer and must use operating system level software written in C and compiled under Microsoft® C Version 4.0.

A number of secondary (desireable rather than mandatory) requirements are also used in the selection including cost,

number of tool users, length of tool usage, stability of supplier, development environment and availability of source code. The secondary selection criteria are used to rank the expert system building tools that satisfy the mandatory requirements.

Although several commercially available expert system building tools meet the mandatory requirements based upon available information including independent benchmark tests, product reviews and reported user experience in using these products, none is ranked higher than a NASA expert system building tool, CLIPS.

CLIPS, a tool for the development of expert systems, was created by the Artificial Intelligence Section of the Mission Planning and Analysis Division at NASA/Johnson Space Center. CLIPS provides an inference engine and language syntax which provides the framework for the construction of rule-based systems.

CLIPS was entirely developed in C for performance and portability. The key features of CLIPS are:

- Forward Chaining Rules

- Powerful Rule Syntax: CLIPS allows free form patterns, single and multi-field variable bindings across patterns, user defined predicate functions on the LHS of a rule, and other powerful features.

- Portability: CLIPS has been installed on over half a dozen machines with little or no code changes.

- High Performance: CLIPS' performance on minicomputers (VAX, SUN) is comparable to the performance of high powered expert system tools in those environments. On microcomputers, CLIPS outperforms most other microcomputer based tools.

- Embeddable: CLIPS systems may be embedded within other C programs and called as a subroutine.

- Interactive Development: CLIPS provides an interactive, text oriented development environment, including debugging aids.

- Completely Integrated With C: Users may define and call their own functions from within CLIPS.

- Extensible: CLIPS may be easily extended to add new capabilities.

- Source Code: CLIPS comes with all source code and can be modified or tailored to meet a specific users' needs.

- Fully Documented: CLIPS comes with a full reference manual complete with numerous examples of CLIPS syntax. Examples are also given on how to create user defined functions and CLIPS extension. A User's Guide to introduce expert system programming with CLIPS is also available.

CLIPS is available through:

Computer Software Management and
Information Center (COSMIC)
NASA Software Dissemination Center
University of Georgia
Athens, Georgia.

DEVELOPMENT APPROACH

The CLEAR system is being implemented in three phases. The first phase, which was completed February 28, 1987, was the rapid prototyping phase. The prototype was developed on a Symbolics 3640 Lisp Machine using the Automated Reasoning Tool (ART). The first phase demonstrated the expert system technology and an understanding of the problem domain using an advanced development environment. The products of this phase that transfer to the next are the user

interface and the knowledge base.

The second phase of implementation is the operational prototyping phase. This operational prototype is being developed on the Kandl AT using CLIPS and the Microsoft® C 4.0 compiler. In this phase, the CLEAR team will evaluate hardware and software performance, and locate problems using the actual operational environment driven by simulated operational data. In addition to the early determination and resolution of problems, the knowledge base is being enhanced and the entire system will transfer to the final phase.

The third and final phase of the CLEAR system implementation is the installation in the COBE MOR of the hardware and software developed in the second phase. Deferred components and enhancements stemming from the second phase will be developed at this time. This operational prototype will be integrated, tested and available to assist the MOR operator/ analyst during COBE operations.

During the first two phases, CLEAR receives simulated data from a locally written Data Simulator. The simulator transmits data resembling the data that CLEAR will receive from the MSOCC Applications Processor after installation and integration in the third phase. The CLEAR team developed the Data Simulator to allow testing and debugging of the expert system without having to wait for simulated test data in the last phase.

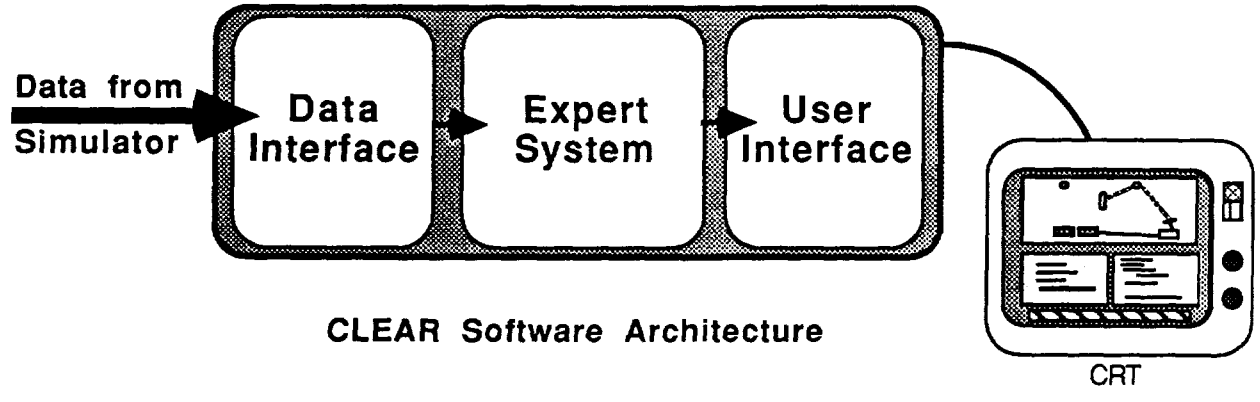
The Data Simulator is a software program that resides on a VAX 8600. This design prevents the simulator from interfering with the processing time of the computer on which CLEAR is running. The design also allows the simulator to be used to test CLEAR on both the Symbolics 3640 in the first phase and the Kandl AT in the second without rewriting or transporting the software program.

In the third phase, CLEAR will have a physical interface with the MSOCC Applications Processor (AP).

SYSTEM ARCHITECTURE

The CLEAR system software architecture consists of the *Expert System* and two *Interface Subsystems* (Figure 1).

The *Expert System* is a forward-chaining, rule-based system. It is implemented using the C Language Integrated Production System (CLIPS), an expert system development tool developed by Johnson Space Center. CLIPS was chosen because it is forward-chaining, portable, and supports integration of external functions written in the



CLEAR Software Architecture

Figure 1

programming language 'C' .

The *User Interface* subsystem interfaces the user's CRT display to the Expert System. This subsystem generates all of the text and graphics to be displayed to the user on the CRT. Independently developed Video Interface Routines are utilized by this subsystem to produce the screen display.

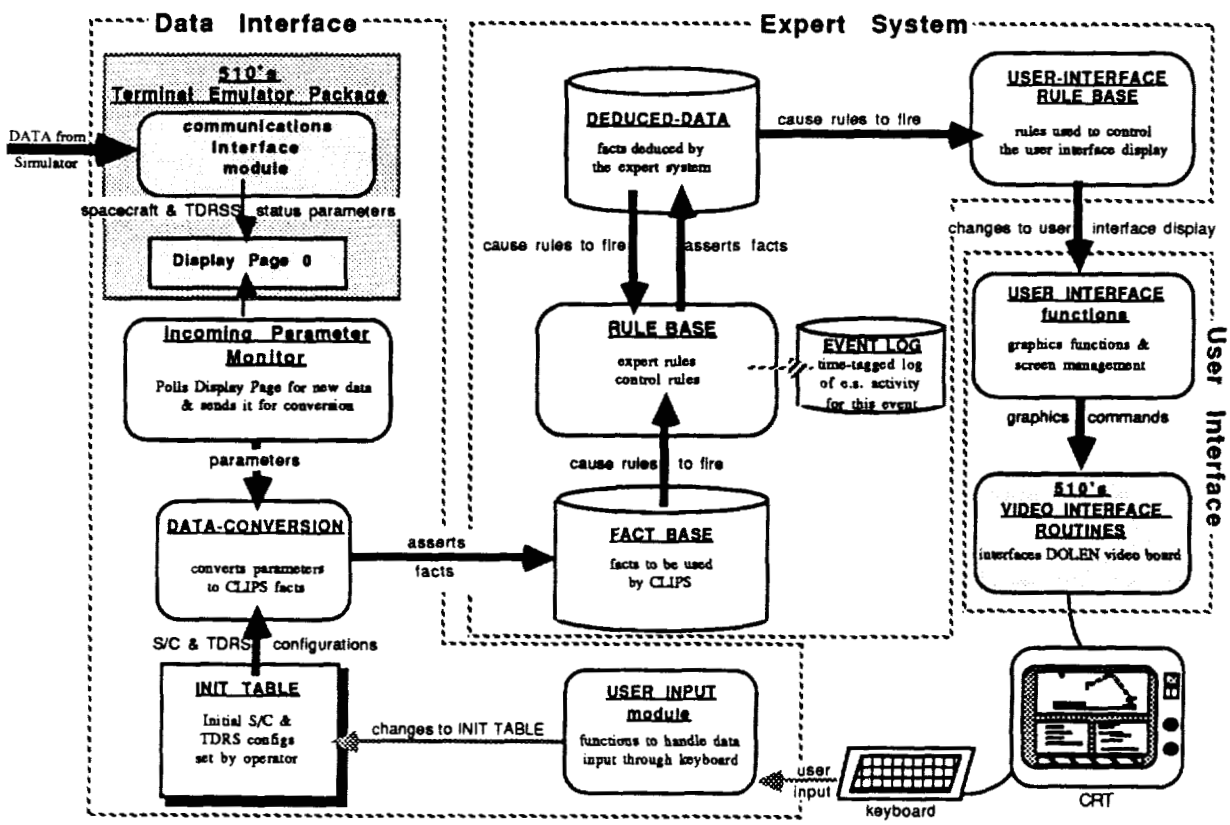
The *Data Interface* subsystem interfaces the Expert System to the Applications Processor in the Multi-Satellite Operations Control Center (MSOCC). This subsystem buffers the ODM and spacecraft status parameters received from the AP utilizing the workstation's communications interface software which is being developed independently. The buffered data is then converted to the format required by the Expert System and passed to it.

These three primary subsystems of the CLEAR system can be further broken down into their functional modules (Figure 2).

Expert System

The Expert System consists of fact bases, a rule bases, an *Inference Engine* and an *Event Log.*

Data enters the Expert System as CLIPS facts asserted by the *Data-Conversion* routine. Each fact represents a piece of information which has been asserted into the *Fact Base* or *Deduced-Data* base. In CLIPS, the Fact Base is properly called a "Fact-List" and the existence or non-



CLEAR Functional Design (phase 2)

Figure 2

existence of facts in this list causes rules in the *Rule-Base* to fire. The actions of rules can cause facts to be asserted in or retracted from both the Deduced-Data base and the Fact Base.

The *Event Log* is a log of all Expert System activity for post-event analysis. The system time tags all identified faults and records the inferences, diagnoses, and recommendations offered to the analyst. The system provides non-realtime utilities to print a formatted copy of the log, to trace and analyze the activity of the inference during the event, and to extract statistics for evaluation of system performance.

The *Deduced-Data* base is a list of facts that are deduced by the Expert System, such as the status of the links and control information. These facts cause rules in the User-Interface Rule Base to fire which, in turn, sends function calls to the User Interface subsystem.

User Interface

The *User Interface Functions* manage the screen display. These functions utilize the *Video Interface Routines* that are being independently developed to drive the Dolen DC-4 (and, subsequently, the DC-8) video board.

Data Interface

The ODM and spacecraft status parameters coming from the AP will be formatted by the *Terminal Emulator* package for viewing upon the workstations' CRT by the FOA. The Terminal Emulator package functionally processes the incoming data as follows.

The parameters enter the POCC workstation through the communications port and are stored in a circular character buffer. Every 30 milliseconds (30 μ-sec), a process checks this buffer for newly arrived data. When a parameter arrives, this process first decodes its value, attributes and screen coordinates, and then stores them in a video buffer named *Display Page 0* (zero).

The *Incoming Parameter Monitor* will poll the Display Page 0 for the new parameters and will send them to *Data-Conversion* when located. Data-Conversion converts these parameters to the corresponding CLIPS facts and asserts them into the Expert System.

A second source of input data is the Initialization Table. This table contains COBE spacecraft and TDRS configuration parameter values. When first set or when modified, these values are sent to Data-Conversion to be converted to the appropriate CLIPS format. If parameters in the Initialization Table are modified during an event, CLEAR can be reset and restarted using these modified values.

The Expert System, written in CLIPS, was the first subsystem to be coded. The coding process was straightforward due to the similarities between CLIPS and ART (in which the phase one prototype was written). The User Interface was the second subsystem to be developed. A highly functional user interface assisted in developing knowledge base because it was only through the CRT display that the capabilities of the expert system could be demonstrated to the "expert" for approval or refinement. The Data Interface was the final subsystem to be coded. Both the Data Interface and the User Interface subsystems were coded in Microsoft® C 4.0.

OPERATION

As a data-driven expert system, CLEAR receives ODM and status data from the AP in realtime, monitors the data, advises

the FOT of problems and recommends possible solutions for each problem isolated. To facilitate greater acceptance and ease of use, user input is minimal.

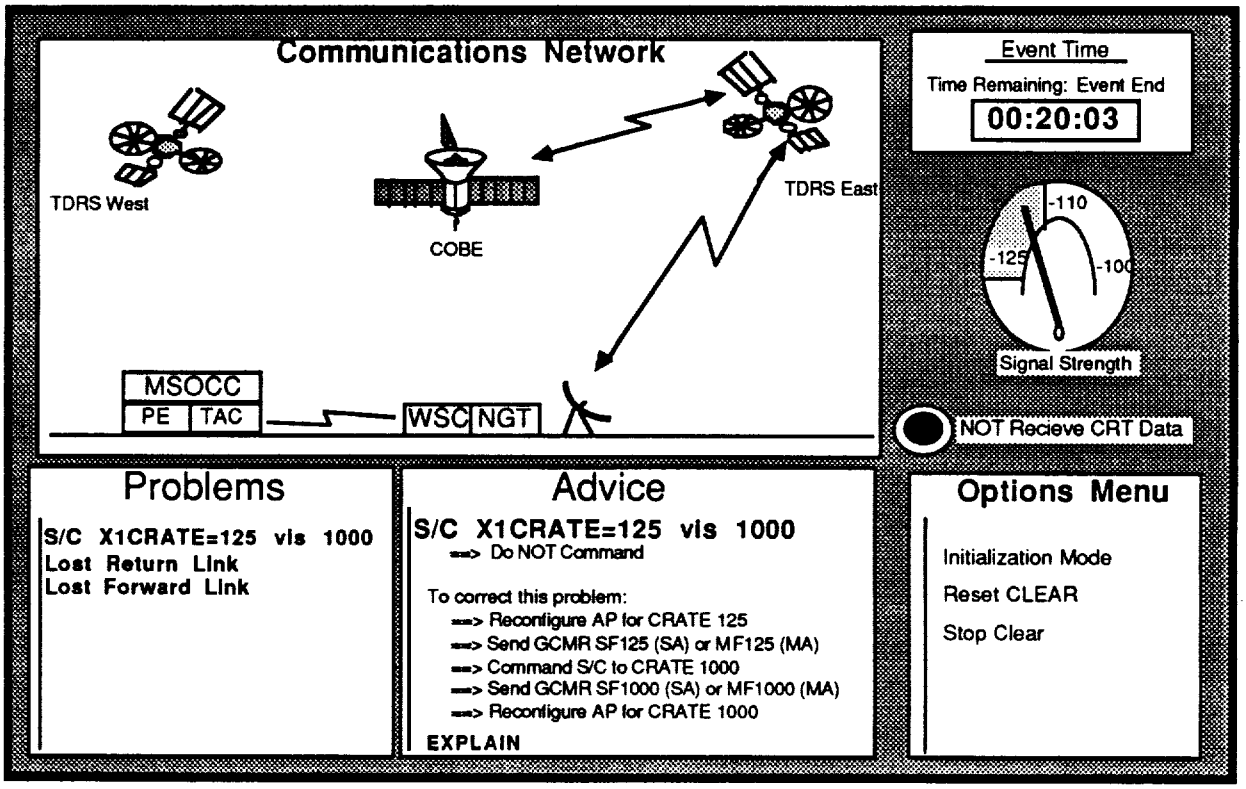
The only user input required by CLEAR is the pre-event initialization of parameters in the Initialization Table. Default settings for each spacecraft and ground system parameter are provided along with an override via keyboard entry.

During a pass, the parameter values of the Initialization Table can be modified by the Flight Operations Analyst (FOA) if the expert system detects and notifies him/her of anomalies between these parameters and the actual COBE or TDRS configuration. The FOA may then reset and restart the expert system using the corrected parameters.

If, on the other hand, the user notices an incorrect value in the Initialization Table before the expert system isolates it, a correction can be made without affecting the Data Interface software or the functioning of the expert system.

CLEAR outputs to the CRT of the workstation. Figure 3 shows the screen display of the user interface of the CLEAR first phase prototype which was developed on the Symbolics 3640. Due to the differences in graphics capabilities between the Symbolics and the Kandl AT, the actual screen display developed in the second phase may be different; however, the CLEAR team will strive to develop a screen display as similar as the graphic routines will efficiently permit.

The display has a COBE-POCC network graphic, "Problem" and "Advice" message



CLEAR User Interface Design
Figure 3

areas, and various graphic indicators which are used for continual display of the time and other important parameters. The "Problem" and "Advice" message areas remain blank until the need for a message is determined by the Expert System, at which time the problem and the appropriate advice are displayed. The "Problem" area displays all the problems isolated, prioritized by the most likely initial cause. The "Advice" area provides recommendations on how to correct the most critical problem of the "Problem" area. In the case of multiple advice options, the first one listed is the best option, followed by other advice in descending order of probable effectiveness.

The COBE-POCC network graphic shown in Figure 3 consists of two TDRS spacecraft, the COBE spacecraft, and the NGT/WSGT and MSOCC boxes. When solid lock on COBE is achieved, there are two green lines (each line denotes both the forward and return links); one from COBE to the appropriate TDRS, and one from TDRS to WSGT. If data indicates that there is a problem with either of these links, the troubled link will turn red and flash while the other healthy link remains green.

FUTURE PLANS

The second phase, implementation of an operational prototype on the EAW, will be finished in the Fall of 1987. The third phase will begin at that time and will include the following tasks:

- implementation of deferred modules,
- addition of enhancements identified in the second phase,
- generation of system documentation,
- generation of training manuals,
- delivery and installation of the CLEAR prototype in the MOR,
- system integration, and
- system test and acceptance by the user.

One additional task, the system performance evaluation, must be deferred until COBE is launched and a baseline of operational experience has been obtained. This task is to evaluate both the efficiency of the system and the effectiveness of CLEAR in assisting the operator in the MOR.

ACKNOWLEDGEMENTS

The authors wish to recognize the assistance of the rest of the CLEAR team: Troy Ames, user interface design, Robert Dominy, first phase prototype, Edward Lewis, tool selection, and David Bludworth, CLEAR expert. We also wish to thank Dorothy Perkins and Robert Dutilly for their assistance in establishing the CLEAR project.

REFERENCES

1. Bludworth, David, *Cosmic Background Explorer (COBE) Expert System Requirements Document* , CLEAR Project, June 1986.

2. Hull, Larry G., editor, *Communications Link Expert Assistance Resource Project Requirement Specifications* , CLEAR Project, September 1986.

3. Dominy, Robert, *Symbolics Prototype* , CLEAR Project, March 1987.

4. Hughes, Peter M., *COBE Expert System CDR Document* , CLEAR Project, March 1987.

5. Culbert, Chris, *CLIPS Reference Manual Version 3.0* , JSC Mission Planning and Analysis Division's Artificial Intelligence Section, July 1986.

The Load Shedding Advisor: An Example of a Crisis-Response Expert System

Terry B. Bollinger
Software Productivity Consortium
1880 Campus Commons Drive, North
Reston, Virginia 22091

Eric Lightner / John Laverty
Bendix Field Engineering Corporation
10210 Greenbelt Road, Suite 200
Greenbelt, Maryland 20706

Edward Ambrose
Goddard Space Flight Center, Code 534.1
Greenbelt, Maryland 20770

Keywords: *Expert systems, Prolog, load shedding, entity-relationship model, relational databases, human factors, hybrid methods.*

Abstract: *The paper describes a Prolog-based prototype expert system that was implemented by the Network Operations Branch of the NASA Goddard Space Flight Center. The purpose of the prototype was to test whether a small, inexpensive computer system could be used to host a load shedding "advisor," a system which would monitor major physical environment parameters in a computer facility, then recommend appropriate operator responses whenever a serious condition was detected. The resulting prototype performed significantly better than was originally anticipated, due primarily to efficiency gains achieved by replacing a purely rule-based design methodology with a hybrid approach that combined procedural, entity-relationship, and rule-based methods.*

1. Introduction

Successfully operating a large computer facility is a task that encompasses far more than simply knowing how to run computers. A computer facility is complex, integrated combination of physical, environmental, and computational systems that must work in unison to achieve the overall purpose of the facility; for example, the failure of a small valve that supplies chilled water to an air conditioning unit can cause a computer system to grind to a halt just as surely as the loss of a critical system file. When the interlinked support systems of a facility work smoothly, it is quite easy to forget about the safety net that they provide; however, when one of those

support systems suddenly fails or is seriously damaged, a good understanding of its relationship to data processing and communications equipment can suddenly become critical.

After a support system failure has been recognized, the decisions made during the (often short) span of time available for responding to the problem can make the difference in whether critical processing must be abandoned, and in certain cases may determine whether facility equipment is physically damaged. The problem of how to respond to a support system failures is aggravated when a facility is part of a larger real-time communications network, since a loss of key functions in such a facility can have a direct impact on sites throughout the network.

At the NASA Goddard Space Flight Center, the problem of how to minimize the impact of support system outages is a very real operational issue. Goddard is the home of the Network Control Center, or NCC, which is the central control facility for the Tracking and Data Relay Satellite System (TDRSS) Network. The TDRSS Network combines a ground based communications network with a geosyncronous relay satellite, and it is used to provide communications support to a variety of satellites and spacecraft, including the Space Shuttle. Due to the central role of the NCC in the TDRSS Network, the failure of one of its support systems can have an impact that goes far beyond the NCC, affecting communication nodes and TDRSS customers at various remote sites. Such outages can in certain instances result in the loss of irreplaceable scientific data; in the case of the Space Shuttle, such an outage could make it necessary to fall back to the Ground Network, an older ground-based communications network with less coverage.

2. The Load Shedding Study

In August 1985, the Network Operations Branch (Code 534.1) of the Goddard Space Flight Center began a study to determine whether expert system methods could be used to assist NCC operators in responding to failures in NCC support systems. The specific area selected for investigation was load shedding, which is defined for the NCC facility as the selective reconfiguration and shutdown of equipment during power, temperature, or humidity crises. Specific goals of the investigation were:

a) *To determine the applicability of expert system methods to the load shedding problem.* Due primarily to the need for real-time responses, expert system methods were not automatically assumed to be the best approach to the load shedding problem. Although a few examples of real-time expert systems (such as Navex[1]) were known when the study began, most examples of expert systems were for stand-alone use only.

b) *To provide a mechanism for formally capturing load shedding expertise.* Even if the load shedding application did not prove to be a good candidate for expert system methods, it was felt that the formal capture of operator expertise would be a valuable result in itself, since it could be used to formulate better manual procedures for load shedding.

c) *To determine processing and storage needs of an operational expert system.* By developing a prototype of the expert system, it was hoped that firmer estimates could be made of the processing and storage resources needed for an operational load shedding expert system.

A key component of the Code 534.1 strategy was to build an actual small-scale prototype of a Load Shedding Advisor, using an off-the-shelf expert system shell on a PC-class computer system. Although it was recognized that a PC-class system was unlikely to be powerful enough to host a fully functional real-time load shedding system, the PC approach had the important advantage of providing a convenient and readily available host system for developing the major features of the knowledge base.

3. Results of Prototype Development

The load shedding study was completed in October 1986, and the results were encouraging. In particular, the final version of the prototype performed significantly better than anticipated, leaving open the possiblity that a PC-class computer could be used to host an operational load shedding expert system. Gains in both efficiency and maintainability were achieved through the use of a "hybrid" design approach that was developed as the prototyping effort progressed. This hybrid methodology, which is described in more detail later in this paper, replaced a purely rule-based design with a combination of procedural, entity-relationship, and rule-based methods.

Three distinct prototypes were constructed for the load shedding study. Each of these prototypes used a different shell or shell version, and each one concentrated on a different aspect of the load shedding problem. All of the prototypes were implemented on an IBM PC XT with 640K bytes of memory, a 30 megabyte hard disk, two half-height floppy disk drives, an EGA graphics board, a mouse, and an MS-DOS operating system. The number of people working on the project varied from one to three, with one person always assigned full time. The prototyping activity lasted fourteen months.

The first two prototypes were aimed at determining the actual rule sets for an NCC load shedding system, and were directly based on expertise gained by interviews with NCC facilities and operations personnel. The third prototype was used to investigate structural and human factors issues, and was designed for use as a demonstration system. To avoid having sensitive data in the demonstration prototype, its knowledge base was constructed around a hypothetical facility that demonstrated features found in most large computer facilities, rather than being based on the NCC. Descriptions of the three Load Shedding Advisor (LSA) prototypes are given below.

4. LSA-1: Interactive Diagnosis of Load Shedding Problems

Design of the LSA-1 Prototype. The first prototype, LSA-1, was a classic Mycin-style expert system that used Newell/Simon production rules[2] to represent load shedding expertise. The rules were implemented with Version 1.3 of Teknowledge's "M.1" expert system shell, a rule based, backwards-chaining shell that is similar in syntax to the AI language Prolog[3]. Like most

Mycin-style expert systems, the operator interface for LSA-1 was "conversational;" the expert system acquired facts by engaging in a selective question-and-answer session with an operator.

The LSA-1 Knowledge Base. A substantial body of rules was collected and formalized for the LSA-1 effort, but for reasons described below, only a small subset (about 30) of these rules were actually coded into the knowledge base. The implemented subset covered diagnosis and response to power distribution problems for a hypothetical facility that included several load centers (power conversion transformers), two commercial power feeders, and two backup power feeders.

Evaluation of Results. Within the limits of the data available to it, the LSA-1 system performed reasonably well. Response times were acceptable, ranging from a few tenths of a second to one or two seconds, depending on how much text was displayed and how much inferencing was required. Certain aspects of the M.1 Version 1.3 user interface, such as the requirement that all entries be terminated with a period, were cumbersome, but the system converged rapidly to conclusions and needed relatively few entries from the operator.

Although the production rule model of LSA-1 provided a good mechanism for collecting and formalizing load shedding expertise, it did not adequately satisfy the primary goal of determining whether a real-time, on-line load shedding advisor was possible. Factors which made LSA-1 inadequate for accurately assessing the load shedding expert system problem were:

a) *Incomplete coverage of the problem domain.* Since the LSA-1 system was built on the assumption that all decision data would be obtained from an human operator, its coverage was necessarily limited to faults whose effects were visible to the operators. Such an approach suffers from the dual problems of poor fault coverage (since only a small subset of the potential range of fault indicators would be used) and poor resolution (since there was not always enough data to distinguish between distinct faults).

b) *Reliance on operators for time critical status data.* Another problem with LSA-1 dependence on the operator interface for data was the time critical nature of many load shedding problems. In a load shedding emergency, the operator may need to respond in less than a minute; in such a case, it is very unlikely that he will want to spend that time getting the expert system "up to date" on what has happened.

The Domain Status Acquisition Problem. The most significant problem with the LSA-1 prototype was its reliance on a question-and-answer dialog to obtain the data it needed to make load shedding recommendations. While this type of dialogue is adequate for situations where the problem is stable over the time of the dialogue (most patients don't die while Mycin is asking questions), it can be highly inappropriate in a crisis situation where a fraction of a minute may make the difference between success and failure. The problem is aggravated by the fact that good load shedding recommedations require more than a simple identification of the problem; unless the advice can be "customized" to the current status of a facility, the system can only give generic advice on how to deal with a problem.

In a crisis situation, generic advice is a poor second to a list of specific instructions. The difference between the two may be seen in the example of telling an operator to "shut down non-critical disk drives," as opposed to giving him a list of exactly which drives should be shut down. Such customization is dependent primarily on the availability of good status data, rather than on the inferencing power of an expert system. Without such customization, a load shedding expert system would in effect become an on-line documentation system, where operators would "look up" standard procedures by entering a short list of index conditions. While such a system could be valuable as an automated replacement to paper-based operating manuals, it would fall substantially short of the full potential of a load shedding expert system.

In the sections below, the set of data that describes the current status of an expert systems's problem domain (in this case a computer facility) is referred as *domain status data*. The problem of how to acquire such data effectively is referred to as the *domain status acquisition problem*.

5. LSA-2: Simulated Domain Status Monitoring

Purpose of the LSA-2 Prototype. Since LSA-1 did not adequately address some of the broader issues of how to implement a load shedding expert system, the LSA-1 system was abandoned in favor of a new approach in which the domain status acquisition problem would be explicitly addressed. To test the effectiveness of integrating domain status data into the expert system, a new prototype, the LSA-2 system, was implemented.

Design of the LSA-2 Prototype. LSA-2 was implemented in M.1 Version 2.0, which provided a number of significant enhancements over Version 1.3. Version 2.0 was written in C rather than Prolog, and as a result it was about five times faster. Screen display commands, which were one of the weakest points of Version 1.3, were also significantly improved, although they still fell short of the capabilities provided by many conventional PC-based programming languages.

The major thrust of the LSA-2 effort was to simulate entry of domain status data through the the use of simple menu-style operator query screens. The operator could selectively modify the recorded status, which would then be recorded as facts in the knowledge base. Upon completion of such entries, the system would evaluate the status values for potential problems, and would automatically prompt the operator if any were found; thus, unlike LSA-1, the new system was able to initiate its own diagnostic sequences without waiting for an explicit operator query.

Like LSA-1, the LSA-2 system was implemented using only production rules. Features such as the menu displays could also have been accomplished by writing a C program that interfaced with the expert system, but it was decided to instead try using the enhanced I/O features of M.1 Version 2.0 to create a more-or-less conventional menu interface.

The LSA-2 Knowledge Base. The decision to implement conventional menu-style interfaces with production rules instead of external code turned out to be rather a disaster, particularly from the perspectives of clarity and

maintainability. Since M.1 Version 2.0 permitted a maximum of 16 variables in any one rule, displays that showed more than 16 variables at once (of which there were several) had to be "coded" by using remarkably opaque trees of display rules. The ability to iteratively modify a parameter, a must for those of us who don't always get it right the first time, could only be implemented by using M.1 metacommands to selectively reset (clear) facts from the knowledge base, an approach which again led to opaque "code."

By adding the requirement that the knowledge base handle more complex data about the status of the facility, it also became necessary for LSA-2 to perform calculations on moderately large sets of data; for example, the total electical load of a facility could only be obtained by adding the individual loads of all active equipment items. Although M.1 was capable of performing arithmetic calculations fairly quickly, the design used in LSA-2 used inferencing as its data access mechanism, an approach which led to very slow evaluations. For example, one simple summation of a few dozen real values took over 60 seconds to perform, a figure that clearly leaves room for optimization.

Evaluation of Results. Unlike the LSA-1 system, which performed reasonably well withing its defined limits, the LSA-2 experiment was obviously no where near its optimal level of performance or structure. Its main value was conceptual; by providing a first-draft attempt to organize and use explicit domain status data in a knowledge base, LSA-2 suggested new ways for organizing such data in a more coherent fashion. After completion of a slow (but functional) LSA-2 system, the prototyping effort shifted its focus to a new tool and a new representation of domain status data.

6. LSA-3: Entity-Relationship Problem Modeling

Switching Over to Turbo Prolog. As described above, the LSA-2 prototype had run into difficulities in its use of rule-based methods for menu displays and numeric calculations. The best solution to this problem appeared to be convert I/O and numeric functions to C, and to use the M.1 shell only for inferential problem solving. Another possibility was translation of the M.1 knowledge base into Prolog, a language which shares many features with M.1, but which contains a fuller range of low-level I/O and numeric functions. Unforunately, most of the PC-based Prolog systems that were available at that time were slow and rather limited, and they often lacked the standard features found in large-machine versions of Prolog.

The situation changed when Borland released Turbo Prolog Version 1.0. Turbo Prolog is actually a subset of full Prolog, since it omits an important feature known as metaprogramming, but the product has a powerful set of high-level and low-level I/O routines, and it is very fast. After a short period of testing, a decision was made to translate the major features of the LSA-2 prototype into Turbo Prolog.

Penalties involved in switching from M.1 to Turbo Prolog included the loss of a rich set of M.1 commands and metacommands, a switch to a less English-like syntax, and loss of the built-in conversational interpreter. These losses were offset by that fact that the LSA-3 design would be built almost entirely around menu-based interfaces, and would rarely need a conversational interface.

Design of the LSA-3 Prototype. As a result of evaluations of the LSA-2 prototype, it was decided that the LSA-3 prototype should explicitly partition the load shedding problem into three components, each of which would use a different conceptual model. The components of this hybrid approach were:

a) *Procedural Programming.* The experiences gained in the LSA-2 model indicated that for many of the current expert system shells, conventional coding may be the best way to implement support functions such as I/O and mathematical calculations. The general rule for deciding whether to use procedural code is that if a function attempts to use inferencing to perform simple, non-heuristic accesses to data, I/O devices, or other rules, then it probably should be implemented with some form of procedural programming. Inferencing is a powerful look-up mechanism, but it is also very expensive in its use of computer time; careless use of inferencing in an expert system can very quickly lead to serious performance problems.

   In Turbo Prolog, "conventional" programming was simulated by selectively using the cut operator to constrain predicates into non-backtracking behavior. Predicates in this form could be used as close analogs of conventional subroutines and functions, and various Turbo Prolog compiler optimizations provided excellent speed and memory performance for constrained predicates.

b) *Entity-Relationship Modeling.* Entity-relationship (E-R) modeling[4,5] is an idea that has become popular in recent years in the database management community. The E-R model of a problem is actually a form of the well-known relational database model[6], differing only in that "relationships" between tables of information ("entities") are explicitly named and defined, rather than being implicit as they are in the relational model.

   The importance of E-R models to expert systems is that they can be used to separate problem modeling from problem expertise. In the case of the load shedding problem, E-R models can be used to represent sets of equipment and their properties, while rules for handling support system faults can be be generalized to address entire classes of equipment, rather than individual items. Since most expert system shells are built around relational databases, simple forms of the E-R model can be directly implemented in products such as M.1. Prolog, with its inherently relational structure, is a particularly good language for implementing E-R problem models.

c) *Rule-Based Knowledge.* In the hybrid approach, production rules are reserved for their classical application of modeling human expertise about specific, well-defined problem domains. However, unlike most rule-based expert systems, the rules of a hybrid expert system should make their assertions primarily in terms of an underlying entity-relationship model of the problem domain, rather than in terms of the external world of the expert system user. Changes to status data in an E-R database can be hidden from the expertise rule set, so that these rules can view the E-R database as if it were a direct mapping from the real-world problem domain. The major advantage of having rules address

the E-R model instead of the external world is generality; rules stated in terms of formally defined E-R facts will tend to be more powerful than rules that refer directly to less formalized structure of the real world.

The LSA-3 Knowledge Base. Using the hybrid design paradigm described above, the LSA-3 knowledge base was divided into two parts: an E-R component that describes the hypothetical facility with relational tables, and a set of production rules for describing expertise in solving load shedding problems. Conventional procedures, which were implemented by using constrained Prolog predicates, were used to transparently update the E-R database to the current (simulated) status of the facility.

The LSA-3 User Interface. The LSA-3 user interface strongly emphasized human factors related to the load shedding problem. In particular, the number of operator keystrokes needed to respond to alarms was kept to a minimum, data entry mechanisms were arranged to make invalid entries difficult or impossible, color coding was used to help operators identify out-of-tolerance values at a glance, selective keyboard lockouts were used to prevent invalid data entries, and all screen displays included information on "what to do next." For normal conditions the system the system is passive, requiring no interactions from operators. For serious emergencies, the system goes into an "imperative" mode in which visual and audible alarms are activated. The alarms will remain on until an operator either follows the recommendations of the system, or explicitly overrides the alarms with an explanation of their cause.

Using "instantiation" of general conclusions against the E-R model of active equipment, the prototype is able to convert a general conclusion into a specific list of specific, item-by-item recommendations that are then presented to the operator as a series of imperative menus. These menus are presented in priority order, with the most important sets of equipment given first. Equipment lists are grouped by system to help the menus correspond more closely to the physical components of the facility. If multiple problems are detected, the system responses to those problems are queued in a prioritized order. In such a case, the system will continue to prompt the operator with new alarms until all known problems have been resolved.

Although the primary mode of interaction between the LSA-3 system and an operator is through menus, a special rule interpreter was written in Turbo Prolog for use in cases where reliable domain status data is unavailable. In this situation, it is planned that the load shedding system will fall back to a mode that is similar to to the LSA-1 prototype, in which observable symptoms of faults are used by an operator to access general advice as to what actions to take. The interpreter provides a conversational interface, and is designed to be embedded in a Prolog program, rather than to act as a stand-alone shell. It is backward-chaining, and implements a slightly modified form of Bayesian certainty factors[7] to handle reasoning with uncertain facts.

Finally, the LSA-3 prototype includes a set of routines for creating colored bar gauges, which are used to display temperature and humidity data. The routines are designed for general-purpose use, and can be embedded in any Turbo Prolog program.

LSA-3 Performance. The performance of the LSA-3 prototype was exceptionally good, especially when compared to earlier LSA-2 design that attempted to provide similar features. Evaluations of the E-R status database were completed at a rate of four times a second, and an 80386-based version of the system is expected to run roughly 20 evaluations each second. Both of these speeds are far in excess of anticipated needs. User interfaces were all very prompt in responding, and "smart" display features used in the menus allowed users to see the implications of data entries almost instantly. The only PC resource that was appreciably stessed by the application was memory; in some instances, approximately half of the available 640K bytes of memory was used by the application.

Overall, the performance of the LSA-3 prototype was sufficiently good that the idea of using an 80386-based PC computer to host an on-line Load Shedding Advisor became a real possibility, one which will be investigated in future load shedding work.

7. Advantages of Using Entity-Relationship Models in Expert Systems

The explicit use of an E-R model in the LSA-3 prototype led improvements in both the clarity and performance of the knowledge base, and is a concept that would appear to have considerable generality. Advantages of using E-R models in the construction of knowledge bases include:

a) *Increased Structural Clarity of the Knowledge Base.* In many expert systems, modeling of the problem domain (such as a computer facility or an automobile engine) is intimately mixed with heuristic expertise about the domain (such as which symptoms indicate which classes of faults). The E-R model allows a clear separation of such knowledge, and a corresponding increase in clarity.

b) *Increased Maintainability of the Knowledge Base.* The E-R model of a problem domain will record information that is much more readily accessible than problem solving expertise, and it will record it in the form of easy-to-maintain tables. For example, in a well-structured E-R model, the removal of an equipment item from a computer facility would be shown by simply removing the item from its entity table; the load shedding expertise of the rule base would not need to be changed at all.

c) *Avoidance of Redundant Rule Instantiations.* A common problem in rule-based expert systems is the inadvertant specification of rules which simply repeat a general principle for two or more similar objects. An E-R model helps substantially in eliminating such redundancies by providing a single, coherent terminology for describing classes of similar problem domain objects.

d) *Easier Identification of Generic Rules.* By providing a formal terminology for describing the problem domain, the E-R model can also help an expert notice broader relationships when defining rules.

e) *Increased Clarity and Conciseness of Rule Definitions.* The availability of a formal E-R description of the problem domain also helps make

individual rules more precise and less prone to inadvertant use of synonyms.

f) *Faster KB Evaluations Through Specification of E-R Access Mechanisms.* By defining special forms of relationships which order or otherwise modify the appearance of entities at the rule level, explicit search mechanisms can be imposed on rules, without affecting the way in which rules are stated.

g) *Simplification of Interfaces to Conventional Software.* As described for the LSA-3 model, the E-R database can be updated independently by conventional software, allowing the rule base have rapid access to external data without having to acquire it via inferencing.

h) *Support of Constrained Learning Via E-R Modification.* An valuable property of a well-structured E-R database is that it is very easy to update, provided that the changes to it do not violate the particular E-R model used. This feature could readily be used to create knowledge bases that actively learn about changes in the problem domain, a feature that could be very useful for an application such as the Load Shedding Advisor, where the problem domain (a computer facility) may have frequent changes to items such as equipment lists. A limited learning feature would greatly extend the usability of the expert system in such an environment.

j) *Enhanced Support of "Deep" Reasoning.* Finally, the formal problem domain descriptions of E-R models could be used as "axiom sets" for some forms of self-referential reasoning. The analogy is that the expert system can reason more deeply because it can "see" a formal model of the problem domain, rather than having to relay on implicit problem domain knowledge that is scattered all through the knowledge base.

8. Future Directions

Work on the Load Shedding Advisor has continued into a new phase in which mechanisms for linking a load shedding advisor into NCC status systems are being investigated. With the successful of the LSA-3 prototype in demonstrating that a load shedding system is feasible on even a small computer system, the emphasis has now shifted to the significant problem how to acquire timely, up-to-date data for refreshing the E-R status database.

An equally interesting result of the load shedding effort has been the development of the hyprid approach to developing new expert systems. In particular, the idea of explicitly partitioning knowledge bases into an E-R problem domain model and independent expertise about that model seems to hold considerable promise as a way to extend expert systems into a wider range of real-time applications.

---

[1] Maletz, M.C. 1984. *NAVEX System Architecture and User's Manual.* Los Angeles: Inference Corp.

[2] Barr, A., and Feigenbaum, E. A. 1981. *The Handbook of Artificial Intelligence, Volume 1.* Los Altos, CA: William Kaufmann, Inc.

[3] Clocksin, W.F., and Mellish, C.S. 1984. *Programming in Prolog.* New York: Springer-Verlag.

[4] Foard, R.M. 1985. A Data Manager Using Entity-Relationships. *IBM Tech Journal 3, 10* (Oct 1985).

[5] Chen, P.P. 1976. The Entity-Relationship Model -- Toward a Unified View of Data. *TODS 1, 1* (March 1976), 9-36.

[6] Codd, E.F. 1970. A Relational Model of Data For Large Shared Data Banks. *CACM 13, 6* (June 1970), 377-387.

[7] Barr, A., and Feigenbaum, E. A. 1981. *The Handbook of Artificial Intelligence, Volume II.* Los Altos, CA: William Kaufmann, Inc.

A Local Area Computer Network Expert System Framework

Robert Dominy/Code 522.1
NASA Goddard Space Flight Center
Greenbelt, MD 20771

Abstract

Over the past year I have been developing an expert system called LANES designed to detect and isolate faults in the Goddard-wide Hybrid Local Area Computer Network (LACN). As a result, the need for developing a more generic LACN fault isolation expert system has become apparent. This paper explores an object oriented approach to create a set of generic classes, objects, rules, and methods that would be necessary to meet this need. The object classes provide a convenient mechanism for separating high level information from low level network specific information. This approach yields a framework which can be applied to different network configurations and be easily expanded to meet new needs.

Introduction

A hybrid local area computer network (LACN) connects a variety of computers, ranging from mainframes to micros, throughout several buildings of the Goddard Space Flight Center. The network is based primarily on an Ethernet bus topology (Figure 1) but a star configuration using the Cable TV (CATV) wiring and Applitek equipment allows networks in different buildings to be interconnected.

As this network grows, it becomes more and more difficult to detect and isolate problems. Currently operators do a small amount of network checking manually, but to do a thorough check of the entire network is tedious because of the large number of items involved. As a result, most fault isolation takes place after a user has discovered and reported a problem.

The Local Area Network Expert System (LANES) began in 1985 as an effort to automate the process of fault detection and isolation. Development work is being performed on a Symbolics 3645 LISP machine using an expert

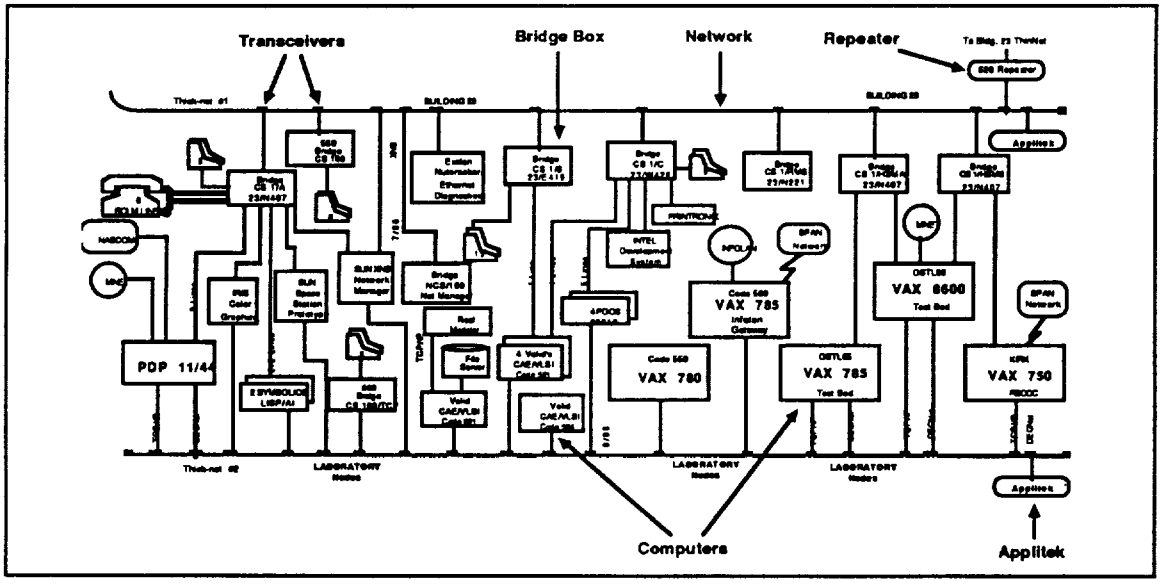


Figure 1 -- Portion of Goddard LACN

system development tool, the Automated Reasoning Tool (ART) from Inference Corp. An operationally useful prototype has been completed and is available for demonstration and use.

The LANES screen is depicted in Figure 2. Each network is depicted graphically and a global map of Goddard Space Flight Center that shows connections between buildings (Figure 3) can be reached by clicking on the window connection icon (the upward arrow) labeled Map. To tell the system to begin checking the entire network a user clicks on the START button by positioning the mouse cursor over the START button and pressing the left mouse button. The button changes to STOP. The expert system heuristic rules then attempt to detect and isolate problems in the network. Results are displayed with text and graphics. For example, in Figure 2 the Bridge box CS1-D cannot be reached (an X is placed in the box) and the Bridge box CS1-B has a high statistical CRC (cycle redundancy check) error count (a question mark is placed in the box).

LANES checks each device on each network, repeatedly cycling through all the networks, until the user clicks on STOP. As each network is checked, the window being viewed will automatically be changed if the current network is on a different screen and the Switch Screens button is set. At any time the user can edit the network design using NetEdit, an interactive graphical network editor. Devices can be added, moved, modified, or removed. The changes are reflected immediately by the expert system and can be used temporarily or saved to a permanent

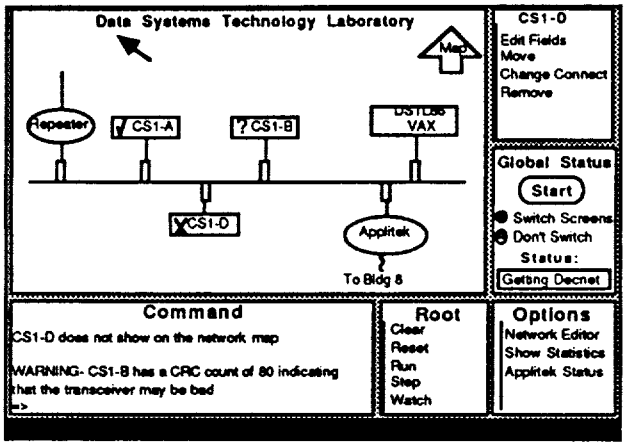


Figure 2 --LANES Screen

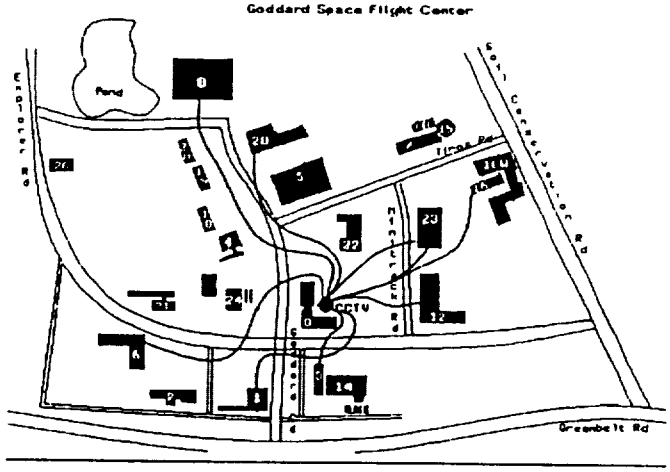


Figure 3 -- Map of Goddard

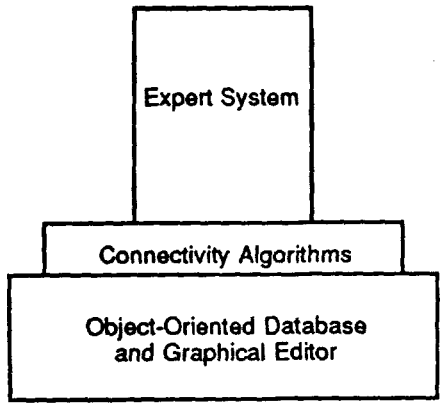


Figure 4 -- Framework Layers

file.

The current version of LANES checks networks located in five different buildings. It gets the up/down status and health statistics of seventeen bridge boxes, gets the up/down status of ten VAX computers, and derives the status of five Applitek devices.

Object-Oriented Framework

Demonstrating LANES has generated interest within Goddard and at other NASA centers. However, because LANES has been designed specifically for the Goddard LACN much of the code that was developed would have little application to other LACNs. In addition, a number of functions and rules within LANES are replicated for similar object types. For example two rules which get the status of an object may only differ by the function which they invoke. This not only duplicates code and effort, but also makes global changes more difficult to make. Improved knowledge abstraction with increased modularity and ease of modification are being added to help solve these problems.

The next step in the evolution of LANES is to build a generalized LACN expert system framework. To build a framework that is reusable, highly modular, and easily modifiable will require the base programming language to provide object-oriented features such as dynamic binding of functions (methods) to data (objects), inheritable object and methods, and object abstraction. A number of object-oriented languages provide all of these features.[1]

Framework Layers

Conceptually the framework is composed of three primary layers (Figure 4), an object-oriented database and graphical editor, a set of connectivity algorithms, and the expert system. This concept allows the lower layers to be fully reusable without the upper layers, but not the reverse since the expert system would need both the database and connectivity layers to function.

The object-oriented database and graphical editor are key elements of the system. The editor provides the main portion of the user interface with which users will create, modify, and add to networks. The database defines the relationship between network components and keeps track of component attributes such as network addresses and protocols.

1. In the computer science field, object-oriented programming is gaining popularity as a software engineering methodology and programming style. The reason for this is that object-oriented programming excels in software reusability, modularity, and ease of modification. Brad Cox discusses the concept of using object-oriented programming to develop reusable software integrated circuits, Software-ICs (Cox, Brad J., Object Oriented Programming: An Evolutionary Approach, Addison-Wesley, 1986). A commercially available product from Apple Computer called MacApp (Doyle, Ken, Wallace, Scott, and Rosenstein, Larry, *MacApp™: An Object-Oriented Application Framework*, Apple Report No. 4, Apple Computer, Inc., September 1986.) provides a highly reusable object oriented framework to implement the standard Macintosh interface, reducing both the time and amount of code needed to build applications.

Connectivity algorithms find paths between components. For fault isolation on the Goddard LACN the main algorithm is a simple depth first search (loops are not allowed in the network) along connections that support the protocols of the machines being tested. Another algorithm would be used to find "backroads" between equipment. An example would be to use a modem to bypass default network components to check the status of a computer. For other LACN configurations (e.g., a network that allowed loops), programmers could use the object-oriented capability of overriding methods to customize the algorithms.

The functions of the expert system layer are fault isolation, component monitor/polling, graphical and text fault notification, and explanations. Fault isolation is provided by both high-level and low-level methods and heuristics. Monitor and polling methods provide the capability to continuously monitor specified components or request an instantaneous diagnosis. Explanations and fault notification are built upon the user interface of the graphical editor.

Object Hierarchy

Each of these layers are built within the object hierarchy shown in Figure 5. The top node, TComponent, is the parent or superclass of all the other network object types. Each subsequent child node or subclass, therefore, inherits methods and fields from TComponent. This, for example, allows defaults to initially be set up in TComponent and later customized in subclasses as required. Some of the classes along with their methods and fields are discussed below. A more detailed design is currently under development.

TComponent has a field, fSubcomponents, which can be used to define TComponent objects as abstract entities such as a network or computer that can be broken down into its fSubcomponents. A polling method is then defined to poll these subcomponents. Another field, fConnectedTo, is a list of components that the TComponent object is connected with.

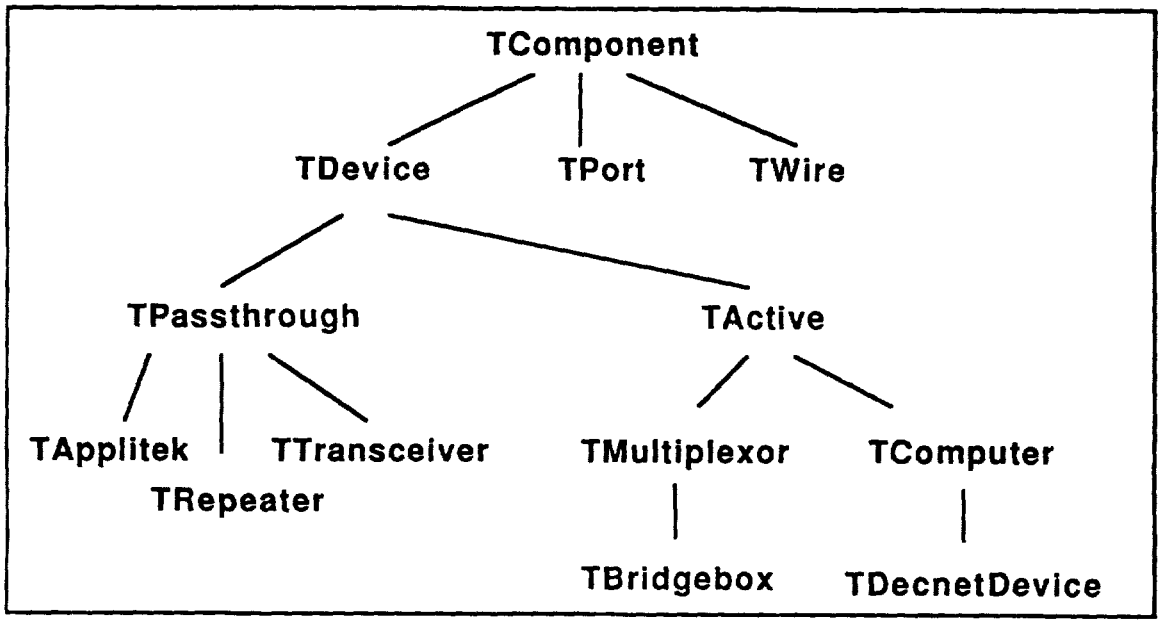


Figure 5 -- Object Hierarchy

TDevice, TPort, and TWire, are major subclasses of TComponent and define high-level information and knowledge. The TPassthroughDevice class is for devices that act as information relays and are essentially transparent to other devices on the network. During fault isolation, the up/down status for devices in this class would have to be based on the ability of TActiveDevice objects to communicate through the TPassthroughDevice object. The TActiveDevice objects can be communicated with directly to determine up/down status. The bottom nodes of the hierarchy (TApplitek, TMultiplexor, etc.) contain the most network specific information and knowledge and

is where most modifications and additions will take place.

Future

Currently a detailed design of the framework is under progress. In addition, a number of potential application areas are being explored. Once one is chosen, the hardware and software to implement the framework will be selected. Currently a variety of object-oriented programming languages (C++, Objective-C, Object Pascal, SmallTalk, LISP Flavors, ART 3.0) which exist on a number of machines are being considered.

When the LANES framework is complete, it should be applicable to a variety of LACN fault isolation problems. It could even be expanded to handle LACN design and performance analysis.

I suspect that fault isolation will not be the only area where frameworks are built. Hopefully we will see a number of frameworks being built to handle other expert system areas such as scheduling, planning, control. What will be key to their success, however, will not just be functionality, but ease of modification and expansion. Object-oriented programming will play a large part in that success.

FIESTA: AN OPERATIONAL DECISION AID FOR SPACE NETWORK FAULT ISOLATION

Dawn Lowe (NASA/GSFC)
Bob Quillin (Stanford Telecommunications, Inc.)
Nadine Matteson (Stanford Telecommunications, Inc.)
Bill Wilkinson (Stanford Telecommunications, Inc.)
Steve Miksell (Stanford Telecommunications, Inc.)

Abstract

The Fault Isolation Expert System for Tracking and Data Relay Satellite System (TDRSS) Applications (FIESTA) is a fault detection and fault diagnosis expert system being developed as a decision aid to support operations in the Network Control Center (NCC) for NASA's Space Network. This paper presents the operational objectives which influenced FIESTA development and provides an overview of the architecture used to achieve these goals. The approach to the knowledge engineering effort and the methodology employed are also presented and illustrated with examples drawn from the FIESTA domain.

1.0 INTRODUCTION

This paper discusses the FAULT ISOLATION EXPERT SYSTEM for TDRSS APPLICATIONS (FIESTA). FIESTA is an expert system which is being developed to provide operator support in the Network Control Center (NCC) at Goddard Space Flight Center in the area of fault isolation.

Section 2 provides an overview of the Space Network and the role of the NCC. Section 3 describes the development environment established for FIESTA project effort.

Section 4 discusses operational concepts which have influenced the development and the displays which support these components. Section 5 presents an outline of the system architecture which supports the processing required to realize the operational concepts. Section 6 then describes the fault isolation methodology which FIESTA employs to provide the desired system capabilities focusing on knowledge engineering aspects of the FIESTA project.

2.0 SPACE NETWORK OVERVIEW

Figure 1 illustrates NASA's Space Network (SN) which combines space and ground segments to provide tracking and data acquisition services for spacecraft in near-Earth orbit (200 to 12000 km). The space segment of the baseline Space Network will consist of two operational geostationary satellites, Tracking and Data Relay Satellite East and West (TDRS-E and TDRS-W), as well as one spare satellite. From their geosynchronous orbit, the two operational satellites will be able to provide services to user spacecraft during 85 to 100 percent of their orbits. The TDR satellites are monitored and controlled from the White Sands Ground Terminal (WSGT), in White Sands, New Mexico. Collocated with WSGT is the NASA Ground Terminal (NGT), which provides the communications interface for the transfer of data from WSGT to the other Space Network elements and users, via the NASA Communications (NASCOM) network.

The Network Control Center (NCC) is the operations control facility for the entire Space Network. It serves as the focal point to network elements and user spacecraft facilities for coordination of all network support, and resolution of problems. The NCC's primary functions include scheduling network resources, equipment configuration direction, and service quality monitoring and assurance.

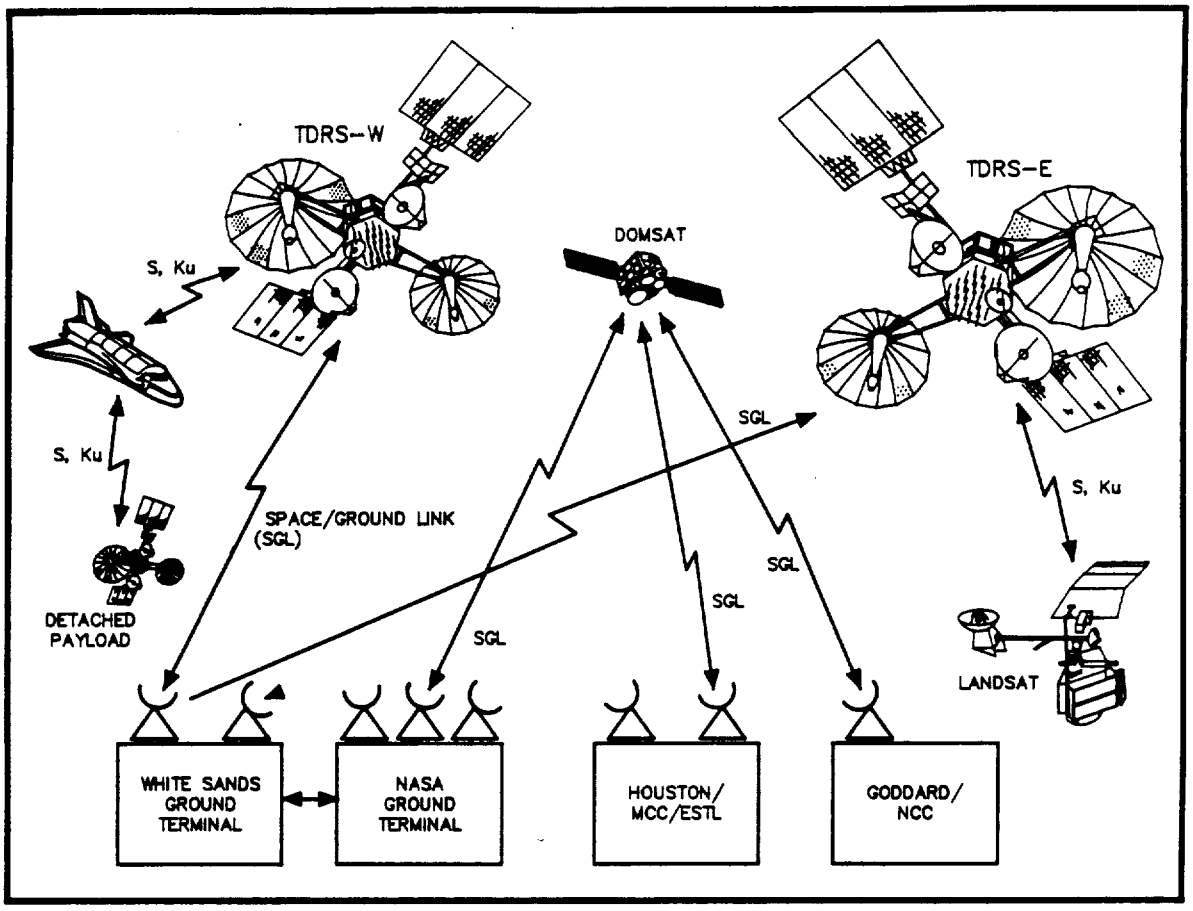


FIGURE 1: NASA SPACE NETWORK OVERVIEW

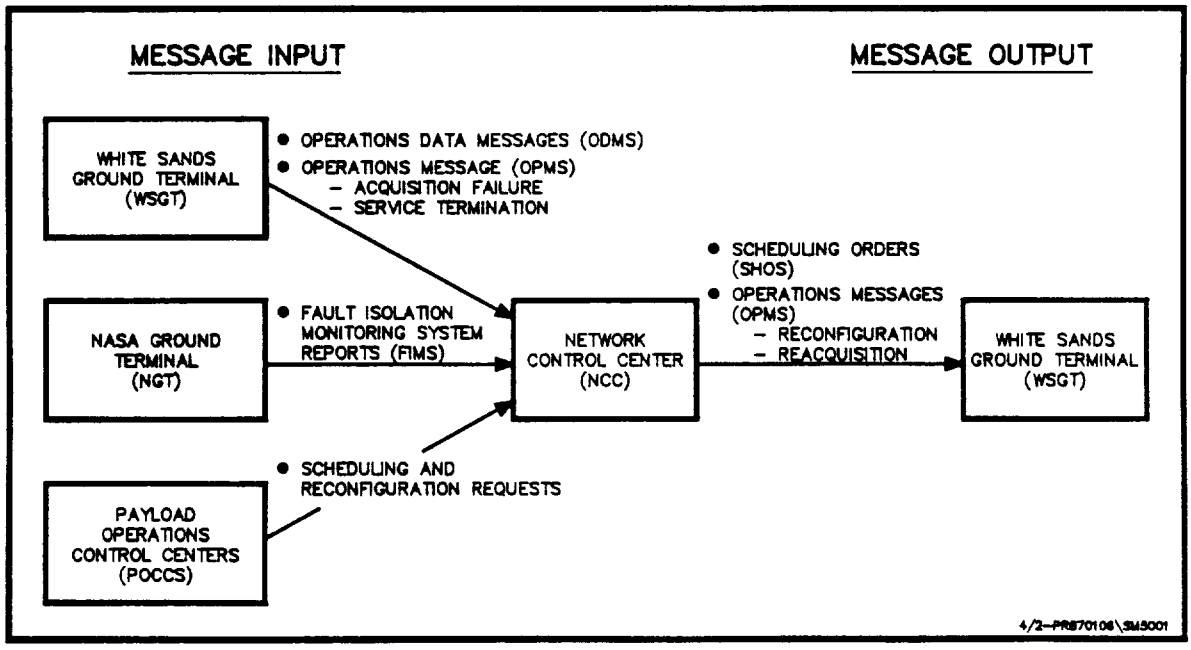


FIGURE 2: FIESTA DATA SOURCES

Role of the Network Control Center (NCC) in Problem Detection/Fault Isolation

Based on requests from user spacecraft control facilities, the NCC schedules "events", consisting of one or more services for a single user. Prior to the start of the event, Schedule Orders (SHOs) are transmitted by NCC to the network elements, indicating the start and stop time of each service, and specific configuration information. During real-time operations, NCC operations personnel monitor the status of network services to detect anomalies so that corrective action can be promptly initiated. The real-time nature of many user spacecraft operations, together with the high bandwidth data flow through the Space Network, combine to increase the critically of NCC's fault detection and isolation responsibilities. It is essential that problems are detected immediately, and service outages are resolved quickly, to minimize data loss and impact to the user mission. This function requires that NCC controllers and analysts continuously monitor network performance indicators, and compare expected versus actual values to detect anomalies. The NCC does not monitor user spacecraft telemetry, command, or tracking data directly. Rather, NCC receives network performance data (related to user services) in the form of high-speed electronic messages from the Space Network elements.

From WSGT, NCC receives Operations Data Messages (ODMs) every 5 seconds. These are generated by the Automatic Data Processing (ADP) equipment at WSGT, and indicate the status of all ongoing services through the TDRSS. They include such parameters as RF beam pointing angles, link status, signal strength, and bit error rate. At NGT, data quality monitoring is performed using Frame Analyzers. Data quality information produced by the Frame Analyzers is combined into Fault Isolation Monitoring System (FIMS) reports, which are sent to the NCC every 5 seconds. These messages include parameters such as frame sync lock status, and percentage of frames in lock. Figure 2 illustrates the additional high-speed message flow which is currently available to support NCC operations.

Currently, the contents of these messages are combined and presented on display screens to NCC operations personnel. Network Controllers and Performance Analysts monitor these displays, as well as ground control (reconfiguration) messages that alter the scheduled equipment configuration, to detect problems and determine appropriate courses of action. This task is an extremely labor intensive process. The quantity of information contained in the messages requires at least one (and in some cases two) display screens per service.

At the time of this writing, only one TDRS is operational, supporting a small number of user spacecraft. The Network Controllers and Network Performance Analysts are able to work together in teams to provide complete coverage of all ongoing services. However, they are reaching an information saturation point. When the second TDRS is launched, and the number of user spacecraft increases, the manual approach to network monitoring and fault detection/isolation will be inadequate. Some form of automation will be needed in order for the NCC to satisfy its mission-critical requirement to assure the quality and continuity of Space Network services. The purpose of FIESTA is to provide an intelligent assistant to the Space Network Controllers and Performance Analysts, that will continuously monitor selected services, detect faults, bring these faults to the attention of the controller/analyst, and isolate the source of a problem to the major system component level.

3.0 THE PROTOTYPE DEVELOPMENT ENVIRONMENT

The FIESTA prototype is currently being built in an off-line environment. This parallels conventional software development in which on-line validation occurs after off-line development and thorough system testing. The off-line development environment is made to look as much as possible like the NCC in terms of the data available and the software environment. A decision was made early to use a front end processor to isolate and simulate NCC interface functions. The benefit of this approach is that it separates the procedural interface functions from the knowledge-based functions that are the primary interests of the project.

High Speed Message (HSM) traffic from the Space Network is recorded at the NCC. These magnetic tapes provide "Automatically-obtainable situation data" (AOSD) used by the FIESTA standalone prototype. Data sources include actual missions as well as simulations run to create test data sets.

The FIESTA prototype consists of two computers and two bodies of software that communicate via commercially available networking tools. For the expert system component, the decision was made to use "off-the-shelf" hardware and a well supported expert system development shell. Such tools provide essential capabilities (e.g., an inference engine, a flexible data and knowledge representation language) and allow the development effort to be focused on the knowledge engineering task. Inference Corporation's Automated Reasoning Tool (ART) hosted on a Symbolics 3640 Lisp Machine was chosen. The FIESTA Front End Processor (FFEP) was prototyped on a VAX 11/730. The VAX was used for the sake of convenience and compatibility with other tools.

The NCC Simulator and AOSD Transformer

The FFEP software was christened "NSAT" for NCC Simulator and AOSD Transformer software. The NSAT was built to perform the following functions for the FIESTA prototype's development:

- To "simulate" the NCC insofar as providing AOSD to the rest of FIESTA,
- To isolate the expert system components from the preprocessing functions performed in the NSAT, the goals being to make the expert system component think it is fielded and to isolate NCC interface functions,
- To offload the Lisp Machine, translating coded HSM values to symbolic expressions amenable to use by the expert system application, and
- To provide a tool useful in development and testing of FIESTA - one for "feeding" the expert system components controlled amounts of data at controlled times; the programmer needs a tool to replay given scenarios to test and debug the expert system.

Figure 3 depicts the main components and data flows of the prototype. The prototype has been implemented in the STI development lab shown in Figure 4. The NSAT reads and translates AOSD from a disk file, transmitting it to the Lisp Machine under user control. A synchronizing Lisp Machine process reads AOSD over the net and does a bit of housekeeping. It makes the AOSD available in the Lisp world for the FIESTA process. (The Lisp Machine has only one address space so that global symbols can be used to pass arbitrary objects between Lisp Machine processes.) The FIESTA process then parses the AOSD and asserts it as a set of "facts" (one fact per service) into the ART database. When asserted as facts this data becomes available automatically to fault detection and diagnosis rules.

4.0 FIESTA OPERATOR/SYSTEM INTERFACE

The basic operational concept for FIESTA is to support NCC operations by automating network monitoring and fault detection as well as the reasoning process involved in fault isolation. To support these functions (monitoring, detection and isolation), an effective operator interface is required to allow FIESTA to serve as an intelligent decision aid.

This interface is provided by various types displays on a video terminal. Some information needs to be continuously and immediately available to the operator. This information includes the current services being monitored, their status, time of the latest message received, and an area for the display of alarm notifications. The current FIESTA system status and an operator options interface are also required. This information is grouped in a reserved area at the top of the screen. The space below is then free space, which the operators can customize with the displays of immediate interest. These five key windows are shown in Figure 5.

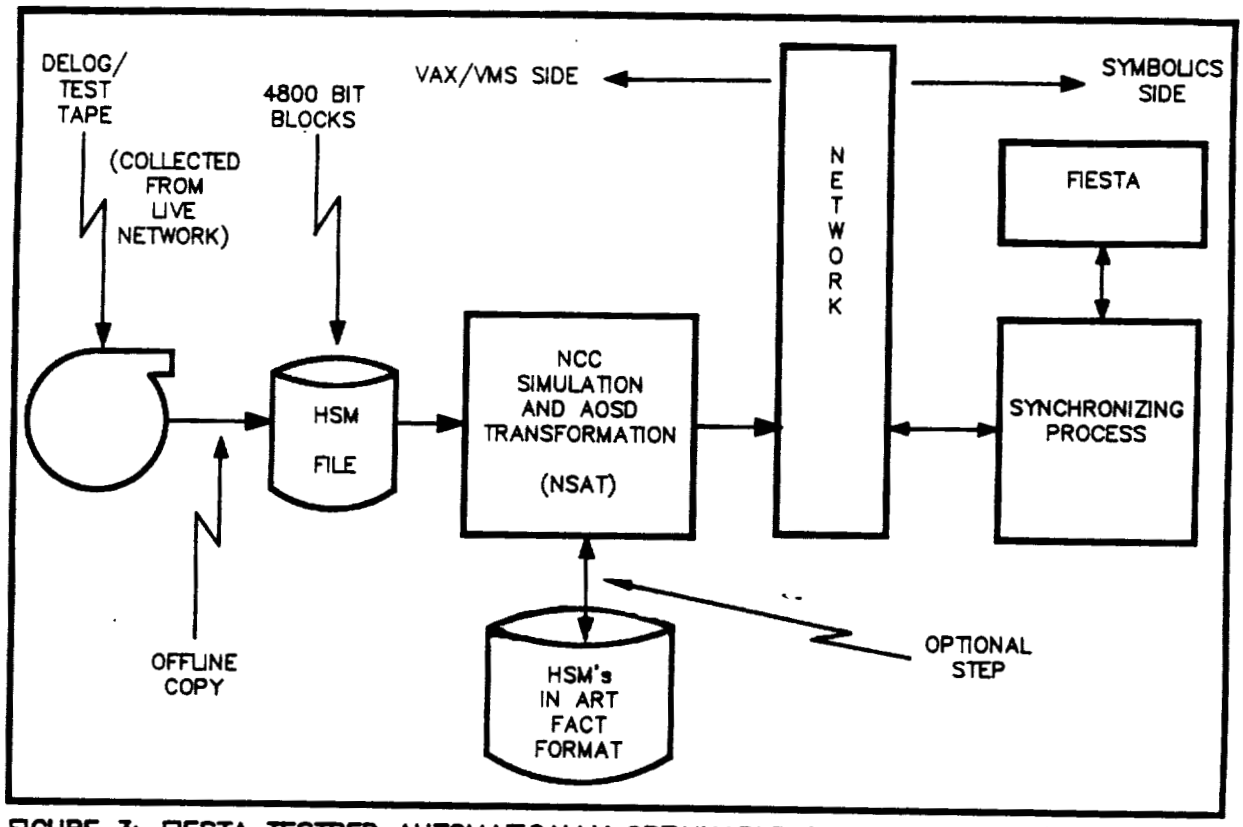


FIGURE 3: FIESTA TESTBED AUTOMATICALLY OBTAINABLE SITUATION DATA (AOSD) FLOW

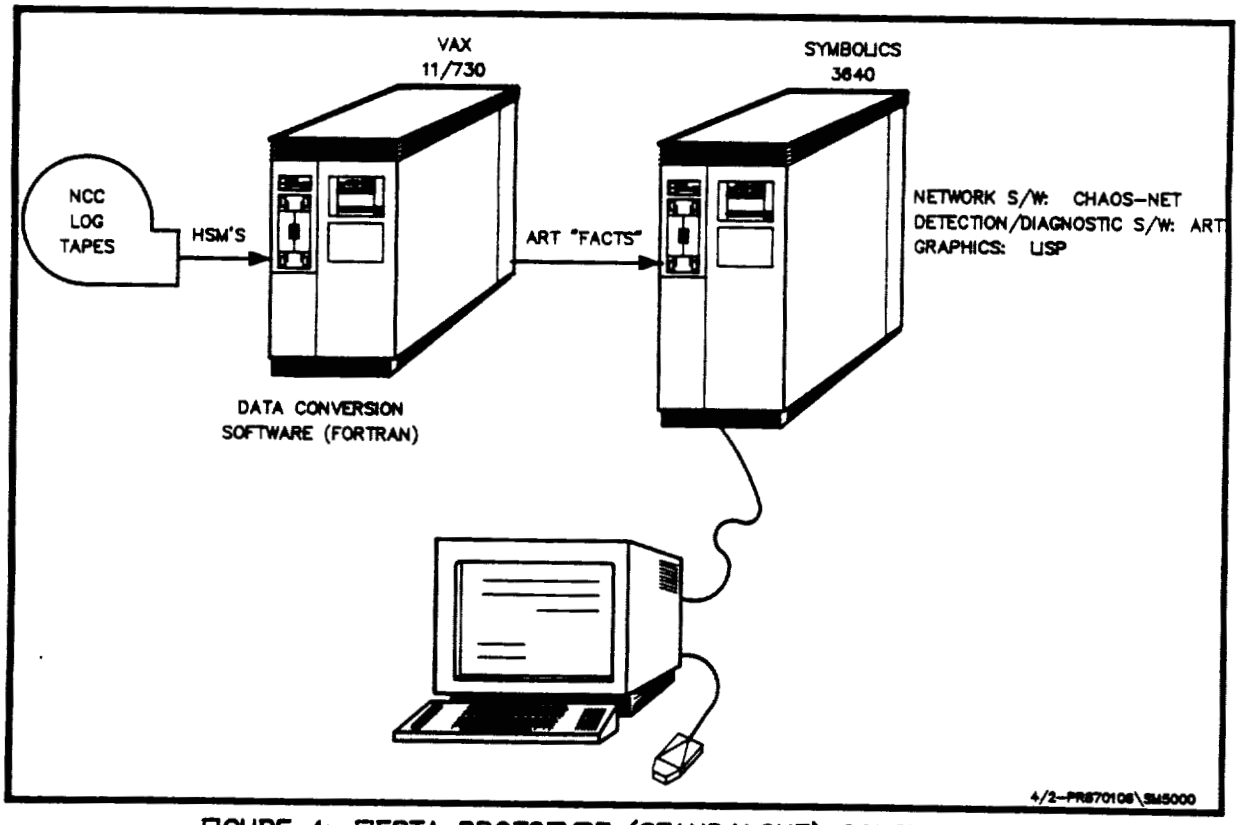


FIGURE 4: FIESTA PROTOTYPE (STANDALONE) CONFIGURATION

4.1 MONITORING DISPLAYS

Monitoring display are provided in several ways. The services window provides a high-level summary of the service status by reporting it as "acquiring", "nominal", "non-nominal", "transitional", or "late". A status of "acquiring" indicates that the link is in the process of being established, and dropouts should be expected. "Nominal" indicates that a good link has been established and all data looks good. "Non-nominal" signifies that bad data has been detected or that acquisition time is excessive. "Transitional" indicates that the service had been non-nominal, but now good data has returned. (to avoid treating rapid transient dropouts as separate faults a stabilization period is introduced before returning the status to nominal). Finally, "late" indicates that there are less than two minutes of scheduled service time remaining, a period when dropouts are common.

At a more detailed monitoring level, dynamic displays show relevant parameters in the high-speed messages. These are available as either cockpit or trending displays. Cockpit displays provide a "snapshot" of the most important parameter values and are automatically refreshed as new data arrives. The trending displays provide a history of parameter variation updating the graphs with additional information as messages are received. Figure 5 also provides example of each of these display types.

4.2 FAULT DETECTION DISPLAYS

In order to draw special attention to the occurence of significant fault related events, a notification window was implemented.

Whenever faults are detected or diagnoses are updated, a short message indicating the affected services and time of occurrence is posted. This is in addition to the summary status of non-nominal which will appear in the service window when a fault is detected.

4.3 FAULT ISOLATION DISPLAYS

When the notification is acknowledged by the operator, an explanation window is displayed. This window provides a description of the anomaly and the most likely diagnosis. The explanation window contains a justification option. When "justification" is selected, another window is opened which displays the evidence used in determining the conclusion shown. Both of these displays are shonw in Figure 6.

Additional diagnosis support is provided by the alternative hypotheses display. This display shows all the diagnostic alternatives currently under consideration and the positive and negative evidence for each. Further support as an intelligent decision aid is provided by the Recommended Action option. The recommended action window presents a series of alternative actions in order of probable usefulness.

The types of screens presented support the display requirements associated with the operational concept. Both operators and human factors engineers aided in the evaluation and evolution of the display concepts. Their involvement has contributed to a display system which is now ready for evaluation in an on-line environment.

5.0 ARCHITECTURAL OVERVIEW

The FIESTA Architecture was influenced by three critical elements. These were:

- The presence of independent data sources providing real-time situation data
- The requirements for supporting the operator interface to the system.
- The function of reasoning and inferencing which was central to the application.

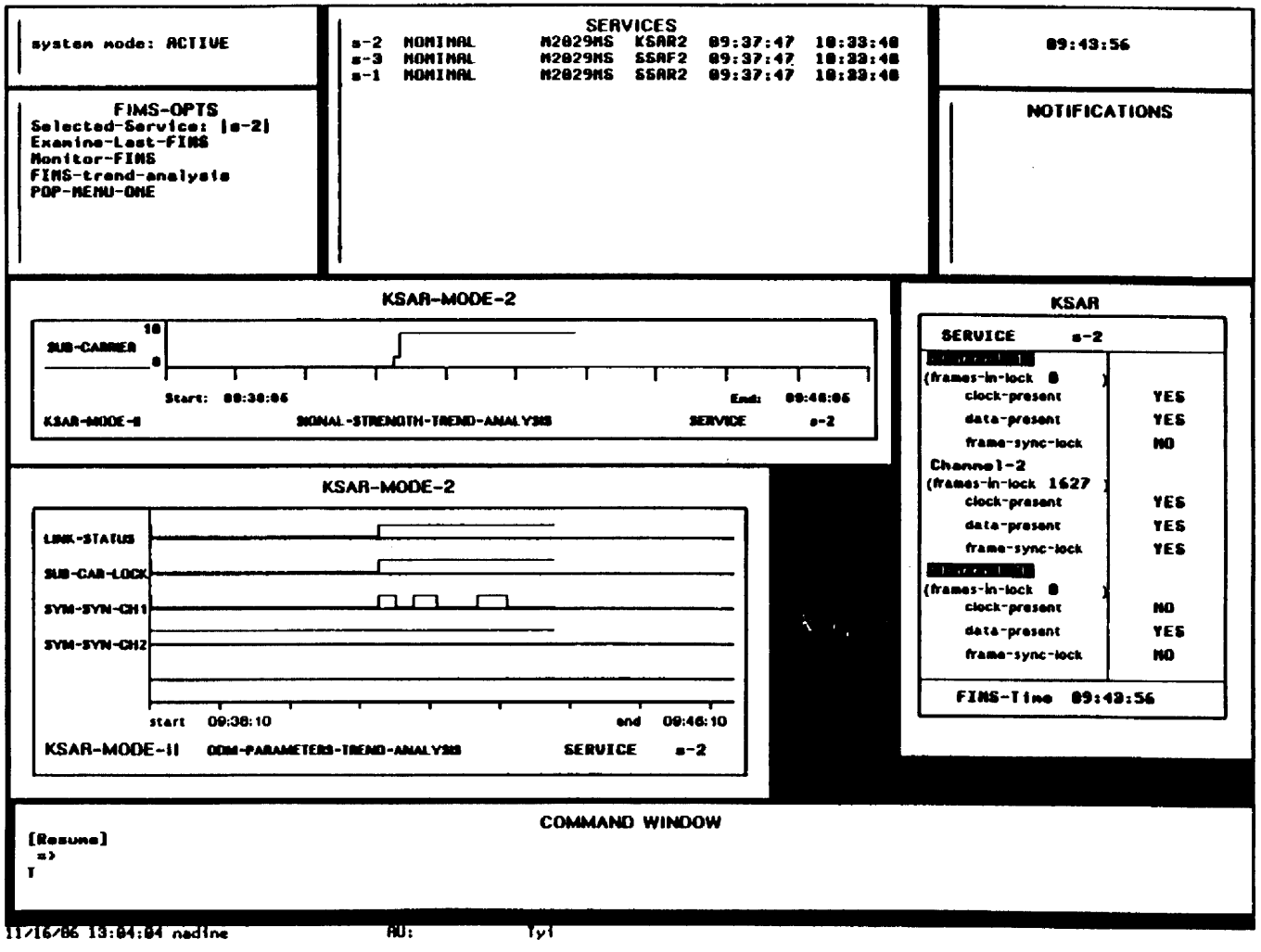


FIGURE 5: MONITORS AND DETECTION DISPLAYS

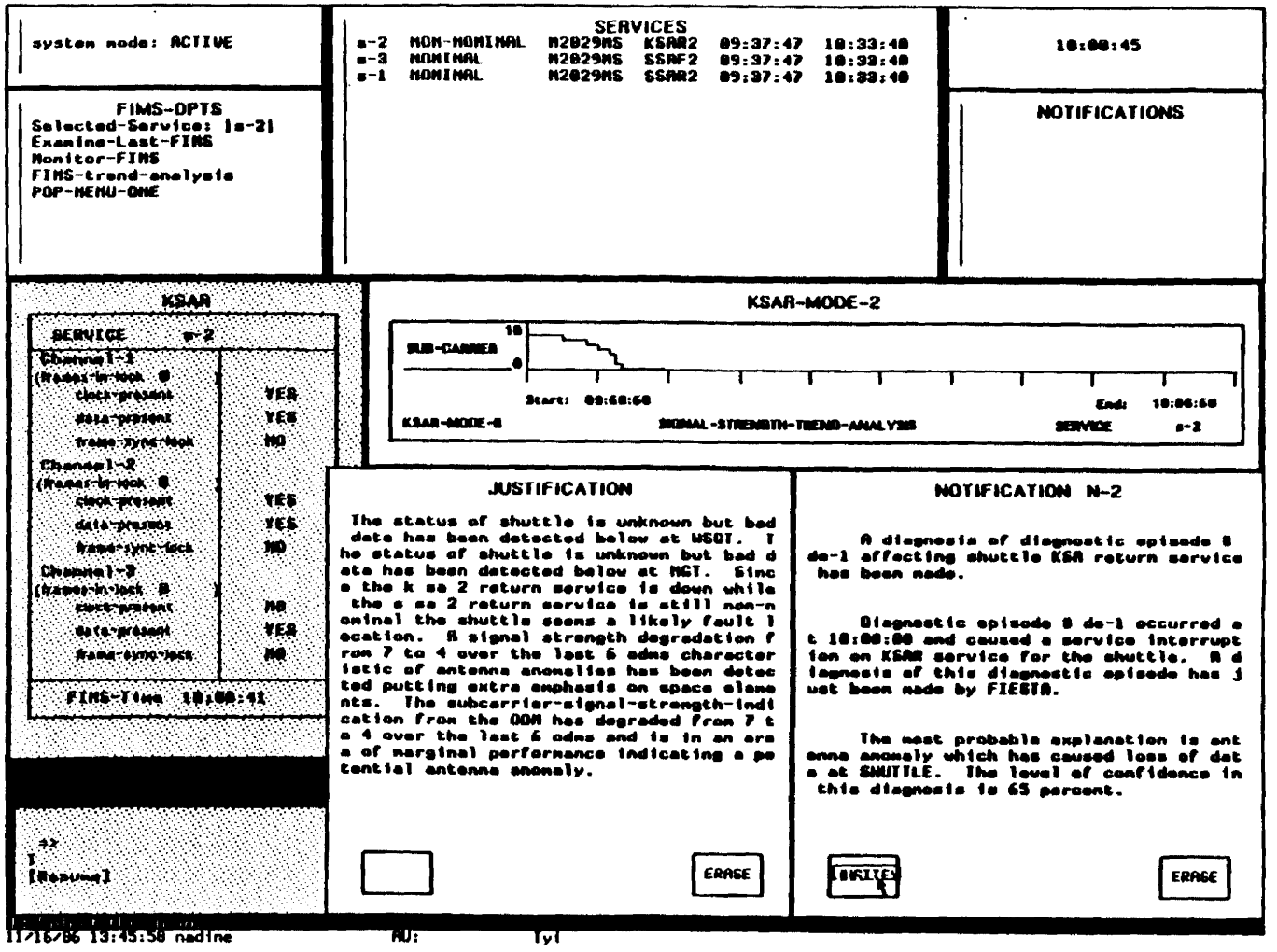


FIGURE 6: DIAGNOSTIC DISPLAYS

FIESTA rules ("productions") fall into three basic categories, corresponding to the major architectural design drivers. These are:

1) The front-end processor rules (interface to situation data);
2) The run-time monitor (interface to the operator); and
3) The expert system component (reasoning).

The expert/reasoning portion may be further subdivided into three parts (Kernel, Diagnostic and Assistant) reflecting specific fault reasoning functions performed by the system.

The FIESTA system architecture features three-way asynchronous operation of data handling, reasoning, and operator interaction to coordinate the elements involved. This asynchronous operation is accomplished by interfacing all components through a central knowledge base. Figure 9 provides an overview of the system architecture. The FIESTA front-end processor serves as a gateway for AOSD which is arriving from external data sources in real time. The FIESTA reasoning expert combines the situation data with its prestored knowledge base of rules, propositions, and frames to derive conclusions about the health of the system. Finally, the run-time monitor sends alerts and requested displays to the operator. It also enables the operator to request more information or enter any Manually Obtainable Situation Data (MOSD) that may be available. This data provides further information (via voice, special console, or hardcopy report) beyond that which is available automatically through high-speed message traffic.

The synchronization necessary to coordinate these components is provided by the fact database. The front-end processor, FIESTA expert, and run-time monitor consist of processes and rules which maintain and monitor this fact database. Activation of the rules is a data-driven function of the inference engine (provided by the ART package). Thus the processing of the inference engine addresses the problems associated with dealing with external data sources in real-time while simultaneously reasoning about the incoming data and responding to operator requests for data/advice (without overly restricting the timing of those

requests). Figure 7 summarizes these architectural concepts, indicating the coordinating role played by the knowledge base and the groups of rules which provides the system capabilities.

6.0 METHODOLOGY

This section discusses the fault isolation methodology applied to the problem domain with an emphasis on how the application domain structure was mapped to a software design solution. The following elements comprise this domain structure:

- Operational Support Organization
- Acquisition Determination
- Fault Detection
- Fault Diagnosis
- Uncertainty Management

In designing and developing a knowledge-based system, it is extremely important to recognize and take advantage of the natural organization of this knowledge. The way the expert views and approaches the problem should drive the design of the system.

6.1 OPERATIONAL SUPPORT ORGANIZATION

Operational support at the Network Control Center is organized on a user spacecraft basis. For example, during Space Shuttle flights, a Shuttle operations team handles the Shuttle fault detection and diagnosis responsibilities at the NCC. With other users, a different set of operators is responsible for their support. Thus, operational specific knowledge has developed around each individual user spacecraft with a common base of general knowledge that is applicable to all users.

Real-time operations are organized on an event basis. An event is defined as a scheduled support period for one user spacecraft for one pass by the Tracking and Data Relay Satellite (TDRS). The event can be further broken down by services within that event. A service is defined

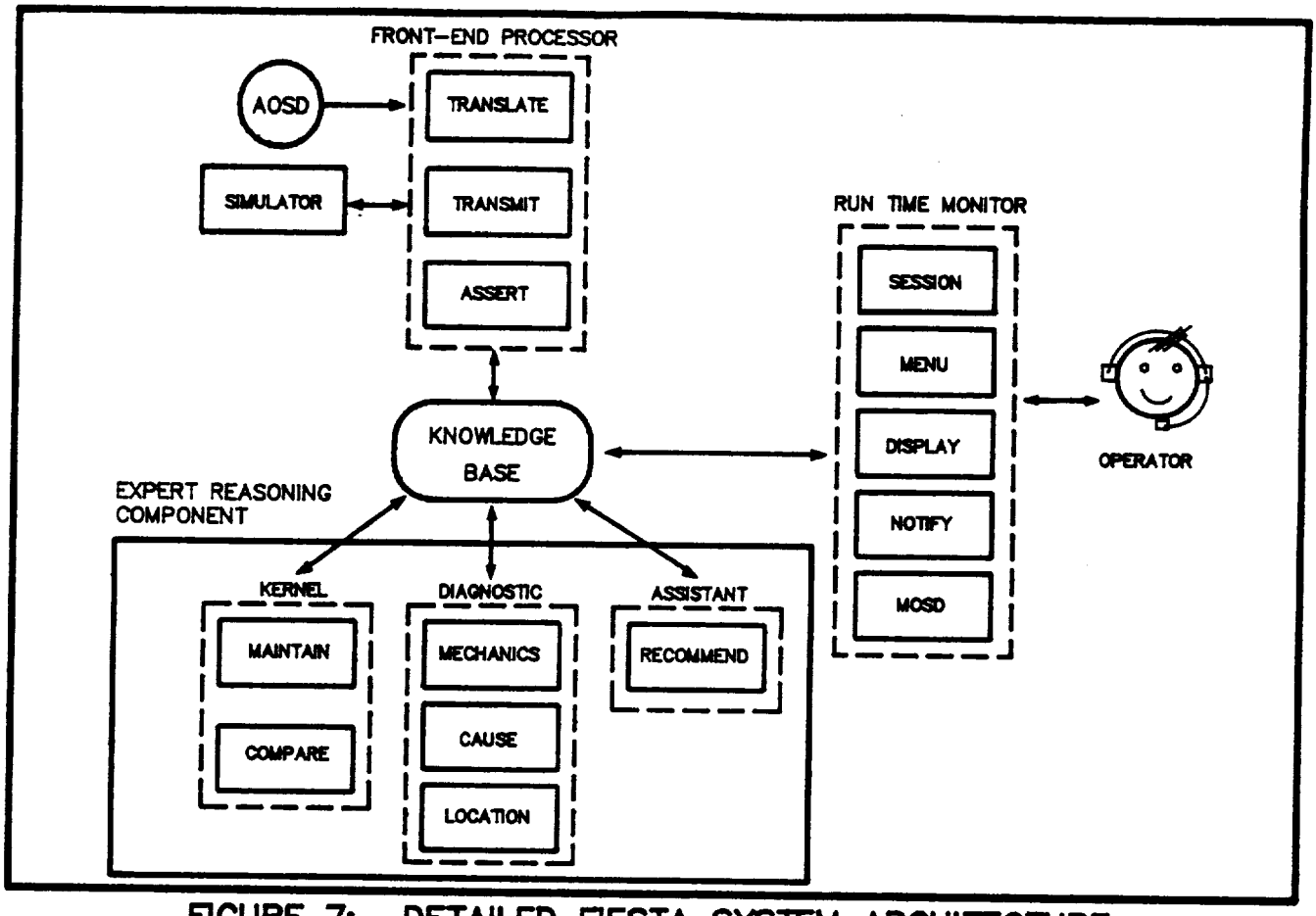


FIGURE 7: DETAILED FIESTA SYSTEM ARCHITECTURE

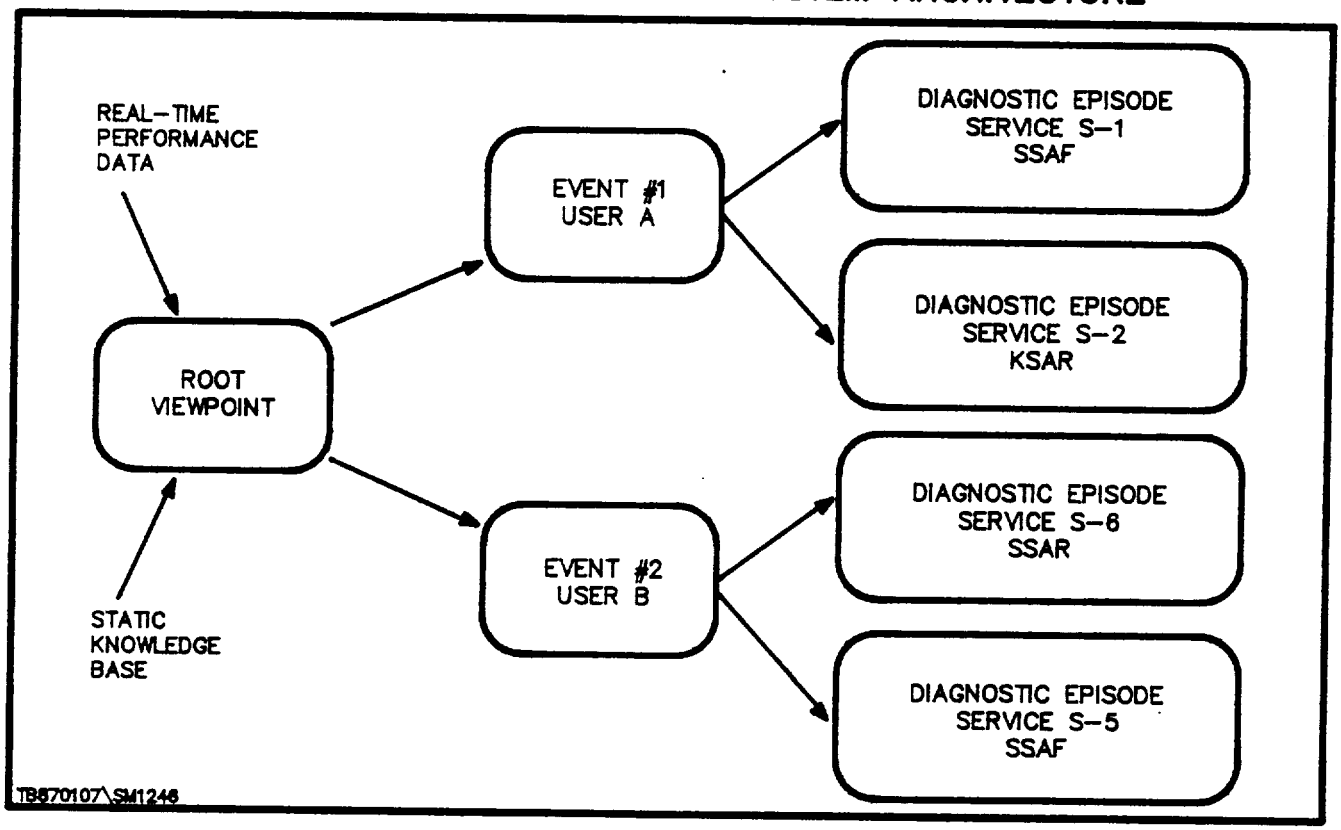


FIGURE 8: VIEWPOINT STRUCTURE

as a single communication link provided by the network (e.g., Ku-Band Single Access Return, or Multiple Access Return). All performance data received at the NCC is service specific, and thus faults are detected and diagnosed from a service perspective within the context of the event in which they occur.

Space Shuttle support was selected for this initial FIESTA prototype development. A Shuttle event consists of three services: S-Band Single Access Forward (SSAF), S-Band Single Access Return (SSAR), and Ku-Band Single Access Return (KSAR).

Figure 10 illustrates the type of data organization which was employed to separate generic Shuttle support knowledge from event-specific knowledge and event-specific knowledge from service-specific/fault-related knowledge. (The ART development package supports this tree-like organization, which includes inheritance, via its viewpoint mechanism. The nodes are referred to as viewpoints and this terminology will be employed).

The root viewpoint contains the static knowledge base (e.g., generic knowledge of the Space Network) as well as the dynamic situation data. The viewpoints sprouted off the root relate to separate user events currently monitored by the operator. Event-specific knowledge, such as appropriate nominal values and ranges, scheduled equipment chains, and event status and trends, reside in this viewpoint. The event viewpoints have access (through inheritance) to the background knowledge and situation data from the root viewpoint. All monitoring and real-time service support occurs in this event viewpoint.

When FIESTA detects an anomaly on a specific service, it sprouts a new viewpoint off the affected event node. This is termed a diagnostic episode. Detection of faults on other services will in turn result in new (diagnostic episode) viewpoints sprouted off the event node. This allows the system to independently reason about each diagnostic episode

and also to share information among simultaneous diagnostic episodes (searching for commonalities) through the event viewpoint, where they share a mutual parent.

6.2 ACQUISITION DETERMINATION

The concept of acquisition plays a major role in FIESTA's diagnostic approach due to signal behavior differences before, during, and after acquisition. The normal event start time for the Shuttle precedes actual signal and data acquisition by several minutes. This implies that for the first several minutes of service the performance parameters will indicate the presence of no signal and no data. As the Shuttle comes into line-of-sight of the TDRS, the signal may behave erratically before finally locking up. Not until the signal exhibits steady signal-strength and demodulator-lock values will the operators consider service nominal. In addition, an important differentiation must be made between signal and data acquisition because:

- Signal acquisition can occur without data acquisition
- The Shuttle may schedule data on a channel for an entire event but only transmit on that channel for a portion of that event
- Positive signal acquisition without data acquisition points the problem to different diagnostic areas than combined negative signal and data acquisition.

The FIESTA design had to account for these operational characteristics to recognize that Shuttle services routinely lock up late, that erratic behavior can be expected during acquisition, and that acquisition is a two-step process (signal and data).

Immediately following event startup, FIESTA tracks the status of all signal-related performance parameters for the return services. When these parameters have remained nominal for two consecutive minutes on a particular service, FIESTA determines acquisition is complete and will

begin to monitor the service for ensuing anomalies. Any signal-related out-of-range or out-of-tolerance conditions that are detected will no longer be considered acquisition idiosyncrasies but actual faults. At this point FIESTA changes the status of the service from "ACQUIRING" to "NOMINAL" both internally and externally (notifying the user). Data acquisition is handled similarly through monitoring of appropriate bit/symbol-sync lock and frame-sync lock parameters.

6.3 FAULT DETECTION

On the surface, fault detection seems trivial: identify anomalous conditions through the monitoring of performance parameters or through user notification and initiate the fault diagnosis process. Upon closer inspection, the problem becomes more complex and logic-based than one would first expect. Before operators can recognize an out-of-tolerance or out-of-range condition, they must first identify (albeit subconsciously) the nominal value or range that is exceeded. For example, nominal signal strength range will vary among users and allocated equipment chains. Operators also routinely ignore apparent signal or data dropouts caused by service configurations, reconfigurations, and service-to-service handovers. The ability to differentiate seemingly anomalous behavior from actual anomalous behavior is an expert capability that is easily overlooked. FIESTA anticipates the majority of these types of conditions and does not open up diagnostic episodes for expected service dropouts. This capability proves to be extremely important for system performance considerations to avoid paying unnecessary diagnostic processing penalties and to minimize false alarms.

Fault detection rules provide an independent source of network status information to the diagnostic rules. Diagnostic rules rely on this network status data to reason about current conditions. The detection rules monitor for anomalous conditions, detect non-nominal parameters and initiate diagnostic episodes based on detected conditions, continue

to monitor and provide network status information to the diagnostic rules through a diagnostic episode, and determine when the diagnostic episode can be terminated.

6.4 FAULT DIAGNOSIS

To model the fault diagnosis process, the FIESTA design tried to parallel the thought patterns of the experts by selecting the most likely fault from a pool of known possibilities. While experts often arrive at the solution immediately, the subconscious steps they took to get there will comprise the diagnostic methodology FIESTA needs to follow. The model developed involves hypothesizing all known fault causes and fault locations (at a high level), immediately ruling out some or most, and pursuing others to a lower level of detail. This hierarchical fault cause/location model utilizes a hypothetical reasoning structure to rank the possibilities and present the most likely pair.

The highest level hierarchical breakdown occurs between fault location and fault cause. Both of these branches provide significant information on fault conditions in the network. The two branches can be viewed as two independent experts, a fault location expert who determines the point at which data was lost and a fault cause expert who explains why. After their analysis, the two experts confer and present the most likely fault location and cause to provide a twofold explanation of the anomaly.

Utilizing the natural hierarchy of TDRSS Network fault causes and fault locations provides FIESTA with the flexibility to diagnose locations and causes in as much detail as possible. In some cases, the available data may not be sufficient to pin down a specific piece of equipment or a particular network operator, but the data may be sufficient to isolate the location to a network element (e.g., a specific ground terminal), set of elements (e.g., either the Shuttle or the relay satellite), or isolate the cause to an operational type of error at a high level.

Hence, this structure attempts to give the diagnosis as much flexibility and depth to accommodate varying levels of operator experience and skill and to adapt to a variety of data availability conditions.

6.5 UNCERTAINTY MANAGEMENT

A fundamental design characteristic of an expert system is its uncertainty management approach. The inferences humans make are often uncertain; certain conditions may "suggest," "sometimes result in," or "may mean" a corresponding conclusion. Thus, a mechanism must be developed to:

- Represent probabilistic statements;
- Gather and combine evidence for and/or against a certain hypothesis; and
- Present this hypothesis and its "certainty" to the user.

Various alternatives exist for representation of uncertainty (Dempster-Shafer Theory of Evidence, fuzzy logic, Bayesian inference). The technique chosen for FIESTA was the MYCIN CF Model [1].

The following reasons form the basis for our choice of the CF Model: (1) the MYCIN CF Model is a standard in the field in that it has withstood the test of time, scrutiny, and numerous implementations outside of its original application; (2) the MYCIN domain was diagnostic as is FIESTA's -- FIESTA monitors incoming symptoms (network performance data), detects and diagnoses the problem and acts as an operational consultant; and (3) this model is easily implementable via LISP and ART code. Another feature we have found through our prototyping efforts is that the CF Model is easily understandable and integrates well into the operational psyche of its intended user community.

---

[1] Buchanan, B. G., and Shortcliff, E. M., 1984. Rule-Based Expert System: The MYCIN Experiments of the Stanford Heuristic Programming Project, Reading, Mass., Addison-Wesley.

Summary

This paper has presented some highlights of the expert system FIESTA and the knowledge engineering effort which has supported its development. FIESTA is targeted for on-line deployment in the NCC. Current development efforts are focused on the transition from its current standalone prototype mode to that of an on-line test bed environment. Issues of real-time data acquisition and real-time performance of the demonstrated capabilities are currently being addressed. These capabilities support the operational concepts of automation and provide an illustration of how expert system methodology can expand automation concepts to include reasoning and decision aid support.

ACKNOWLEDGMENTS

FIESTA development has been undertaken by Stanford Telecommunications, Inc. as a subcontractor to the Bendix Field Engineering Corporation (BFEC). Project support has been provided by the National Aeronautics and Space Administration under Contract #NAS5-27600.

The overall support and encouragement of Paul Ondrus and Anthony Maione of NASA/GSFC are particularly appreciated and acknowledged. The knowledge engineering effort was fortunate to have the operational expertise and cooperation of Joe St. John and Bill Webb of BFEC. Testing materials and network insights which have proved invaluable in development efforts have been provided both directly and indirectly by Tom Gitlin (BFEC). The administrative and logistics efforts of Hugh Bath (BFEC) and his technical staff were most supportive.

Portions of this paper have been drawn from the article "Expert System Fault Isolation in a Satellite Communication Network" prepared for inclusion in the book EXPERT SYSTEM APPLICATIONS TO TELECOMMUNICATIONS, J. Liebowitz, ed. to be published by John Wiley and Sons, Inc. (tentatively 1987).

Expert System Support for HST Operations

by

Bryant Cruse
Lockheed Missiles and Space Corp.
Code 400.8, NASA/GSFC
Greenbelt, MD 20771

Charles Wende
Space Telescope Project - Goddard
Code 400.2, NASA/GSFC
Greenbelt, MD 20771

ABSTRACT

An expert system is being developed to support vehicle anomaly diagnosis for the Hubble Space Telescope. Following a study of safemode entry analyses, a prototype system was developed which reads engineering telemetry formats, and, when a safemode event is detected, extracts telemetry from the downlink and writes it into a knowledge base for more detailed analysis. The prototype then summarizes vehicle events (limits exceeded, specific failures, etc.). This prototype, the Telemetry Analysis Logic for Operations Support (TALOS) uses the Lockheed Expert System (LES) shell, and includes over 1600 facts, 230 rules, and 27 goals. Although considered a prototype, it is already an operationally useful system.

The history leading into the TALOS prototype will be discussed, an overview of the present TALOS system will be presented, and the role of the TALOS system in contingency planning will be delineated.

The Hubble Space Telescope (HST) is not, for the most part, an autonomous spacecraft. Its engineering telemetry will be monitored for vehicle health and safety on a nearly continuous basis from the ST Operations Control Center (STOCC) at the NASA/Goddard Space Flight Center in Greenbelt, MD. STOCC personnel must recognize and respond to anomalies by initiating the appropriate contingency procedures. One exception to this dependance on ground personnel is the vehicle safemode system. An on-board computer continually tests critical vehicle subsystems. When one of these tests fails, a predefined sequence of stored commands is exercised to place the vehicle in a safe configuration, or a "safemode". Several safemodes have been defined and are activated depending on the nature and severity of the malfunction. Each safemode is designed to isolate the failed subsystem or component and then to place the vehicle in a stable, powerconserving attitude. The safemode system buys time for the STOCC personnel to respond to a serious on-board situation. It is still incumbent upon the STOCC to recognize that the vehicle has entered safemode, to determine which safemode test or tests have failed, and to diagnose the cause of the problem. These tasks must be acomplished before the vehicle can be recovered and the science schedule resumed. The development of tools that can speed up these analyses, therefore, has a very high payoff for enhancing mission operations.

Analysis of a vehicle safemode event requires analyzing raw telemetry which appears in one of a variety of formats depending upon, among other things, the type of safemode entry (which is to be determined!). Following a safemode recovery study in 1984, it was recognized that, because of the complexity of this task, some sort of ground software assistance would be needed if the HST were to be operated efficiently. Lockheed Missiles and Space Company, the Mission Operations Contractor (MOC) for HST, undertook to write a prototype expert system (the Telemetry Analysis Logic for Operations Support, TALOS, system) to attack this problem in the summer and fall of 1986.

In its current state of development, TALOS operates in either of two modes. In the monitor mode, TALOS scans a telemetry history file (optionally starting from a specified time) and looks for the existence of any safemode event; if an event is found, it automatically changes to the diagnostic mode. Upon entering the diagnostic mode, TALOS

extracts the values of specific telemetry monitors from the history file and writes them to the system knowledge base. The TALOS system then performs the following analysis tasks:

determines whether the HST itself is in a safemode;

and if so,

assesses the sequence of vehicle events,

summarizes what happened and when, and

verifies that the vehicle response was correct.

If desired, the operator can ask for a rationale explaining why any particular conclusion was reached. The TALOS system consists of four major subsystems, of which two were provided by the MOC and two were provided in the Lockheed Expert System (LES) shell:

- a Data Interface (developed by the MOC)
- a Knowledge Base (populated by the MOC),
- an Inference Engine (provided by LES), and
- a Knowledge Interface (also provided by LES, but customized for this application).

The Data Interface consists of an adaptive telemetry extraction program written in FORTRAN. Presently it reads data only from an HST engineering telemetry history tape; enhancements will allow reading real-time engineering telemetry streams or disk-based data. The extractor selects 170 monitors (data points) out of the 4690 monitors available, and performs quality checks before reformatting and forwarding the data. Eleven telemetry formats are available, with up to 4015 parameters being downlinked in any one format. Each of these parameters are sampled at least once every two minutes, and some are sampled many times in that interval. The telemetry format itself may be changed autonomously by the HST spacecraft when a safemode situation is encountered. Format changes in the telemetry stream are recognized automatically and are handled almost instantaneously by the Data Interface. The set of monitors being extracted can be changed in less than five seconds under the control of either the console operator or the expert system. The telemetry commutation schemes are stored in a database and are subject to change during the mission.

However, a different commutation scheme can be loaded into TALOS in a matter of seconds under either operator or expert system control. Thus, old data can be revisited for testing, training, or comparison purposes without requiring significant software changes or substantial operator intervention.

The Knowledge Base includes 1600 facts, 230 rules, and 27 goals. As an entity, it is already more knowledgeable about safemode entry than the average console operator was during the Ground System Thermal-Vacuum Test. Knowledge is represented in four ways:

backward chaining, goal-driven rules (if ... then...),

forward chaining, data-driven rules (when ... then...),

facts stored as slots in frames, and

goals which can be run concurrently with dynamically changeable priorities.

The following example illustrates a Backward Chaining Rule in the Knowledge Base:

```
HEADING:

  RULE_NAME          'SMEVENT1'
  FROM_WHOM          'BRYANT CRUSE'
  ACT_TIME           '17 NOV 1986'
  AUTHOR_ENGLISH     'If, after a gyro test failure in low'-
                     ' mode, the gyros are found in high'-
                     ' mode, the software response to the'-
                     ' test failure is nominal.'

IF:

  TYPE_ENTRY         'STATE'
  ACTOR              'RESULT[SMTEST(PNAME=SMTEST1)]'
  ACTION_VERB        '='
  OBJECT             'FAILED'

  TYPE_ENTRY         'STATE'
  ACTOR              'VALUE[MONITOR(PNAME=QDFHILO)]'
  ACTION_VERB        '='
  OBJECT             '0'

  TYPE_ENTRY         'ACTION'
  ACTOR              'ROBOT(PNAME=LES)'
  ACTION_VERB        'PRINT'
  OBJECT             'The DF-224 has responded normally to '-
                     'DESCRIPTION[SMEVENT(PNAME=SMEVENT1)].'

THEN:

  TYPE_ENTRY         'STATE'
  ACTOR              'EFFECT[SMEVENT(PNAME=SMEVENT1)]'
  ACTION_VERB        '='
  OBJECT             'VERIFIED_NOMINAL'
  LIKELIHOOD         '100'
```

This rule fires when the condition following the THEN statement is exactly matched by a condition following an IF statement of another rule or by an Hypothesis statement of a goal. When the rule fires the system then tries to find a match for the conditions following the IF statement in in the knowledge base or in the THEN clauses of other rules. Note the ACTION statement (third entry under IF). LES will execute such a statement in an IF clause when the other two statements are matched. In this case, a text block is written to the screen to inform the user of the result.

The following example illustrates a Forward Chaining rule:

```
HEADING:

  RULE_NAME          'SMWHEN-9'
  FROM_WHOM          'BRYANT CRUSE'
  ACT_TIME           ' 9-DEC-1986'
  AUTHOR_ENGLISH     'When the value of DTMFDC (telemetry '-
                     ' format data content monitor) is '-
                     ' determined and it is not equal to '-
                     ' 145 (S format) or 48 (C format) then'-
                     ' the number of safemode events equals'-
                     ' the value safemode fault recorder'-
                     ' pointer divided by 8.'

WHEN:

  TYPE_ENTRY         'STATE_CHANGE'
  ACTOR              'VALUE[MONITOR(PNAME=SSFRPTR)]'
  ACTION_VERB        'IS_DETERMINED'

  TYPE_ENTRY         'STATE'
  ACTOR              'VALUE[MONITOR(PNAME=DTMFDC)]'
  ACTION_VERB        'IS_DETERMINED'

  TYPE_ENTRY         'STATE'
  ACTOR              'VALUE[MONITOR(PNAME=DTMFDC)]'
  ACTION_VERB        '^='
  OBJECT             '145'

  TYPE_ENTRY         'STATE'
  ACTOR              'VALUE[MONITOR(PNAME=DTMFDC)]'
  ACTION_VERB        '^='
  OBJECT             '48'

THEN:

  TYPE_ENTRY         'STATE_CHANGE'
  ACTOR              'NUMBER_OF_EVENTS[EVENT_SEQUENCE(PNAME='-
                     'SMSEQUENCE)]'
  ACTION_VERB        '='
  OBJECT             'VALUE[MONITOR(PNAME=SSFRPTR)] / 8'
```

This rule will fire only when all four conditions following the WHEN statement are met. Those conditions are checked by the system each time a condition defined by a TYPE_ENTRY of 'STATE_CHANGE' undergoes some change in the knowledge base. When the rule fires, the condition following the THEN statement becomes true.

The following example illustrates a simple category file storing facts in slots in frames. This file defines the different safemodes to the expert system. Any number of attributes can be defined.

```
FILENAME:SAFEMODE_LEVEL.CAT

        SAFEMODE_LEVEL


/*** ATTRIBUTE DEFINITIONS

  ATTRIBUTE_NAME    ACTIVE
  TYPE_ATTRIBUTE    ACTIVE                'TRUE-FALSE'
  ASKABLE           ACTIVE                'FALSE'

  ATTRIBUTE_NAME    GROUND_COMMANDED
  TYPE_ATTRIBUTE    GROUND_COMMANDED      'TRUE-FALSE'
  ASKABLE           GROUND_COMMANDED      'TRUE'

/*** TOKENS

  PNAME             'SMLEVEL0'
  DESCRIPTION       '  No safemode events have occurred.  '

  PNAME             'SMLEVEL1'
  DESCRIPTION       '  The vehicle is in Inertial Hold Mode. '

  PNAME             'SMLEVEL2'
  DESCRIPTION       '  The vehicle is in Software Sunpoint
                    Safemode. '

  PNAME             'SMLEVEL3'
  DESCRIPTION       '   The vehicle is in Hardware Sunpoint
                    Safemode. '

  PNAME             'SMLEVEL4'
  DESCRIPTION       '  The vehicle is in Gravity Gradient
                    Mode. '

  PNAME             'SMLEVEL5'
  DESCRIPTION       '   The vehicle is not in Safemode.'-
                    '  However one or more safemode events
                    have occurred. '
```


C-3

The following example illustrates a Goal with a default priority of 95:

```
  PNAME                          'SMGOAL4'
  DESCRIPTION                    'determine the safemode level if
                                 any'
  GOAL_PRIORITY                  '95'
  SUBJECT_CATEGORY               'DETERMINE_SAFEMODE_LEVEL'
  FIND_ALL_SOLUTIONS             'TRUE'
  GOAL_RESULT_PUTOUT             'FALSE'
  GOAL_MESSAGE                   'I am now determining whether
                                 the vehicle'-
                                 ' has entered safemode and if so
                                 what level.'

  HYPOTHESIS

  '(ACTIVE[SAFEMODE_LEVEL(PNAME=SMLEVEL0)] = TRUE >< '-
  ' ACTIVE[SAFEMODE_LEVEL(PNAME=SMLEVEL1)] = TRUE >< '-
  ' ACTIVE[SAFEMODE_LEVEL(PNAME=SMLEVEL2)] = TRUE >< '-
  ' ACTIVE[SAFEMODE_LEVEL(PNAME=SMLEVEL3)] = TRUE >< '-
  ' ACTIVE[SAFEMODE_LEVEL(PNAME=SMLEVEL5)] = TRUE >< '-
  ' ACTIVE[SAFEMODE_LEVEL(PNAME=SMLEVEL6)] = TRUE )'
```

Within LES it is possible to alter the priority of a goal and cause a new line of reasoning to be pursued. This change is implemented using a forward chaining rule of a type generally called "Demons". An example of a demon follows:

HEADING:

```
  RULE_NAME          'SMWHEN-01'
  FROM_WHOM          'BRYANT CRUSE'
  ACT_TIME           '11-AUG-1986'
  AUTHOR_ENGLISH     'When no safemode events have occurred'-
                     ' reduce the priority of Goal-6 to 0'
```

WHEN:

```
  TYPE_ENTRY         'STATE_CHANGE'
  ACTOR              'ACTIVE[SAFEMODE_LEVEL(PNAME=SMLEVEL0)]'
  ACTION_VERB        '='
  OBJECT             'TRUE'
```

THEN:

```
  TYPE_ENTRY         'STATE_CHANGE'
  ACTOR              'GOAL_PRIORITY[GOAL(PNAME=SMGOAL6)]'
  ACTION_VERB        '='
  OBJECT             '0'
```

The Inference Engine provides the standard expert system functions. While in the monitor mode, TALOS reasons in a data-driven manner and awaits the detection of a safemode event before proceeding with the analysis. 20 of the 27 defined goals are initially set to a priority of zero. Upon entering the diagnostic mode, TALOS begins processing goal-driven, backward chaining rules. Then, the priority of a goal may be raised or lowered by data-driven forward chaining rules, depending upon how the analysis proceeds. (Did this fail? Are these monitors available in this format?, etc.) New data can cause further refinements of the priorities. Thus LES is capable of abandoning one line of reasoning and switching to another course of analysis depending upon what it discovers about the state of the HST spacecraft.

The Knowledge Interface (user interface) uses windows to keep the operator appraised of what it has found. At any time, one window is maintaining summary statistics on safemode events, while another window is giving details of the ongoing analysis. At the conclusion of its analysis, TALOS presents its findings and the operator may ask for a printout, or may ask for a detailed rationale behind the findings. By design, TALOS serves to advise the operator and cannot of and by itself issue any corrective commands to the HST spacecraft.

The TALOS has demonstrated its ability to scan a telemetry history tape, to identify an initial safemode event, and to analyse a complex sequence of events correctly. A particularly complex but logically consistant series of safemode events were placed on a telemetry history tape using the Hardware/Software Laboratory at Lockheed in Sunnyvale, CA. Analysis of the telemetry to decipher this sequence would present a real challenge, even to the most expert analyst, would typically require an hour. This sequence of failures proceeds as follows:

> First, the current in the vehicle's magnetic torquer bars exceeds safe limits. This anomaly causes a safemode test to fail, and an on-board computer commands the vehicle to the first level of safemode: Software Sunpoint.
>
> As a result, the solar panels are commanded to rotate. But, since there are no solar panels in the laboratory, another safemode test fails.

Next the battery depth-of-discharge fails through two successive limits (logically, since the solar arrays are mis-aligned). This last failure would normally result in entry into the next level of safemode: Hardware Sunpoint.

In this last level of safemode, a backup computer shuts down the primary on-board computer. However, the lab doesn't have a backup computer either, so the vehicle response is again anomalous.

The printout produced at the end of the analysis clearly shows this sequence of events. On an unloaded system, this entire analysis takes only a few minutes.

TALOS should be understood as applying current technology to the contingency analysis problem. Contingency planning includes:

anomaly recognition,

immediate action definition,

diagnostic techniques, and

recovery plans.

Present TALOS capabilities include fault identification with rationale. Contingency planning maps directly into present and potential TALOS functions; the further development of TALOS will build on our contingency planning. Conversely, TALOS will provide a framework for codifying such planning. By merging the two, it is expected that the TALOS development will force higher degrees of organization, consistency and completeness upon the contingency planning process. The cost will be in training operations personnel to care and feed the TALOS knowledge base, and in the time it takes for these people to insert their contingency plans into the knowledge base itself. However, by testing TALOS against HST spacecraft or simulator data, the contingency analyses can be validated directly, a more thorough testing of TALOS is provided, and a training tool is provided for personnel. Further, the self-documenting nature of the TALOS knowledge base provides paper procedures when needed, while the explanation feature of TALOS provides a teaching tool for new personnel and develops rationales for some unexpected cases.

Development costs thus far have been on the order of a few months of effort and liberated time on a shared VAX/8600 (TALOS also operates quite well solo on a MicroVAX II/GPX). The concept has been demonstrated, and its capabilities will be expanded. The near-term development of additional TALOS capabilities will proceed cautiously, as the a cost of augmenting contingency planning with an expert system will have to be ascertained. TALOS will not be immediately expanded to cover all possible contingencies, but instead will be directed at a small number of high return situations. Three diagnostic modules will be added to service the pointing control system (PCS), the electrical power system (EPS), and the data management system (DMS), and these modules will be limited to handling contingencies related to vehicle safemodes.

The results to date have been very promising.

MIS30G

A HIERARCHICALLY DISTRIBUTED ARCHITECTURE FOR FAULT ISOLATION EXPERT SYSTEMS ON THE SPACE STATION+

STEVE MIKSELL
SUE COFFER*
STANFORD TELECOMMUNICATIONS, INC.

Abstract

The SAFTIES (Space Station Axiomatic Fault Isolating Expert Systems) System deals with the hierarchical distribution of control and knowledge among independent expert systems doing fault isolation and scheduling of Space Station subsystems. On its lower level, fault isolation is performed on individual subsystems. These fault isolation expert systems contain knowledge about the performance requirements of their particular subsystem and corrective procedures which may be involved in response to certain performance errors. They can control the functions of equipment in their system and coordinate system task schedules. On a higher level, the Executive contains knowledge of all resources, task schedules for all subsystems, and the relative priority of all resources and tasks. The executive can override any subsystem task schedule in order to resolve use conflicts or resolve errors that require resources from multiple subsystems. Interprocessor communication is implemented using the SAFTIES Communications Interface (SCI). SCI is an application layer protocol which supports the SAFTIES distributed multi-level architecture.

1.0 INTRODUCTION

The SAFTIES project included the following goals:

- Design a hierarchical system where one expert system (executive) coordinates many expert systems to isolate faults and do related processing.

+ Work supported under NASA contract NAS5-29280.

* Sue Cofer participated in this project while employed by STI. She is currently associated with Digital Equipment Corp.

- Define a communications protocol (applications layer) to support expert system interaction
- Build a demonstration system which will form the basis for a test bed for further definition and evaluation of distributed expert system capabilities

This paper provides an overview of the demonstration system which was developed to provide a proof-of-concept testbed for achievability of these goals.

2.0 THE DISTRIBUTED HIERARCHICAL CONFIGURATION

Many of the feasibility questions associated with distributed processing and hierarchical control can be most readily addressed in a testbed configuration. Fault isolation was selected as the base-level element in the hierarchy. In order to develop requirements which would support the development of a meaningful test bed demonstration, it was necessary to identify specific areas where the fault isolation would be performed. Selection was influenced by an awareness of the potential long-term space-based applications. The general domain areas chosen for this processing were:

1) Robotics system (FIRES)
2) Communications system (FIESTA)
3) Environmental factors reporting system (FISHES)

To support project objectives, the architecture for a Space station Axiomatic FaulT Isolating Expert System (SAFTIES) (shown in Figure 1) was developed.

As the figure indicates, two levels of control are present. Within the lower level, the separate expert systems are operating on their defined domains of responsibility. Communication between levels is provided by the SCI (SAFTIES Communications Interface). The DMS (Data Management System) which would support the SCI is presented symbolically in the overview.

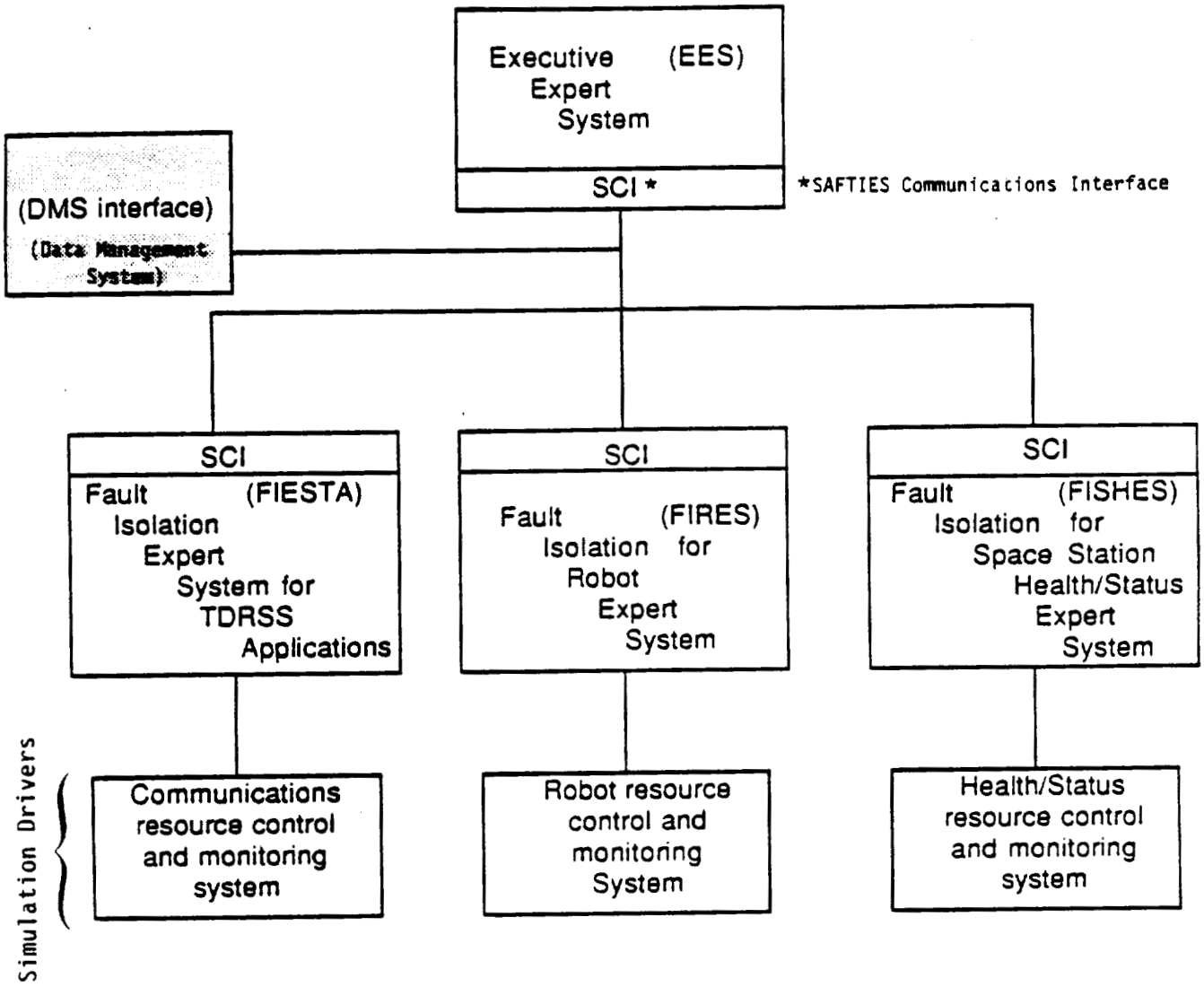


FIGURE 1: SAFTIES SYSTEM OVERVIEW

On the upper level, the executive function provides conflict resolution between the various subsystems as well as allowing for the optimal satisfaction of mission objectives by enforcing a prioritization of subsystem objectives.

A hardware configuration which would provide a distributed environment was identified. Functional allocation over the hardware configuration was performed with establishment of hierarchical control as a major driver. This resulted in the functional allocation shown in Figure 2.

3.0 THE SAFETIES COMMUNICATION INTERFACE (SCI)

This implementation concerned itself primarily with the application layer of the ISO Basic Reference Model of Open Systems Interconnection. A summary of the desired capabilities are given in Table 1. Table 2 gives a list of the different kinds of message types used in the SAFTIES demonstration. There are many other types of messages that could be used in a full-fledged distributed system. The goal of the SCI protocol is to allow expert systems to "plug in" to SAFTIES with minimal alterations.

The Art Implementation

The lower level expert systems, running in ART on the Symbolics 3640, receive SCI data as follows: between every rule that fires, the serial line is read to see if data is present; if so, it reads the data and asserts it as a fact; if not, processing continues as normal.

The Pascal Implementation

The Executive, running in Pascal on the Sperry IT/PC, receives SCI data as follows: whenever the screen is idle (waiting for input from the

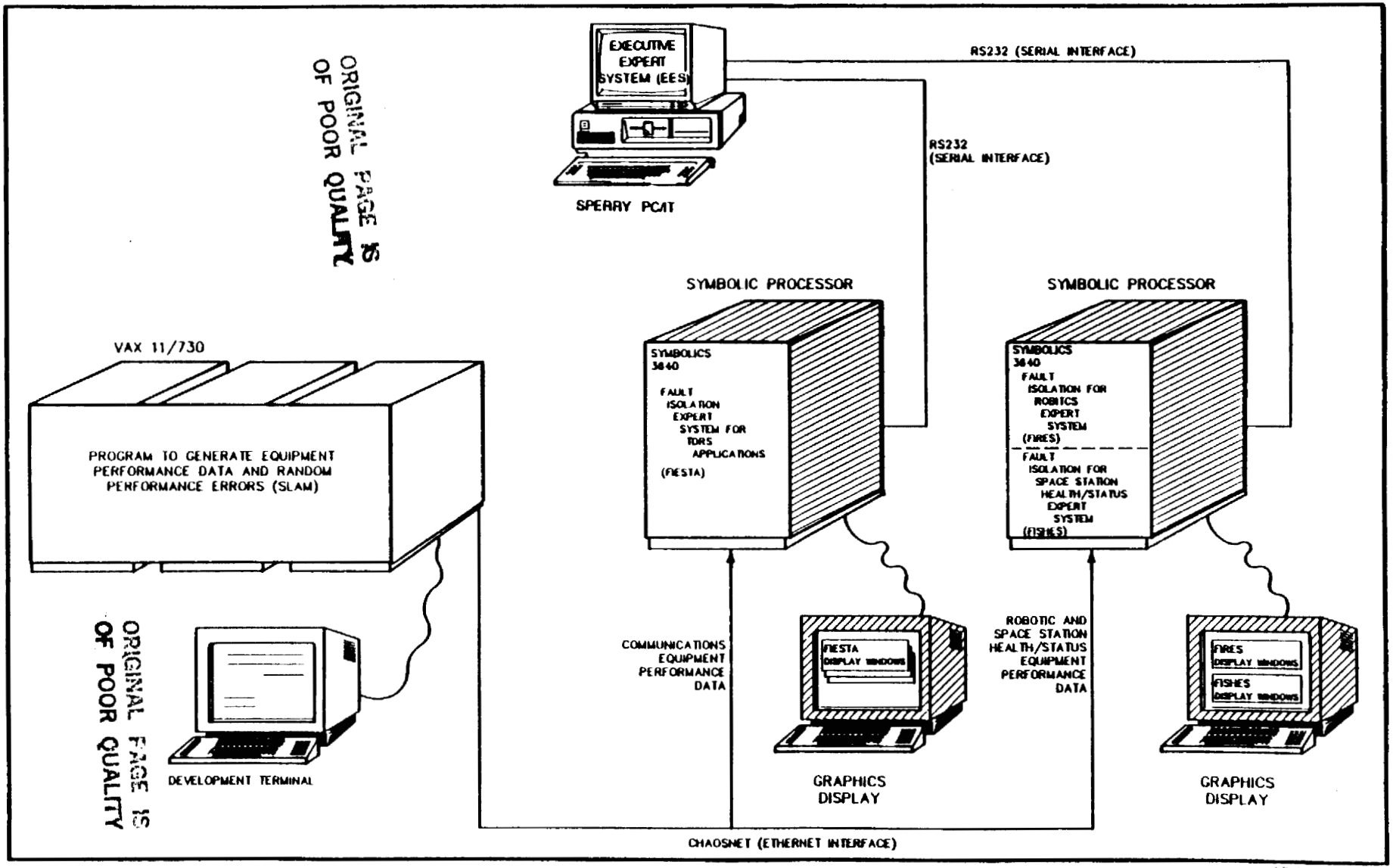


FIGURE 2: FUNCTIONAL OVERVIEW OF SAFTIES

TABLE 1

SUMMARY OF SAFTIES COMMUNICATIONS INTERFACE (SCI)

- A COMMUNICATIONS PROTOCOL THAT ALLOWS EXPERT SYSTEMS TO COMMUNICATE WITH THE EXECUTIVE
- ALL SCI MESSAGES HAVE A HEADER IN THE FOLLOWING FORMAT:
  - PRIORITY
  - MESSAGE TYPE CODE
  - SENDING SYSTEM
  - RECEIVING SYSTEM
  - ACTIVITY/EVENT CODE
  - SUB-CODE (RESOURCE ID, REQUESTED DATA PARAMETER ID)
- FOLLOWING THE HEADER IS A PARAMETER LIST GIVING INFORMATION SPECIFIC TO THE MESSAGE.

TABLE 2

SCI MESSAGE TYPES

MESSAGE TYPE	DEFINITION	PARAMETERS
101	REQUEST FOR DATA	NONE
102	SCHEDULES	BEGIN TIME, [END TIME], [REPETITIONS]
103	COMMAND	COMMAND CODE, COMMAND PARAMETERS
201	PERFORMANCE DATA	VALUE
202	CONFIGURATION DATA	CONFIGURATION LIST
203	FAULT ISOLATION DATA	CURRENT VALUE, ESTIMATED CAUSE OF FAULT
204	EMERGENCY ALARM	CURRENT VALUE, [ESTIMATED CAUSE OF FAULT]
205	EVENT STATUS	EVENT CODE [IN SERVICE, PRESERVICE, ETC.]
900	MESSAGE IN ERROR	[RETURN ENTIRE MESSAGE IN ERROR]

user) and before the keyboard is checked to see if an input is received, the serial "com" ports 1 and 2 are checked to see if data is present; if so, it reads it. This is done on the DOS BIOS level using interrupts and status registers. If it reads the serial line and data is present, it analyzes the data read to make sure an entire SCI record is read; if not, it issues consecutive reads until the entire SCI record is received. If no data is present on either serial line, it checks the keyboard for activity.

4.0 TESTBED CONFIGURATION COMPONENTS

This section presents each of the testbed configuration components. In Section 4.1 the simulation approach to data generation is described. The three independent expert systems are presented in Section 4.2. Section 4.3 then describes the executive expert system.

4.1 SIMULATION DRIVERS

Drivers are used to generate performance data and simulate faults. The simulation driver for each fault isolation expert system is written in SLAM (Simulation Language for Alternative Modeling); a summary of the drivers is given in Table 3. There are separate drivers for each expert system. The variety and quantity of equipment for which data is generated within each simulator is limited only by physical resources of the host hardware. A nominal value, high and low tolerance values, equipment code, and station (location of the equipment) are assigned to each data generating equipment. Every 5 seconds a "reading" is taken; under normal circumstances, the "reading" is a random number with a triangular distribution (mean as the normal value, normal tolerance as low value, high tolerance as high value). At random time periods, however, these values will fall out of range. How the numbers fall out of range is determined by the type of error.

TABLE 3

SUMMARY OF SIMULATION DRIVERS

- WRITTEN IN SLAM (SIMULATION LANGUAGE FOR ALTERNATIVE MODELING)

- A NOMINAL VALUE, HIGH TOLERANCE AND LOW TOLERANCE IS INITIALLY ASSIGNED TO EACH RESOURCE

- EVERY 5 SECONDS DATA IS GENERATED FROM ALL RESOURCES SUPPORTING SERVICES AND EVENTS; FOR EACH RESOURCE, IT IS A RANDOM NUMBER TRIANGULARLY DISTRIBUTED BASED ON THAT RESOURCE'S NOMINAL, HIGH TOLERANCE AND LOW TOLERANCE VALUES

- ERRORS INVOLVING RESOURCE PERFORMANCE ARE GENERATED BOTH RANDOMLY AND AT PRESET TIMES; WHEN AN ERROR HAS OCCURRED, THAT RESOURCE VALUE WILL:
  - EXCEED HIGH TOLERANCE
    - -- SLOWLY
    - -- QUICKLY
    - -- IMMEDIATELY
    - -- DELAYED
  - EXCEED LOW TOLERANCE
    - -- SLOWLY
    - -- QUICKLY
    - -- IMMEDIATELY
    - -- DELAYED
  - START AND REMAIN AT AN OUT-OF-RANGE VALUE

  ONCE IN ERROR, THE ERROR REMAINS FOR A RANDOM TIME PERIOD.

4.2 THE INDEPENDENT EXPERT SYSTEMS

In this section the expert system nodes which occur in the SAFTIES hierarchy are presented. These consist of the three base level expert systems, FIRES (robotics), FISHES (environment), FIESTA (communications) and the top level EXECUTIVE EXPERT SYSTEM (EES).

4.2.1 FIRES (Fault Isolation for Robot Expert System)

4.2.1.1 The FIRES System. An overview of FIRES requirements is presented in Table 4. The FIRES expert system node receives performance/status data from a robotics system. For purposes of the testbed prototyping a Scara-type robot arm (4 degrees of freedom) was selected for simulation. Readings on various aspects of the arm operations are reported to FIRES every five seconds. All readings by FIRES are assumed to have a nominal value and an accepted range of operation as shown in Table 5.

The FIRES system employs a data-driven approach to diagnosis. When performance values which fall outside the acceptable range are encountered the relevant detection rules are activated. The detection rules then report the anomaly and summarize the information in a manner to support further diagnostic processing.

For the FIRES system, Fault Diagnosis has been implemented using a pattern-matching technique in a production system paradigm. This implementation illustrates the technique of inferring fault causes directly from specific patterns of anomalies.

For the demonstration prototype, fault isolation was followed by sending notification to the executive.

FIRES is implemented on a SYMBOLICS 3640 using the Automated Reasoning Tool (ART) from INFERENCE Corporation. ART is an Expert System (ES)

TABLE 4

FIRES REQUIREMENTS SUMMARY

FAULT ISOLATION FOR ROBOTICS
EXPERT SYSTEMS (FIRES)

- FUNCTIONS
  - ISOLATE FAULTS IN THE ROBOT OPERATING SYSTEM
  - INFORM EXECUTIVE OF MISSION CRITICAL FAULTS (ALARMS)
- INPUTS
  - EVENT SCHEDULE (CONTAINING ROBOT RESOURCE CONFIGURATION DATA)
  - COMMAND (FROM EXECUTIVE)
  - PERFORMANCE DATA (FROM ROBOT RESOURCES)
    - -- JOINT POSITION AND/OR TORQUE
    - -- DRIVE TORQUE
    - -- GRIPPER (TACTILE SENSOR, POSITION, FORCE)
    - -- SENSOR DATA (VISION, FIBER OPTICS, THERMAL, PRESSURE, STRAIN GAUGE)
- OUTPUTS
  - FI DIAGNOSTIC DATA
  - EVENT STATUS, AS REQUESTED
  - RESOURCE PERFORMANCE AND CONFIGURATION DATA, AS REQUESTED
- HARDWARE
  - SYMBOLICS 3640 (WITH OPTION TO MOVE TO MICROVAX II AI STATION)
  - COAXIAL AND SERIAL CABLES
- COMMUNICATIONS
  - RS232 SERIAL INTERFACE TO SPERRY IT
  - ETHERNET (CHAOSNET) CONNECTION TO VAX
- SOFTWARE
  - ART
  - LISP

TABLE 5

FIRES MONITORING PARAMETERS
(NOMINAL VALUES AND ACCEPTABLE RANGES)

	NOMINAL VALUE	LOW TOLERANCE	HIGH TOLERANCE	EQUIPMENT CODE	STATION/ LOCATION CODE
ROBOT-BATTERY	18.0	12.0	24.0	E01	01
SERVO TORQUE	10.0	0.0	20.0	E02	01
ARM VELOCITY	0.0	-18.0	18.0	E03	02
GRIP-FORCE	4.0	3.0	5.0	E04	03
GRIP-SENSE	1.0	0.0	1.0	E05	03

Development Tool which supports rapid prototyping. By providing basic ES constructs, it allows expert system development to concentrate on the knowledge engineering aspects of the task.

4.2.1.2 FIRES Simulation Driver. The FIRES driver simulates 5 performance monitoring pieces of equipment, obtaining readings of battery power, servo motor torque, arm velocity, gripper force and gripper presence sensor.

There are 6 possible errors which may occur:

1) Arm locked in place (collision with obstacle)
2) Incorrect gripper (possibly incorrect configuration in schedule)
3) Dropped object
4) Arm broken (collision with obstacle (wall) at high velocity)
5) Arm payload exceeded (picking up an object too heavy for the robot)
6) Battery low on power

These errors occur randomly, approximately one error per minute. At the end of the error, which lasts approximately 30 seconds, the readings return to acceptable range, indicating a correction has occurred.

4.2.2 FISHES (Fault Isolation for Space Station Health/Status Expert System)

4.2.2.1 The FISHES System. The FISHES expert system node receives performance data which is associated with health and status for an assigned area of the spacecraft. Requirements and monitored parameters similar to those presented for the FIRES system were developed for FISHES.

The data-driven approach employed in FISHES was also applied for detection of anomalies in the FIRES domain. The same form of pattern-

matching strategy employed in FISHES was adopted for FIRES. This logical replication allowed for the timely availability of the FIRES node during development and implementation of the testbed.

There was also physical replication since FISHES has also been implemented on a SYMBOLICS 3640 using ART.

4.2.2.2 FISHES Simulation Driver. The FISHES driver simulates 6 performance monitoring pieces of equipment, obtaining readings of cabin temperature, cabin pressure, percent oxygen, percent particulates, and the spacecraft power consumption and power generation.

There are four possible errors which can occur:

1) Hole in spacecraft (broken seal, puncture due to poor docking or meteor, etc.)
2) Fire in cabin
3) Faulty air filter
4) Faulty pressure reading (indicating a malfunction in the monitoring equipment itself)

These errors occur randomly, approximately one every minute, and last for approximately 30 seconds. When the error is completed, the readings return to within the acceptable range.

4.2.3 FIESTA (Fault Isolation Expert System for TDRSS Applications)

4.2.3.1 The FIESTA System. The specific requirements which are relevant to the SAFTIES project are summarized in Table 6.

FIESTA is an evolving prototype expert system which has been developed under the auspices of NASA/GSFC Code 532-1.* FIESTA is designed to isolate faults in a communication network. The requirements guiding the

* Work was performed under the direction of Bendix Field Engineering Corporation, NASA contract NAS5-27600.

MIS30GTAB

TABLE 6

FIESTA REQUIREMENTS SUMMARY

(SAFTIES RELATED)

FAULT ISOLATION EXPERT SYSTEM FOR
TDRSS APPLICATIONS (FIESTA)

- FUNCTIONS
  - RECEIVE AND MONITOR SPACE NETWORK STATUS INFORMATION
  - ISOLATE FAULTS WHICH OCCUR ON THE NETWORK
  - INFORM EES OF ISOLATED FAULTS
- INPUTS
  - STATUS INFORMATION IN THE FORM OF NASA'S CURRENT HIGH SPEED MESSAGES
- OUTPUTS
  - HIGH LEVEL FAULT ISOLATION DIAGNOSTIC DATA
  - ALARMS
- HARDWARE
  - SYMBOLICS 3640
- COMMUNICATIONS
  - RS232 SERIAL INTERFACE TO SPERRY IT
  - ETHERNET (CHAOSNET) CONNECTION TO VAX
- SOFTWARE
  - ART
  - LISP

FIESTA development are contained in "FIESTA Project Development Folder/ Volume II: Prototype Requirements Specification" (STI/E-25190A, 17 December 1985). These requirements define FIESTA operating as a standalone testbed. FIESTA employs a highly developed Axiomatic/ Hypothetical approach to fault isolation. Inclusion of FIESTA within the distributed hierarchy allows an extensive demonstration of this methodology.

FIESTA was incorporated in the distributed hierarchy via the SCI interface described earlier. This served to validate the expansion capabilities afforded by the SCI protocol.

4.2.3.2 FIESTA Simulation Driver. The FIESTA driver simulates 12 performance monitoring pieces of equipment. Errors in any service (KSAR, SSAR or SSAF) occur as follows:

- The signal strength starts to degrade
- When the signal strength falls below 3, then the locks go to Ø
- The number of frames in lock starts to fall
- When the frames in lock equal Ø, then the data present goes to Ø.

Again, the errors occur randomly for a random amount of time. At the end of the error, the readings return to acceptable ranges.

4.3 THE EXECUTIVE EXPERT SYSTEM (EES)

An overview of the requirements established for the Executive Expert System (EES) are presented in Table 7.

4.3.1 The Domain of EES

EES receives high-level fault isolation data from lower-level expert systems. In addition, EES can request operational parameters as needed. All data received is in the SCI format. Upon receipt of notification of a fault, the Executive determines a corrective action

TABLE 7

EXECUTIVE EXPERT SYSTEM (EES)
REQUIREMENTS SUMMARY

- DESCRIPTION
  - EXPERT SYSTEM SHELL DESIGNED TO AID ASTRONAUT/USER IN SCHEDULING, COORDINATING, AND CONTROLLING OF DISTRIBUTED EXPERT SYSTEMS IN THE SPACE STATION
- FUNCTIONS
  - PROVIDE HUMAN INTERFACE/WORKSTATION TO
    - -- RETRIEVE INFORMATION FROM EXPERT SYSTEMS
    - -- COMMAND
    - -- EFFECT FAULT ISOLATION PROCESSING
  - RECEIVE FAULT ISOLATION DIAGNOSIS FROM DISTRIBUTED FAULT ISOLATION EXPERT SYSTEMS AND SCHEDULE CORRECTIVE EVENT(S)
  - ALLOCATE ASSIGNED RESOURCES, RESOLVING CONFLICTS AS THEY ARISE
  - SCHEDULE EVENTS BASED ON PRIORITIES, NEED, AND REQUESTS
- INPUTS
  - SPACE STATION EVENT SCHEDULES (FROM NASA OR SCHEDULER EXPERT SYSTEM)
  - HIGH LEVEL PERFORMANCE DATA
  - FAULT ISOLATION DIAGNOSTIC DATA
  - EVENT STATUS
  - HUMAN EXPERT INPUT AND OVERRIDES
- OUTPUTS
  - COMMAND (TO LOWER EXPERT SYSTEMS)
  - REQUEST FOR INFORMATION
  - EVENT SCHEDULES AND PRIORITIES
  - NOTIFICATION TO GROUND OF NONCORRECTABLE ERRORS
  - HISTORY FILE
- HARDWARE
  - SPERRY IT
    - -- 7.14 MHz CLOCK
    - -- COLOR GRAPHICS MONITOR
  - SERIAL CABLES
- COMMUNICATIONS
  - 2RS232 SERIAL INTERFACE TO SYMBOLICS 3640
- SOFTWARE
  - EXPERT SYSTEM WRITTEN IN TURBO PASCAL
  - SYNCHRONOUS COMMUNICATIONS ON BOTH SERIAL LINES ACHIEVED

and schedules it, based on the current operating state of the system using a "least cost" algorithm.

4.3.2 EES User's Operational Description

The console normally displays the main menu (shown in Figure 3).

When this and most all other screens are displayed, EES is ready to accept input from the user or receive data from lower level expert systems.

For example, if item 2 (FIRES details) is selected from the main menu, the FIRES menu is displayed (see Figure 4) which allows FIRES specific options to be selected.

Whenever a fault is isolated by one of the lower level expert systems and sent to EES, the operator is alerted to an anomolous situation. Notification is accomplished via an alarm window which pops up (non-destructively) in whatever window is currently active. Figure 5 provides an illustration of this feature.

5.0 SUMMARY AND CONCLUSIONS

The major thrust of this investigation was the establishment of the framework for a test bed capable of supporting an investigation of distributed expert system processing with hierarchically organized domains of responsibility and control. As described in this paper, the framework has been established.

The current configuration consists of an executive system which coordinates the activities of three individual expert systems at the next lower hierarchical level. The secondary level expert systems have separate domains and provide status summaries on their individual areas of responsibility to the executive. Although the domains are separate, the functions are similar, namely performance monitoring, fault detection and fault isolation.

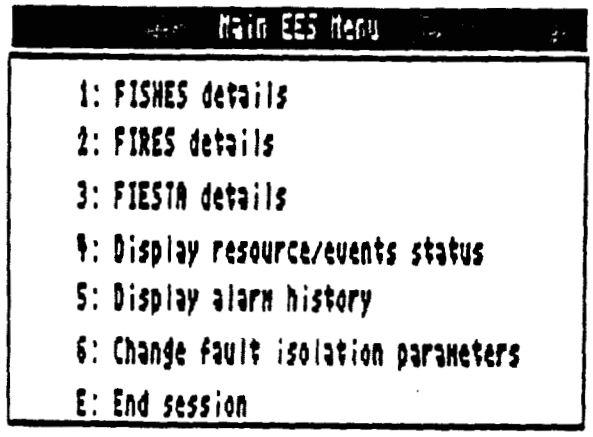


FIGURE 3: MAIN MENU

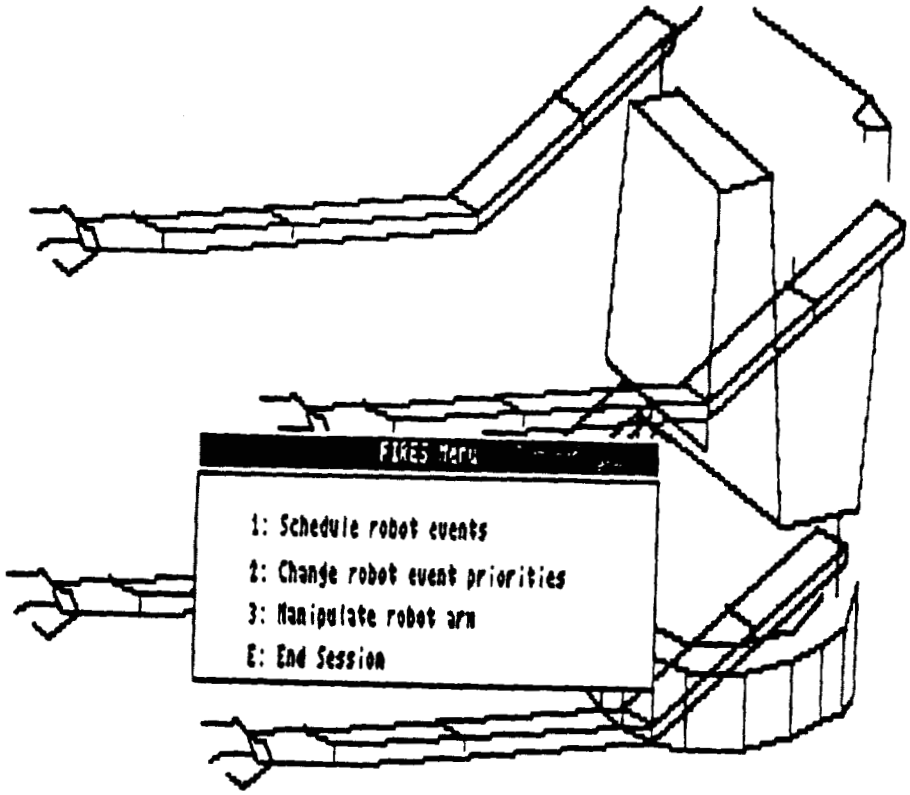


FIGURE 4: FIRES MENU

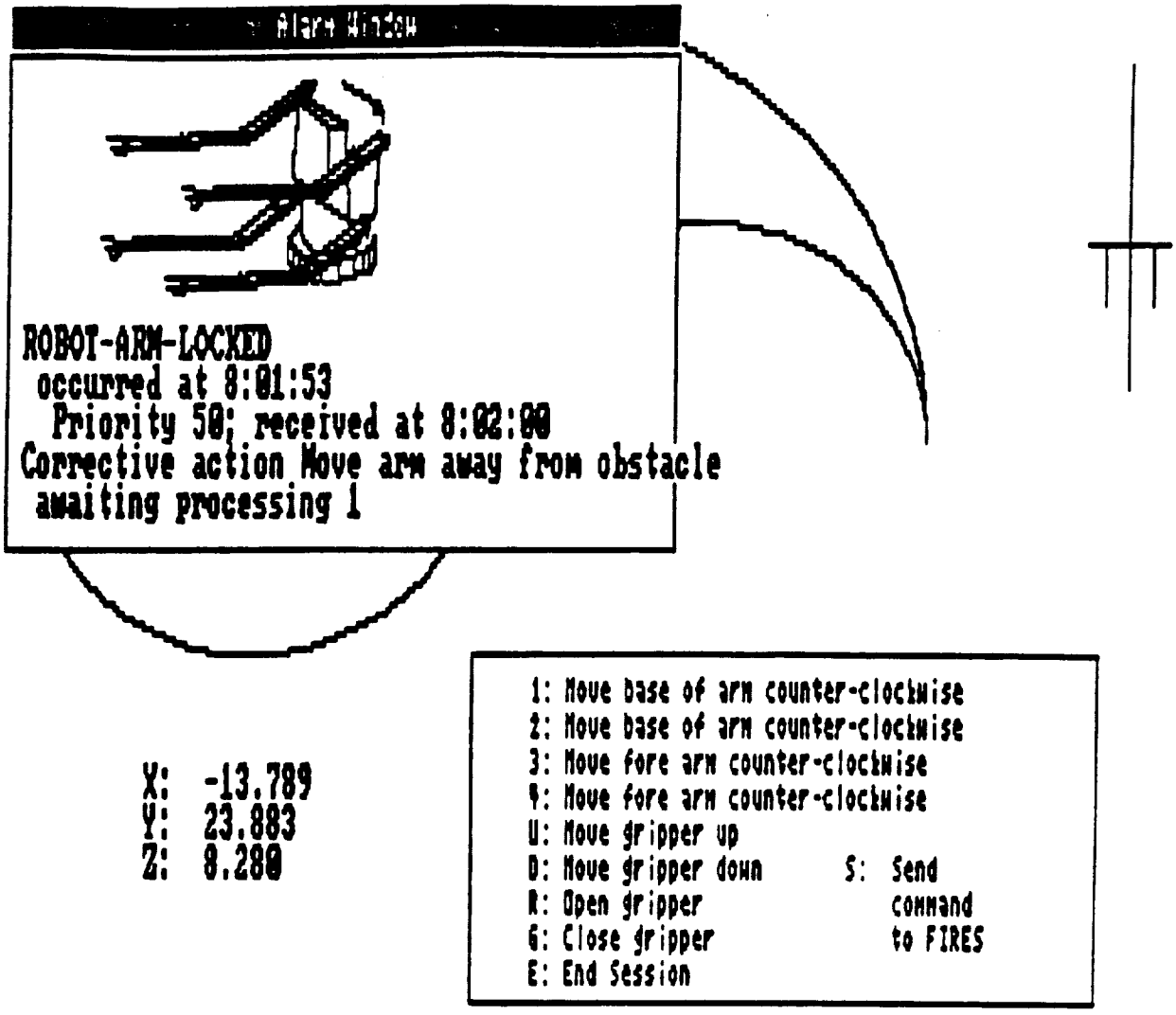


FIGURE 5: ALARM WINDOW NOTIFICATION (ROBOT ARM MANIPULATION WINDOW ACTIVE)

A major conclusion reached during this investigation is that a network consisting of multiple expert systems with hierarchically distributed control can be readily established. The feasibility of such a configuration was verified by establishing a operational test bed exhibiting these characteristics.

The simulation data was correctly monitored by the separate fault isolation system; nominal conditions being (implicitly) noted and non-nominal being detected. A review of the data indicated that the anomalies were being correctly identified. The executive demonstrated its ability to co-ordinate the resource of independent systems and correctly assign available resources to achieve problem resolution.

Another major conclusion (also implemented in the demonstration test bed environment) involved techniques for implementing the hierarchical control. It was shown that conventional software engineering techniques in the area of communication protocol, integrated with an expert system executive process, was capable of supporting the candidate architecture. The SCI was also shown to be capable of supporting integration of an existing expert system (FIESTA) into the hierarchical structure.

Many open questions remain in the area of di.tributed processing and control. The initial test bed structure provides an environment to support investigations is this area.

ACKNOWLEDGMENTS

The results reported in this papaer are drawn from "FINAL Report Phase I Study: Fault Processing using Axiomatic/Hypothetical Methods in a Multi/Level Expert System Environment", TR860155, 26 September 1986 by Steve Miksell, Sue Cofer and Edwin Zakrzewski.

That work was supported under an SBIR (Small Business Innovative Research) contract from NASA; Contract No. NAS5-29280. Research was conducted under the guidance of NASA/GSFC code 735. Their support and encouragement were appreciated and are acknowledged.

AUTOMATION OF SPACECRAFT CONTROL CENTERS

Robert Dutilly
Goddard Space Flight Center

ABSTRACT

The objective of this paper is to describe the further automation of the Payload Operations Control Centers, specifically the Mission Operations Room, by using a series of expert systems interconnected together. The feasibility of using expert systems in the Mission Operations Room is presently being determined. The expert system under development is called the Communications Link Expert Assistance Resource (CLEAR) project. It is the first control center expert system being designed and implemented at Goddard. It will demonstrate the feasibility and practicality of expert systems in a real-time control center environment. This paper has a twofold purpose. The first is to briefly describe the present effort of the CLEAR expert system under development. The second is to describe how a series of interacting expert systems could be developed to almost totally automate the Mission Operations Room within the control center. This paper will describe how these expert systems would be put together and what functions they could perform in the control center. These efforts will provide a great deal of applicability toward the automation of the space station.

Keywords: Automation, Control Center, Expert System, Real-time

Purpose

The purpose of this paper is to describe the means by which automation software, specifically expert systems, can be developed for the spacecraft control centers at Goddard Space Flight Center. These control centers provide an excellent environment to test the feasibility and practicality of real-time expert systems. This paper will detail the control center environment, what work has already been started and how a series of interconnected expert systems will work together. Finally, summarizing this concept of expert systems in this real-time environment will provide the expertise necessary for both the ground system and onboard system of the space station.

Environment

There are two types of spacecraft control centers at Goddard. The dedicated control center, such as the Space Telescope, which handles only one mission and the

Multisatellite Operations Control Center (MSOCC) which supports a number of simultaneous missions. This automation effort that is proposed would work equally as well in either type of control center. The expert system presently under development will be used in the MSOCC for the Cosmic Background Explorer (COBE). This effort will be discussed later in this paper.

The control center environment is a combination of hardware and software. The hardware for the control center is made up of a front end computer system to frame synchronize the telemetry data; the main computer system which performs all of the processing on the telemetry data and commanding the satellite; and the workstations located in the Mission Operations Room (MOR). This main computer system is commonly called the Application Processor (AP). While the front end computer is called the Telemetry and Command (TAC) system. The function of the MOR is to provide a location where the Flight Operations Team works to monitor and control the spacecraft. The MOR contains workstations and stripchart recorders to display the data that has arrived at the AP. The software developed for the mission resides on all of the hardware systems used to support the satellite. Figure 1 provides a descriptive layout of the spacecraft to ground system.

There are several very important aspects that must be accounted for in this type of environment.

a. The control center receives a real-time flow of asynchronous data.

b. There is a large amount of data to process.

c. There must be a continuous performance assessment of the spacecraft and all of its subsystems in real-time.

d. The expert system must be on a separate computer system and receive all of the data electronically in real-time from the AP. The primary reason for a separate system is to avoid interfering with the existing real-time software and operations.

e. The expert system must be able to interact with the human analyst in real-time while receiving the data.

f. A time dependency is critical to this large amount of uncertain data.

What this last point means is that the expert systems are required to take into account what the short and long term trends are to the data. Consequently, the system must construct an internal history of the data to develop a trend analysis. This trend analysis will indicate how the spacecraft or instruments will be performing in the near

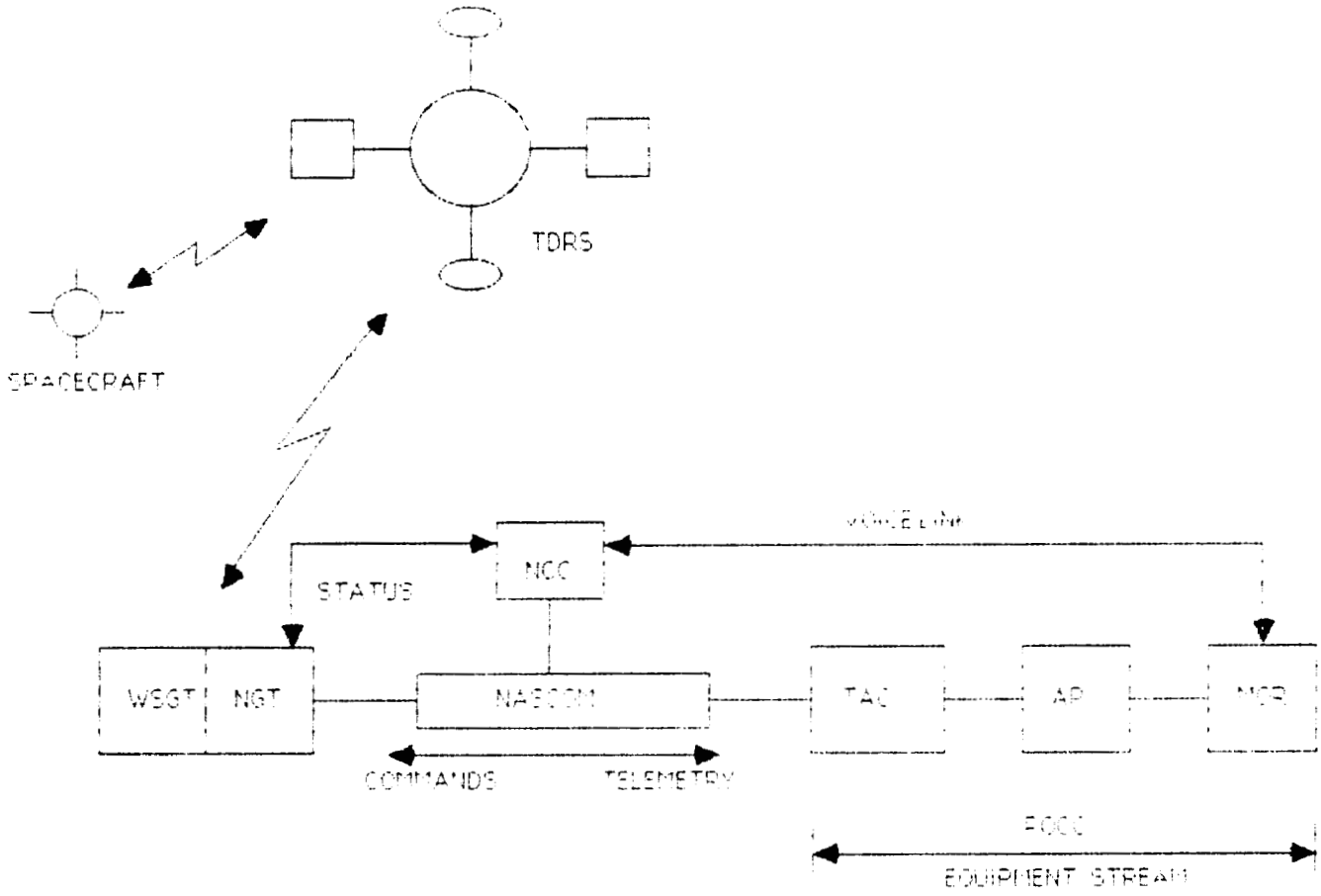


Figure 1 Descriptive Layout of a Satellite to Ground System

term. The expert systems must also be able to relate the data from different areas. This capability will provide a multi-focus analysis of the incoming information. This is similar in nature to the human information gathering capabilities in which pieces of disparate information forms a whole concept. This would be the meta-knowledge required to develop the generalities for the system to use in determining problems with the spacecraft. This meta-knowledge will provide the expert system with the overall concepts on how the spacecraft and its subsystems should operate together. It will provide some capabilities for the expert system to draw conclusions about the health and safety of the spacecraft. Finally for this type of environment, there are three types of systems that can be developed.

a. Passive system -- This expert system acts only as a recommender of actions for the spacecraft analyst. This system does not take any actions and is not provided that capability. This is the type of system that must initially be developed for control centers.

b. Active system -- This type of expert system provides all of the capabilities of the passive system. It also can be directed by the spacecraft analyst to actually take positive action to correct problems. After expert systems have proven their capabilities to find and solve problems, then an active system may prove to be worthwhile.

c. Autonomous system -- As this term indicates, this system would perform all actions in finding, solving and correcting problems. It would operate independently of the spacecraft analyst and would only present the problem and actions taken both during and immediately after the event.

As previously indicated, the first expert systems to be installed in a control center will be of the passive type. All actions will be taken by the analyst on a separate workstation in the MOR. Eventually, after expert systems have been proven, an active system can be implemented. The autonomous system will take a great deal of time and effort to develop. It must not only be proven to be safe for the spacecraft but also be accepted by the operations personnel.

Present Development

The expert system that is presently being developed is called the Communications Link Expert Assistance Resource (CLEAR). CLEAR will operate in the Mission Operations Room during real-time passes of the COBE spacecraft. This is the first expert system to be put into an operational control center environment at Goddard. The purpose of this system will be to monitor and help solve communications problems between the COBE spacecraft and the Tracking and Data Relay Satellite (TDRS). CLEAR is a passive system that advises

the spacecraft analyst when an event has occurred which indicates the COBE-TDRS communications link is degrading or failed. The analyst is provided with possible solutions to the problem. It is the analyst who decides what should be done and uses a voice link with NCC and the workstation to solve the problem. A detailed description of CLEAR is provided in another paper.

Proposed Development

Now to describe how a series of real-time expert systems could be installed into the MOR. These expert systems would be executing on a separate system from the AP. Figure 2 provides a functional layout on how this system would operate. The basic concept is that the AP would be providing a continuous stream of data to the expert systems during the real-time pass of the spacecraft. The main classes of data that would be provided to the expert systems are:

a. The spacecraft subsystem data such as attitude, power or electrical.

b. The spacecraft instrument data for health and operability.

c. Any communications information from the Network Control Center.

The control center is concerned with the health and safety of the spacecraft and the onboard instrumentation. So the specific purpose of this system is to monitor the data sent to it by the AP and to assist the spacecraft analyst in solving problems that occur. This indicates to you that the system must initially be passive so that the human remains in total control. The main requirements of this system are:

a. It must be designed in a modular fashion so that lower level expert systems may be added or deleted easily.

b. Each of the systems must be able to handle multiple faults.

c. Each of the systems must be able to handle any reconfigurations of the spacecraft subsystems.

d. The systems should be able to develop estimates on when a component, instrument or subsystem is failing based on trend analysis.

e. This system must be able to accept information into the data base from the spacecraft analyst via the user interface.

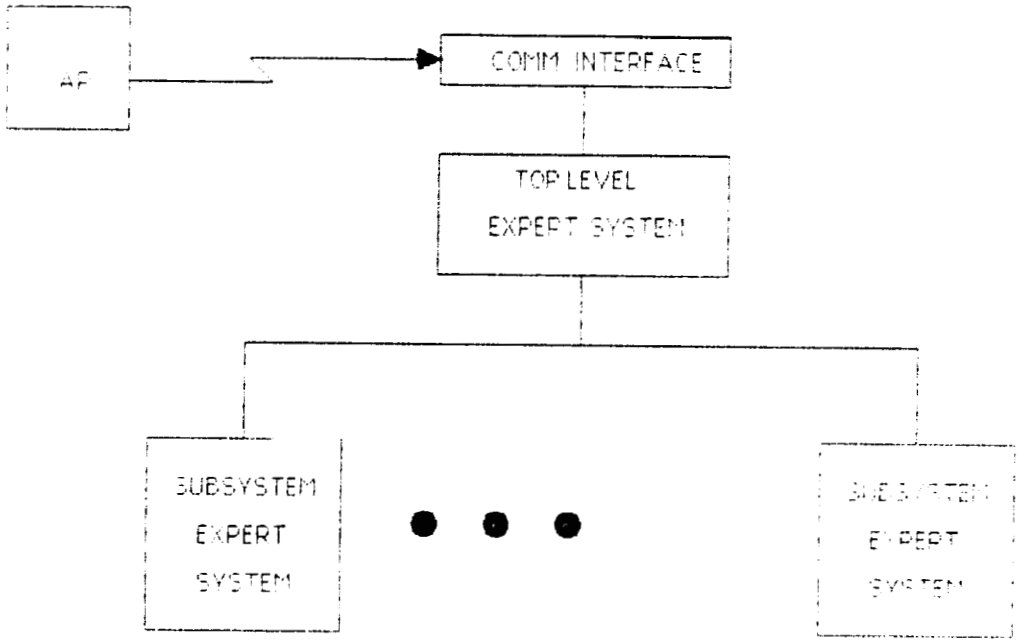


Figure 2 Functional Layout of the Total System

The software architecture for this integrated system is shown in figure 3. What stands out immediately is the main inference engine providing messages to a blackboard and the lower level expert systems obtaining those messages addressed to them. Each expert system has its own Knowledge Base provided for that subsystem. Also available to all the expert systems are the Data Base, the Data Tables and the Logging Facility. The Data Base contains the information about the spacecraft, the instruments, the different telemetry formats and all of the information about the data itself, for example, the limits. The Data Tables contain all of the incoming data, similar in nature to the telemetry table in the AP. As shown in figure 3, the blackboard shown will be a circular file with the main system notifying each subsystem of a problem. The subsystems periodically "wake up" on a short time span and check the blackboard for a message. If a subsystem locates a message, it will read and erase the entry on the blackboard. That subsystem will then access the data table looking for its specific information and use its detailed knowledge base to try to determine the problem or event that occurred. It will then access the user interface to provide this information to the spacecraft analyst. The reason for this hierarchy is to minimize the number of subsystem expert systems from competing for the computer's resources.

Several other points need discussing at this time. This integrated system will require meta-knowledge; the ability to solve problems by combining the data and knowledge from several sources. This is one of the most difficult areas to develop since the system designer must be able to glean from the spacecraft analyst all of the factors involved in making critical decisions. It is not enough to have rules checked based on a telemetry point. The system must be given a "higher" knowledge about how the spacecraft and its subsystems operate together and also how to "reason" about the telemetry data. This leads into the second point on how to work with the uncertain data. The most popular ways today is using probability techniques and fuzzy logic theory to help assist the inference engine in determining if a problem exists. The third point is the testing of this system. With the hierarchy of systems as designed, the testing will be able to focus in on each part. While not perfect, at least this will provide a structured technique that is in use today for testing large systems. Finally, the most important point is to remember to keep the human "in the loop". The human will be able to assist the expert system and correct any false conclusion that had been reached.

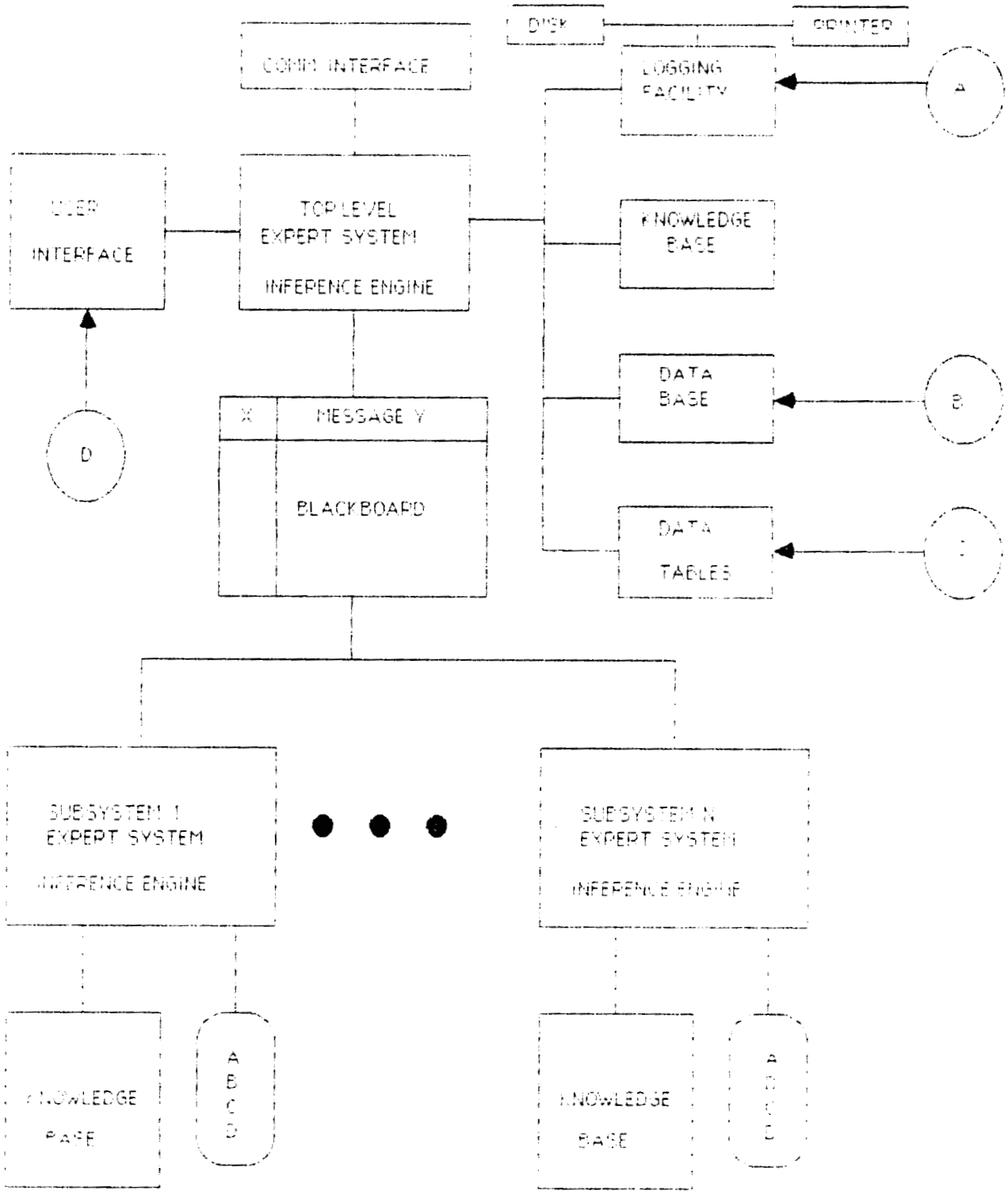


Figure 3. Architecture of the Toron system

Summary

As was indicated at the beginning of this paper, chaining together expert systems to form one integrated system will automate many of the functions of a control center. This will also reduce the complexity of analyzing all of the data by the spacecraft analyst. If this concept proves to be both feasible and practical and can actually be implemented to work properly in the real time environment of a control center, the use of expert systems would have definite applicability for space station support. The control center could serve as the initial test environment on how expert systems would be developed and installed in the space station and any attendant ground control system.

GLOSSARY

AP	Applications Processor; The computer system that sends command data to the spacecraft and receives the telemetry data, decommutates the data and sends that data to the MOR.
CLEAR	Communications Link Expert Assistance Resource; This is the first expert system being developed for use in the control center. It will be used for the COBE spacecraft.
COBE	Cosmic Background Explorer; This spacecraft will be the the first at Goddard to use an expert system in the control center.
Control Center	The hardware and software real time environment that controls the spacecraft and monitors its health and safety.
MOR	Mission Operations Room; That part of the control center in which the Flight Operations Team works. It contains the workstations, displays and strip chart recorders to display the data from the AP.
NCC	Network Control Center; This facility provides the management for allocating and regulating network resources to support network users.
TAC	Telemetry and Command System; The front end computer system that frame synchronizes and time tags the incoming telemetry data before sending it to the AP. It receives the command data from the AP and sends it out to NASCOM.
TDRS	Tracking and Data Relay Satellite; This is the communications satellite which communicates with the spacecraft and White Sands.

Section C

Data Processing/Analysis Expert Systems

SPACELAB DATA PROCESSING FACILITY (SLDPF)

QUALITY ASSURANCE EXPERT SYSTEMS

DEVELOPMENT

Angelita C. Kelly (NASA/GSFC 564)
Lisa Basile (NASA/GSFC 564)
Troy Ames (NASA/GSFC 522)
Janice Watson (Lockheed)
William Dallam (Lockheed)

1987 GODDARD CONFERENCE ON SPACE APPLICATIONS
OF ARTIFICIAL INTELLIGENCE (AI) AND ROBOTICS

APRIL 1987

SPACELAB DATA PROCESSING FACILITY (SLPDF) QUALITY ASSURANCE EXPERT SYSTEMS DEVELOPMENT

Angelita C. Kelly (NASA)
Lisa Basile (NASA)
Troy Ames (NASA)

Janice Watson (Lockheed)
William Dallam (Lockheed)

ABSTRACT

The Spacelab Data Processing Facility (SLDPF) has developed expert system prototypes to aid in the performance of the quality assurance function of Spacelab and/or Attached Shuttle Payloads (ASP) processed telemetry data. The SLDPF functions include the capturing, quality monitoring, processing, accounting, and forwarding of data from Spacelab and ASP missions to various user facilities. The SLDPF consists of two functional elements: the Spacelab Input Processing System (SIPS) and the Spacelab Output Processing System (SOPS). The two expert system prototypes were designed to determine their feasibility and potential in the quality assurance of processed telemetry data. The SIPS expert system, Knowledge System Prototype, (KSP), uses an IBM PC/AT with the commercial expert system shell OPS5+. Extraction of knowledge from SIPS experts was implemented emulating the duties of quality assurance analysts. In an interactive mode, an analyst responds to queries resulting in instructions and decisions governing the reprocessing, releasing or further analysis/troubleshooting of data. Released data is forwarded for further processing on the SOPS Sperry 1100/82. The data are edited, time ordered with overlapping data removed, decommutated, and quality checked before shipment. The SOPS QA analysts isolate problems and select the appropriate action: either accept the data or request the data to be reprocessed. The SOPS expert system emulates this process by using an expert system shell, CLIPS, and the Macintosh personal computer. To date, these prototypes indicate potential beneficial results; e,g., increase analyst productivity, decrease the burden of tedious analysis, provide consistent evaluations of data, provide concise historical records, provide training for new analysts, and expedite the operational retraining of reassigned Spacelab analysts. The logic implemented in the prototypes, the limitations of the personal computers utilized, and the degree of accessibility to input data have led to an operational configuration. This configuration is currently under development and on completion will enhance the efficiency, both in time and quality, of releasing Spacelab/ASP data.

1.0 INTRODUCTON

Expert system applications in the Information Processing Division were first considered for their potential to expedite the SLDPF operations, in particular, the quality assurance (QA) and data accounting (DA) analyst functions of both the Spacelab Input Processing System (SIPS) and the Spacelab Output Processing System (SOPS). The QA/DA task is often demanding and tediously repetitive. The objective of the operational expert systems is to assist the analyst by making decisions and suggesting logical analysis paths based on given data quality information.

The expert system application to assist the QA function of SIPS was assigned to Lockheed under the direction of Code 564; Lockheed Quality Assurance Analysts (QAAs) serve as experts, and system engineers perform the knowledge engineering, coding and project management. The application to assist the QA/DA function of SOPS was tasked to Code 522, Code 564 and Lockheed; Lockheed QA analysts serve as experts, Code 522 performs the knowledge engineering and coding, and Code 564 provides the project management. Code 564 SLDPF personnel provide the technical and overall guidance of the two projects.

1.1 Implementation

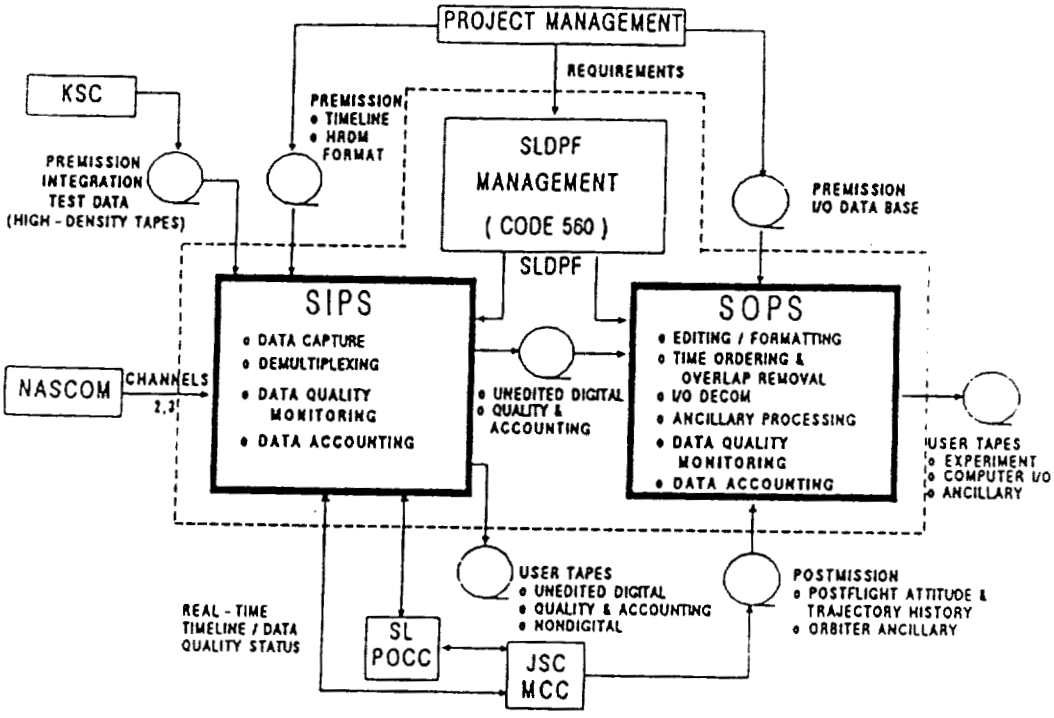
The strategy formulated to accomplish the prototypes was to use commercial expert system shells, code the QA knowledge bases within the shells and implement the shells on personal computers. The SIPS expert system effort is identified as Knowledge System Prototype (KSP). The KSP uses OPS 5+ Development System with a C language interface installed on an IBM PC/AT. The SOPS expert system (ES) was implemented on an Apple Macintosh with CLIPS, an expert system building tool, and an interface written by Code 522.

1.2 Spacelab Data Processing Facility (SLDPF) Overview

The SLDPF processes experiment payload data from Spacelab and ASP missions. The SLDPF functions include the capturing, quality monitoring, processing, accounting, and forwarding of data to various user facilities. The SLDPF consists of two major functional elements; the Spacelab Input Processing System (SIPS) and the Spacelab Output Processing System (SOPS). See Figure 1.

During initial SIPS processing, Ku-band channel 2 and/or channel 3 data are captured onto high-density tapes (HDTs). The primary functions in this phase are the real-time capture, the monitoring of data for quality and status coordination with the Spacelab external interfaces such as the Spacelab Payload Operations Control Center (POCC), the Mission Control Center, and the Network elements. After real-time capture, the HDTs, including playback and direct access channel data are post-processed to produce Spacelab Experiment Data Tapes (SEDTs) and/or Spacelab Input/Output Data Tapes (SIDTs).

To complete SIPS processing, analysts perform quality assurance analysis by the manual evaluation of Spacelab Quality and Accounting Records (SQARs). This analysis is aided with information from several Spacelab reports and logs. The results of the QA analysis determines the release of SEDTs, SIDTs and Spacelab Quality and Accounting Tapes (SQATs) to the SOPS or to users.



SLDPF INTERFACES

Figure 1

03/31/87
LB

Additional data processing is performed by the SOPS. The data are edited, time ordered with overlapping data removed, decommutated, and quality checked before shipment to users. In a similar manner to the SIPS QA analysis, SOPS QA analysts combine information from various summary reports and processed logs to determine the quality of data and to decide the data status (release or reprocess).

2.0 CONFIGURATION OF PROTOTYPES

2.1 SIPS Knowledge System Prototype (KSP)

2.1.1 Overall Description and Function

The SIPS KSP is designed to emulate the performance of experienced SIPS QAAs in the evaluation of Spacelab Quality Control and Accounting Records (SQARs). This function is currently performed through the examination of printouts of the SQAR items.

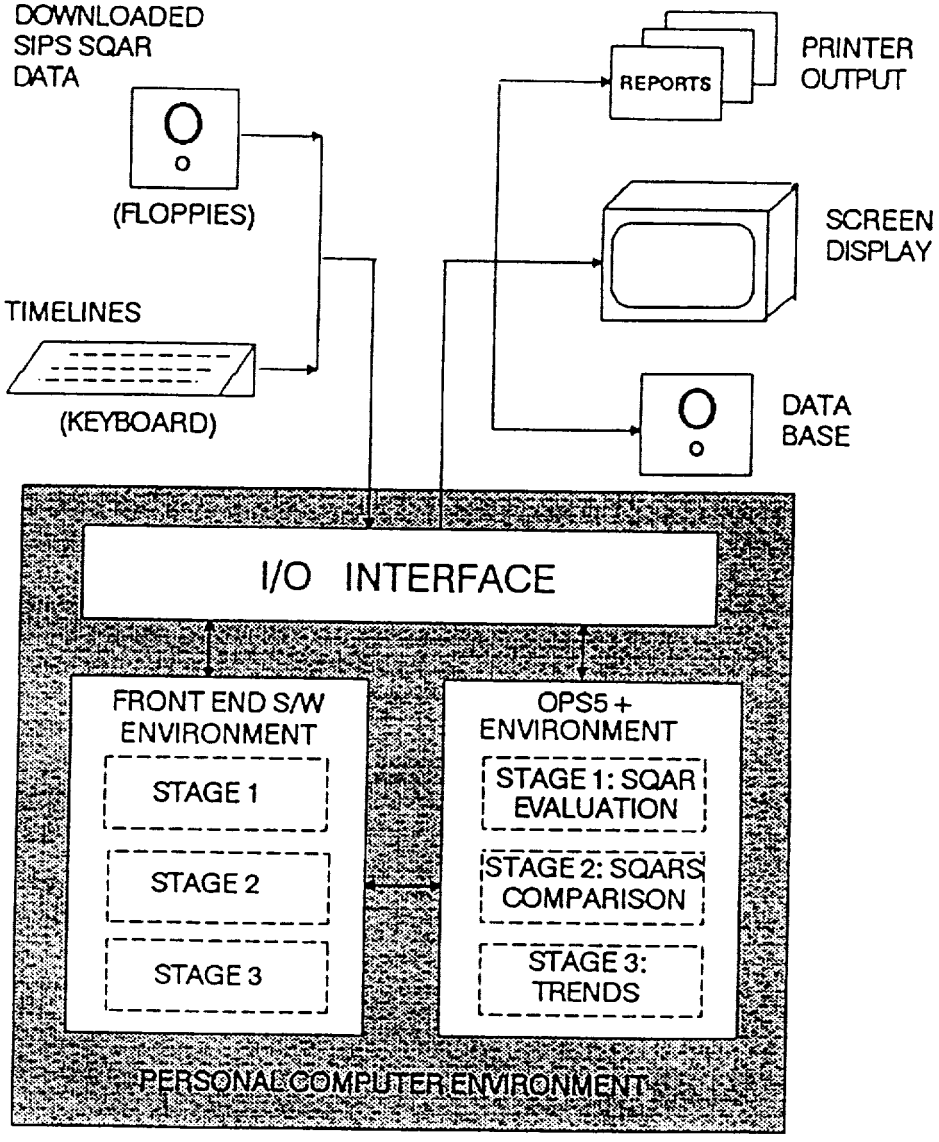
Initially, three problem areas were identified: gathering the expertise of the QAAs, accessing the data which is used in their decision making, and configuring the system on an IBM PC AT. See Figure 2 for a diagram of the KSP configuration.

The first task was the gathering of expertise of the QAAs in the area of SQAR analysis to determine if this area is a practical choice for an expert system. It became apparent that the expert system concept would work, but the scope of the initial effort would have to be restricted due to the extensiveness of the application and the limitations of the prototype hardware and software configuration. Three stages of analysis were established: initial evaluation, comparison of initial and redo runs, and data trends. Each could stand alone logically but needed access to the data and decisions of the others. This problem was addressed by the use of a database to store data as well as the decisions of each stage. The use of the database allowed the expert system to be divided into modules to run with the available memory of the prototype configuration.

The next task addressed was that of accessing the data needed for the decision making. As a test, the most used report, the Spacelab Quality Control and Accounting Record (SQAR) Report was downloaded from the Gould SEL 32/77 to an IBM PC floppy disk. Code was added to the system to read the downloaded report from the floppy and to store the data in the database. The test succeeded and dictated that the data access methods should be automated.

The code surrounding the database continued to grow to include database creation and loading, data validation, data maintenance, SQAR selection, expert system module selection, and expert system report selection. This module is known as the "Front End" because it controls access to and exit from the other expert system modules.

As previously mentioned, the expert system is divided into three parts or stages. Each stage operates independently in the expert system environment. As the expert system modules run, pertinent data and decisions are written to report files from which data base updates and printed summary reports are generated. A Spacelab Quality Assurance and Accounting Record (SQAR) must first be evaluated (Stage 1).



KSP CONFIGURATION

Figure 2

Two evaluated SQARs can be compared and the better one selected (Stage 2). Trends are investigated in Stage 3.

2.1.2 KSP Knowledge Base

The rule-based expert system tool OPS5+ is being used to develop the knowledge base for the KSP. The knowledge elements (rules) are in the following form: "IF <condition(s)> THEN <action(s)>." The KSP knowledge base rules are organized in 3 groups or stages: SQAR evaluation, SQAR comparison, and trends divided (Figure 2).

2.1.2.1 SQAR Evaluation (Stage 1)

In the SQAR evaluation phase (Stage 1) of the KSP (Figure 3) there are 201 rules. The SQAR record produced by the Gould SEL 32/77 is examined and evaluated. The result is a recommendation to accept or reprocess the file in question. The initial SQAR record is placed by the SIPS software automatically in one of four categories: above criteria, abort, hold, or null. The KSP performs further examination to determine how good the data is and if the data can be improved. Analysis occurs for each of the four categories and actions are recommended to the analyst. A summary file is created during the expert system session and is available to be printed at the end of the Stage 1 expert system session.

CATEGORIES:

Above Criteria. SQARs marked "above criteria" are examined for coverage and recovery. Missing intervals are identified and pursued. The KSP can recommend one of three choices: "release (above criteria):, "reprocess (source of improved data identified)", or "release (below criteria, best available)".

Abort. SQARs marked "abort" are examined for coverage, cause of the abort, recovery, data quality, and timing. The KSP can recommend one of two results: "release (above criteria)", "reprocess (abort)".

Hold. SQARs marked "hold" are a mixture of various types of failures. These SQARs are examined for coverage, missing intervals, bad records, and duplicate file components. The evaluation proceeds depending on the problems of the various file components. Recovery, partial (channel) abort, data quality, timing, scheduling, and receipt are examined. Several different types of failure can and do occur simultaneously. The KSP examines each situation and can recommend "release (above criteria)", "reprocess (abort)", "reprocess (source of improved data identified)", or "release (below criteria)".

Null. SQARs marked "null" are one of two types. Either no data was scheduled thus creating a deliberate pause, or data was scheduled and not received. The KSP examines the timeline for scheduling information and various operators' logs to verify data receipt/non-receipt. The KSP can then recommend "release (valid null)", or "reprocess (data expected)".

2.1.2.2 SQAR Comparison (Stage 2)

In the SQAR comparison phase (Stage 2) of the KSP there are 130 rules. This stage allows SQAR records evaluated from Stage 1 to be compared and evaluated

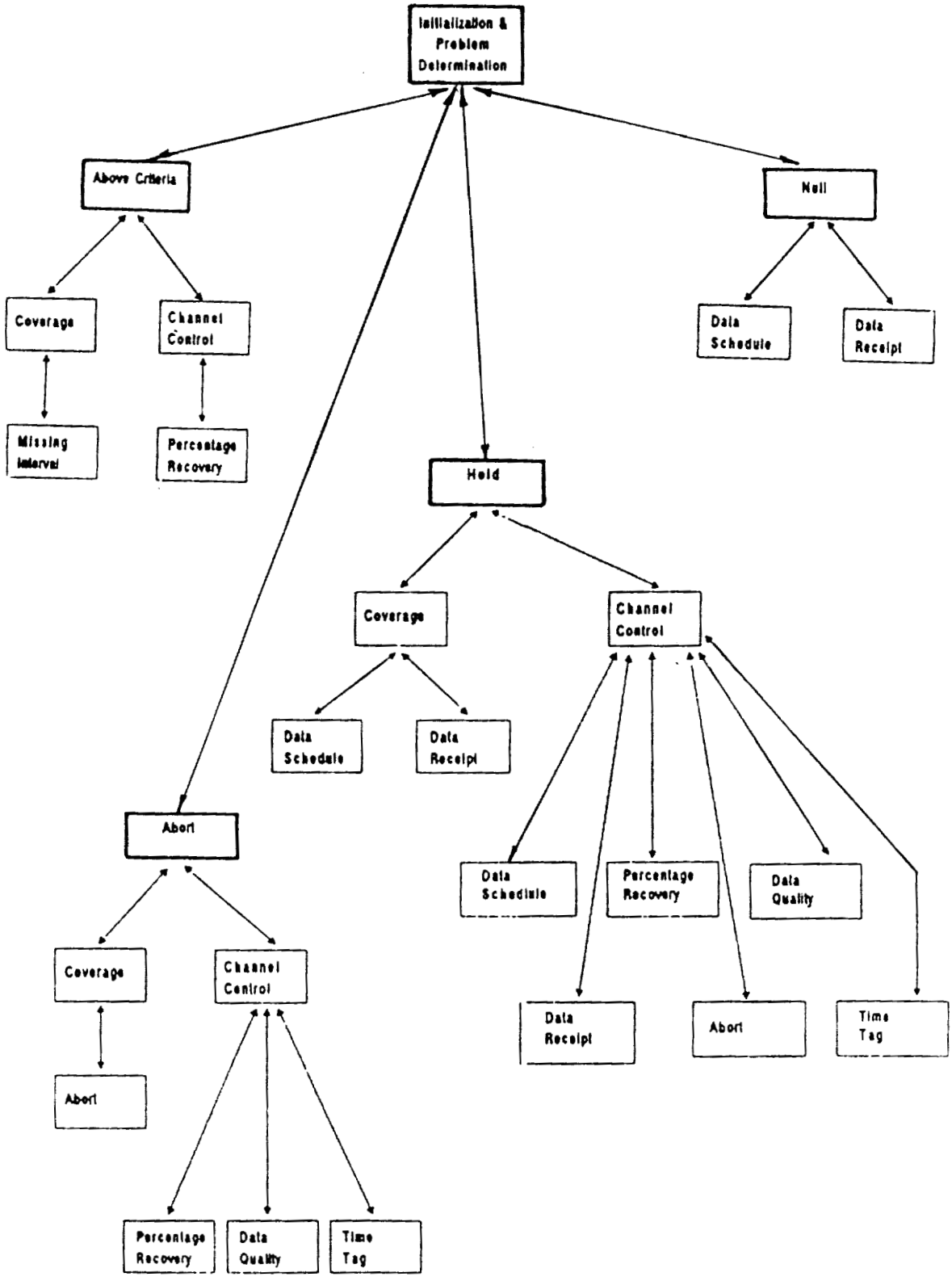


Figure 3. KSP SQAR Evaluation (Stage 1)

(Figure 4). The result is the recommendation of the better of the two SQARs. The comparisons fall into three categories; two null files, one null file and one non-null file, and two non-null files. Extensive analysis is performed on two non-null files. A summary report, a detailed report, and a final status report are available to be printed at the end of the Stage 2 expert system sessions.

Two Non-Null Files. The most meaningful comparison is between two files which both contain data. These files must have the same number of channels, and the channel IDs must correspond. A system of weights assigns values to the data evaluation criteria items: total frames, elapsed time, recovery, data quality, timing, frames without synchronization errors, and frames without timing errors. Evaluations are made on a channel by channel basis followed by a file level recommendation at the end.

One Null File and One Non-Null File. An attempt to compare a null file with a non-null file will be decided in favor of the non-null file. The null file is then marked as redundant.

Two Null Files. An attempt to compare two null files is virtually a draw. The file with the longer elapsed time is selected for retention, and the file with the shorter elapsed time is marked as redundant.

2.1.2.3 Trends (Stage 3)

Trends. Stage 3 of KSP is designed to identify trends from the evaluated SQARs. Indication of trends allows for identifying troubleshooting problem areas. For example, "Do the majority of data failures occur at a certain transmission rate or from a certain piece of equipment?";"Are certain channels failing more than others?"; "Are most aborts located in the same channel or within the same user group?". As a diagnostic tool, this will be beneficial in solving processing problems.

2.1.3 KSP User Interface

The KSP Front End interfaces with the user in the form of selection and input screens. Required responses are limited to one key-stroke if default values are selected (Figure 5). Page forward and page backward options are provided. Data input/viewing screens are provided to allow input and data maintenance (Figure 6).

The KSP expert system Stage 1 interfaces with the user in the form of a running dialog. It is initiated by loading and initializing the evaluation program after entering the OPS5+ environment (Figure 7). Data not directly downloaded is obtained by querying the user. Response requirements are limited to one character. During the Stage 1 expert system operation, a summary report is created that is printed on request (Figure 8). The Stage 2 program is loaded and initialized to perform the comparison analysis (Figure 9). This stage operates without intervention from the user as the SQAR comparison is executed. During Stage 2 operation, both a summary report and a detailed report are created and may be printed on request (Figure 10 and 11).

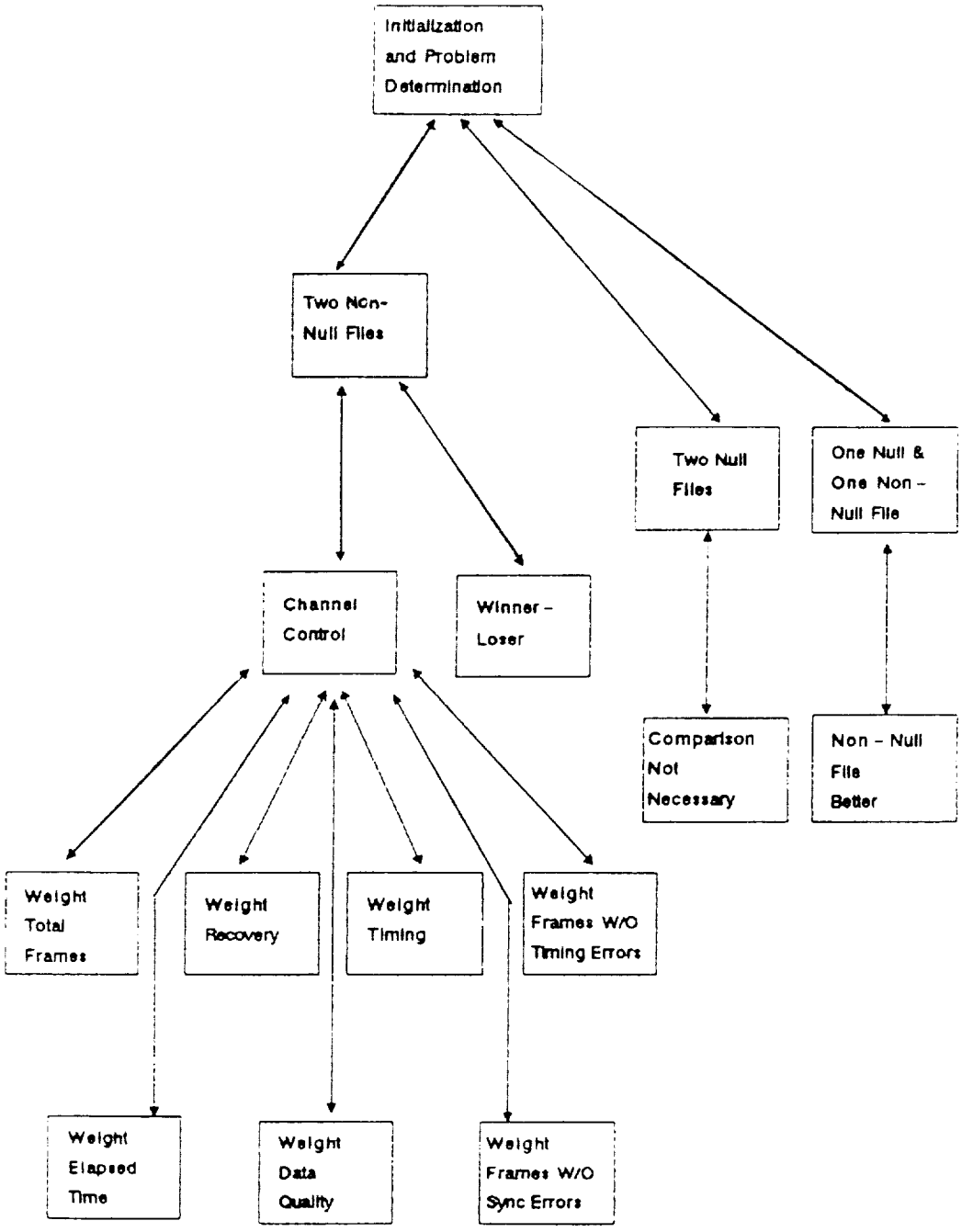


Figure 4. KSP SQAR Comparison (Stage 2)

```
*********SPACELAB INPUT PROCESSING KNOWLEDGE SYSTEM PROTOTYPE*************
************************STARTUP SCREEN*************************************
*                                                                          *
*                                                                          *
*                     TYPE  U, S, C, E, P, OR X                            *
*                     -------------------------                            *
*                                                                          *
*              U (DATA BASE UPDATE PROGRAM)                                *
*                                                                          *
*              S (SCREENS PROGRAM)                                         *
*                                                                          *
*              C (COMPARE PROGRAM)                                         *
*                                                                          *
*              E (EXPERT SYSTEMS PROGRAM)                                  *
*                                                                          *
*              P (PRINT EXPERT SYSTEM SUMMARY REPORT)                      *
*                                                                          *
*              X (EXIT TO OPERATING SYSTEM)                                *
*                                                                          *
****************************************************************************
```

Figure 5. KSP Main Menu

```
******** SPACELAB INPUT PROCESSING KNOWLEDGE SYSTEM PROTOTYPE ************
***************************** FILE MENU ************************************
*                                                                          *
*    FILE TIMELINE START TIME:                                             *
*    YEAR (YY)    DAY (DDD)     HOUR (HH)    MINUTE (MM)     SECOND (SS)   *
*          NN          NNN           NN           NN              NN       *
*    FILE SUTC START TIME:                                                 *
*    YEAR (YY)    DAY (DDD)     HOUR (HH)    MINUTE (MM)     SECOND (SS)   *
*          NN          NNN           NN           NN              NN       *
*                                                                          *
*    FILE TIMELINE STOP TIME:                                              *
*    YEAR (YY)    DAY (DDD)     HOUR (HH)    MINUTE (MM)     SECOND (SS)   *
*          NN          NNN           NN           NN              NN       *
*    FILE SUTC STOP TIME:                                                  *
*    YEAR (YY)    DAY (DDD)     HOUR (HH)    MINUTE (MM)     SECOND (SS)   *
*          NN          NNN           NN           NN              NN       *
*                                                                          *
*    NUMBER OF CHANNELS (1-8): N                                           *
*                                                                          *
***NEXT PAGE**PREVIOUS PAGE**EXIT*****************************************
```

Figure 6. KSP File Menu

```
*****************************************************************************
* Copyright (c) Computer * Thought Corp., 1985, 1986.                       *
* Welcome to OPS5+                                                          *
* : (load "eval.ops")                                                       *
* ****#****#****#****#****#****#****#****#****#****#****#****#****#*****    *
* ****#****#****#****#****#****#****#****#****#****#****#****#****#*****    *
* ****#****#****#****#****#*****                                            *
* : (watch 0)                                                               *
* : (make start)                                                            *
* : (run)                                                                   *
*                                                                           *
*                    THEN THE KSP STAGE 1 EXPERT SYSTEM RUNS                *
*                           UNTIL ALL THE PRODUCTIONS                       *
*                                  HAVE FIRED.                              *
*      .                             .                          .           *
*      .                             .                          .           *
*      .                             .                          .           *
*                    WHEN IT IS DONE, THE SYSTEM RESPONDS WITH              *
*                                  THE MESSAGE:                             *
*                                                                           *
* No production true                                                        *
* : (exit)                                                                  *
* Goodbye                                                                   *
*                                                                           *
*****************************************************************************
```

Figure 7. Load KSP Stage 1

```
============================================================================
SQAR EVALUATION                                                  A0001A/01
============================================================================
FQC = H
File expected = 1001 seconds.  File actual = 716 seconds.  File MI = 285 seconds
ACTION:   Process HRM file when received from DACON.
============================================================================
----------------------------------------------------------------------------
Channel = 1    CQC = A
Computed frames = 238607          Recovery = 83.52814 percent.
FAILURE:  Recovery/external.
ACTION:   Set CQC to F.  ** REPROCESS **
----------------------------------------------------------------------------
Channel = 14   CQC = T
ACTION:   Set CQC to F.  ** REPROCESS **
============================================================================
ACTION:   Set PSC to REQ.
ACTION:   Set FQC to F.  ** REPROCESS **
============================================================================
```

Figure 8. KSP Stage 1 Summary Report

```
*****************************************************************************
* Copyright (c) Computer * Thought Corp., 1985, 1986.                       *
* Welcome to OPS5+                                                          *
* : (load "comp.ops")                                                       *
* ****#****#****#****#****#****#****#****#****#****#****#****#****#*****    *
* ****#****#****#****#****#****#****#****#****#****#****#****#****#*****    *
* ****#****#****#****#****#*****                                            *
* : (watch 0)                                                               *
* : (make start)                                                            *
* : (run)                                                                   *
*                                                                           *
*                  THEN THE KSP STAGE 2 EXPERT SYSTEM RUNS                  *
*                        UNTIL ALL THE PRODUCTIONS                          *
*                              HAVE FIRED.                                  *
*     .                        .                        .                   *
*     .                        .                        .                   *
*     .                        .                        .                   *
*                  WHEN IT IS DONE, THE SYSTEM RESPONDS WITH                *
*                              THE MESSAGE:                                 *
*                                                                           *
* No production true                                                        *
* : (exit)                                                                  *
* Goodbye                                                                   *
*                                                                           *
*****************************************************************************
```

Figure 9. Load KSP Stage 2

SQAR COMPARISON SUMMARY		
CHANNEL ID	A0003A/01	B0004B/08
3	839.389800	1355.610000 * GREATER *
4	1331.000000 * GREATER *	865.000000
5	1103.000000 * GREATER *	1093.000000
Total	3273.389800	3313.610000 * GREATER *

Figure 10. KSP Stage 2 Summary Report

DETAILED CHANNEL COMPARISON

CHANNEL ID 3	A0003A/ 1 Value	A0003A/ 1 Weight	B0004B/ 8 Value	B0004B/ 8 Weight
Total Frames	142804	266	71402	133
Delta Time	1348	266	674	133
Percent Recovery	77	200	77	200
QP1	99.015000	150.000000	99.644000	150.000000
QP4	99.644000	50.000000	99.644000	50.000000
F3	703	---	703	---
Frames Without Sync Errors	142101	333	70699	166
F5	254	---	254	---
Frames Without Timing Errors	142550	66	71148	33
FINAL CHANNEL GRADE	1331.000000		865.000000	
	**** GREATER ***			

CHANNEL ID 13	A0003A/ 1 Value	A0003A/ 1 Weight	B0004B/ 8 Value	B0004B/ 8 Weight
Total Frames	9573	199	9600	200
Delta Time	715	199	716	200
Percent Recovery	83	200	83	200
QP1	94.463000	149.574800	99.850000	150.425200
QP4	99.822000	49.992990	99.850000	50.007010
F3	530	---	520	---
Frames Without Sync Errors	9043	249	9080	250
F5	17	---	15	---
Frames Without Timing Errors	9556	49	9585	50
FINAL CHANNEL GRADE	1095.568000		1100.432000	
			*** GREATER ***	

FINAL GRADE FILE	2426.568000	1965.432000
	*** GREATER ***	

OPTIONS: (1) OVERRIDE GRADE
(2) ACCEPT GRADE

Figure 11. KSP Stage 2 Detailed Report

2.2 SOPS EXPERT SYSTEM (ES)

2.2.1 Overall Description and Function

Code 522 developed the knowledge base for the prototype using the rule-based expert system language CLIPS. In a rule based ES all knowledge elements are represented and processed in the form of If... then... rules. The if is followed by a set of conditions and then by a set of actions that will only take place when all the conditions following the if are met.

The prototype SOPS Knowledge base can be logically divided into sets called knowledge island. Each knowledge island consists of rules to diagnose a problem, drive the user interface, and to retrieve data specific to that knowledge island. This knowledge base structure simplifies the process of modifying the ES. A knowledge island can be modified or replaced to reflect a procedural change in SOPS without affecting the other knowledge islands.

The SOPS prototype ES consists of four knowledge islands: Run Stopped Early, Data Gap Between files, Coverage, and Data Quality. The following sections present a simplified graph depicting the internal structure of each knowledge island along with a brief description. The knowledge islands were implemented in the prototype ES only to the detail required to realistically demonstrate an operational SOPS ES. The project team will expand each knowledge island for future implementation to include particulars uncovered by this prototype ES.

2.2.2 SOPS ES Knowledge Base

Run Stopped Early. This knowledge island determines if the run stopped early and attempts to determine why (see Figure 12a). The prototype ES will determine if the run stopped early by comparing the processed stop time on the SIDT report with the run stop time on the MIDT report. The QA is required to account for the missing data if the time difference is greater than five seconds. If the two stop times are within five seconds then the ES can continue to the next knowledge island; if not, the ES will attempt to determine the cause for the missing data. The ES will first check if the time from the last major frame used in the file is the same as the run stop time on the MIDT report for an indication of a possible run abort during processing. This condition is typically caused by a hardware problem such as a bad tape drive. If the times are the same, the ES will prompt the QA to look in the SIDT database and the card deck to check if the correct files were used for this run.

Data Gap Between Files. This knowledge island determines if there is missing data between two files in the run (see Figure 12b). The prototype ES accomplishes this by comparing the stop time of the first file in the run with the start time of the next file of the same type. The two types of files, high data rate and low data rate, are not compared to each other. The ES will continue to compare the stop and start time for each successive file of the same type. If no gaps greater than five seconds are found between the files, the ES can continue to the next knowledge island.

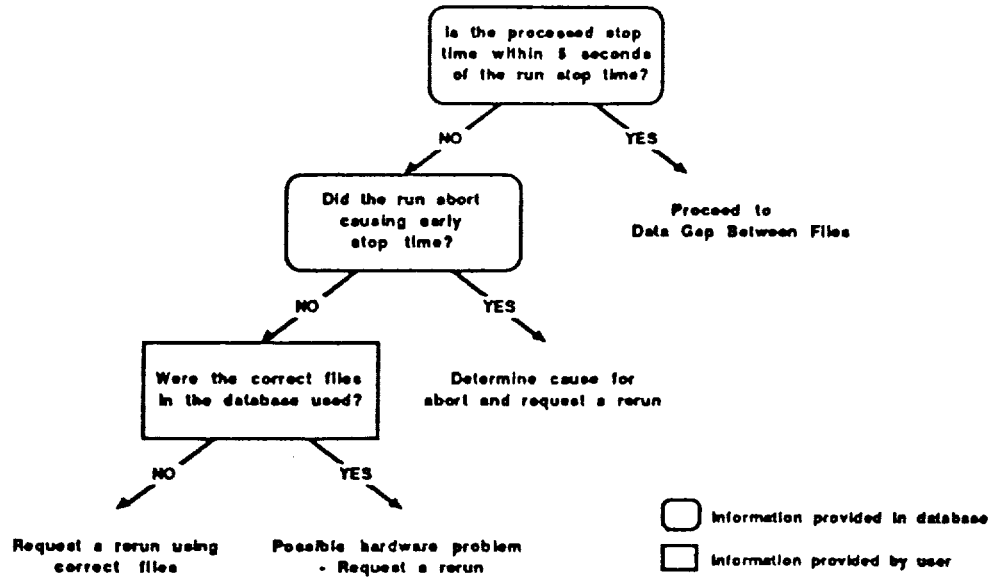


Figure 12a. Run Stopped Early

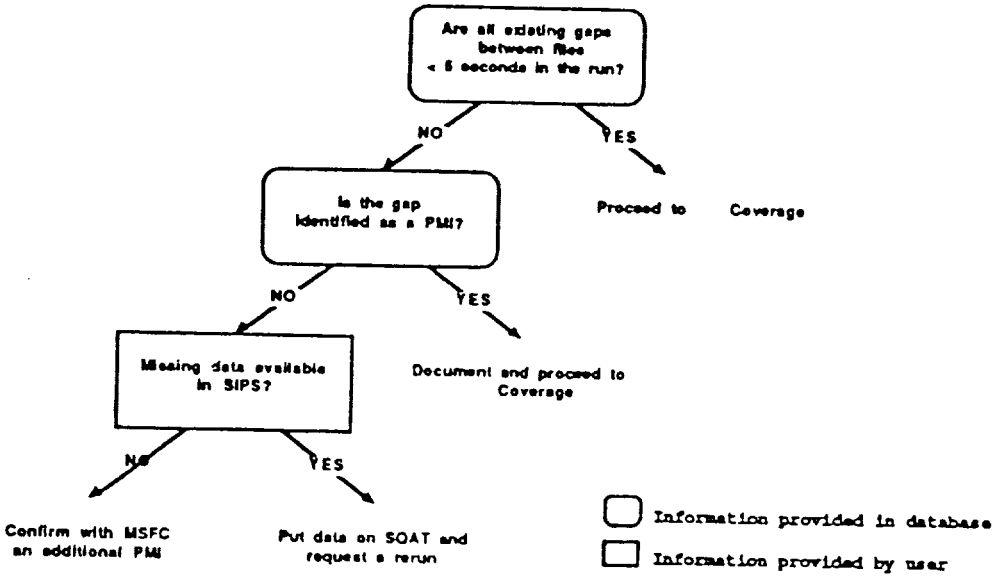


Figure 12b. Data Gap Between Files

In the event a gap is found, the ES will check if the gap is listed on the Permanently Missing Interval (PMI) List; if not, the ES will prompt the QA to determine if the data is available in SIPS by checking the Event Summary Report (ESR), Configuration Controller (CFC) log, Playback Summary log, and Data Processing Summary Report (DPSUM). The QA will request the missing data records, if available, to be placed on a Spacelab quality control and accounting tape (SQAT), the tape loaded into the SOPS database, and the run reprocessed.

If the data is not available, the ES will prompt the QA to determine if the gap is an undocumented PMI. If the QA confirms an undocumented PMI, the ES will insert the PMI on the PMI List and continue.

Coverage. This knowledge island determines if there is missing data within a file in the run (see Figure 12c). The first step the ES takes is to calculate the minor frame coverage for the time between the channel start and stop times for each file. If the minor frame coverage for each file is greater than 98 precent, the ES can continue to the next knowledge island.

If the minor frame coverage is less than 98 percent, the ES will prompt the QA to check the ESR and CFC log for comments about gaps and dropouts in data. In the case where gaps, including PMIs, are noted in the logs, the ES will recalculate the minor frame coverage over the file times containing the gap. If the minor frame coverage is still below 98 percent, the ES will prompt the QA to check on the High Data Rate Recorder (HDRR) for the missing data.

Data Quality. This knowledge island is concerned with the quality of the data and if it can be improved (see Figure 12d). The ES determines the quality of the data by calculating the percentage of error flags set. If the percentage of error flags is greater than two percent, the ES will attempt to check the file quality codes. If SIPS did not release the data below criteria as the best available, the ES will prompt the QA to check the ESR and CFC log for comments about dropouts or poor data. Where no explanation for the poor quality is found in the logs, the ES will prompt the QA to determine if the data can be cleaned up before proceeding.

2.2.3 ES User Interface

The SOPS ES prototype uses many of the features that are standard for applications running on the Apple Macintosh . The features include the use of multiple windows, pull-down menus, and dialog boxes. Figure 13 is an example of the default screen layout used in the prototype.

Dialog boxes and windows may contain buttons, scroll bars, or space for the analyst to type in additional information called a text field. Whenever possible, the ES will set a default value for the text fields. If the analyst changes the value of a text field, the ES should perform consistency checks and prevent the analyst from entering invalid values. For example, if a text field requires a number, the prototype will only allow digits to be typed in. The consistency checking on the operational ES should be expanded to confirm that the value, as it is being typed in, is within the correct range and notify the analyst if it is not.

2.2.3.1 Windows

The primary windows that will be viewed by the QA analyst are the Transcript, Time Line, and Conclusion windows. The Transcript window maintains a log of the ES

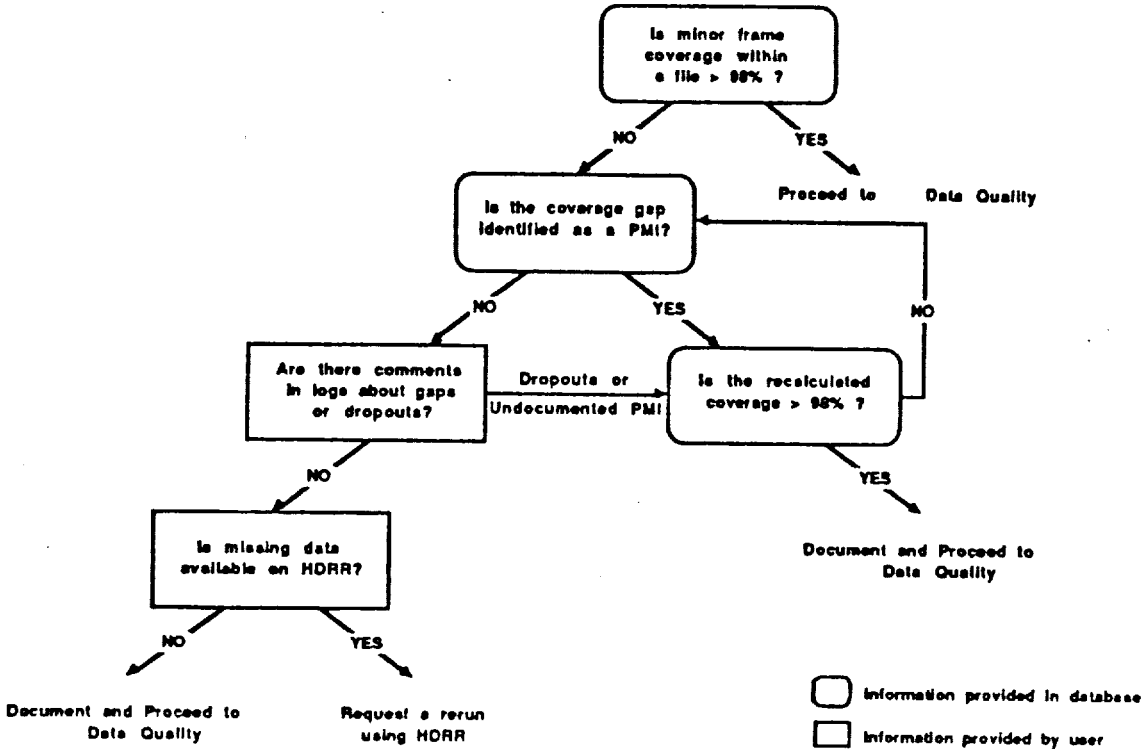


Figure 12c. Coverage

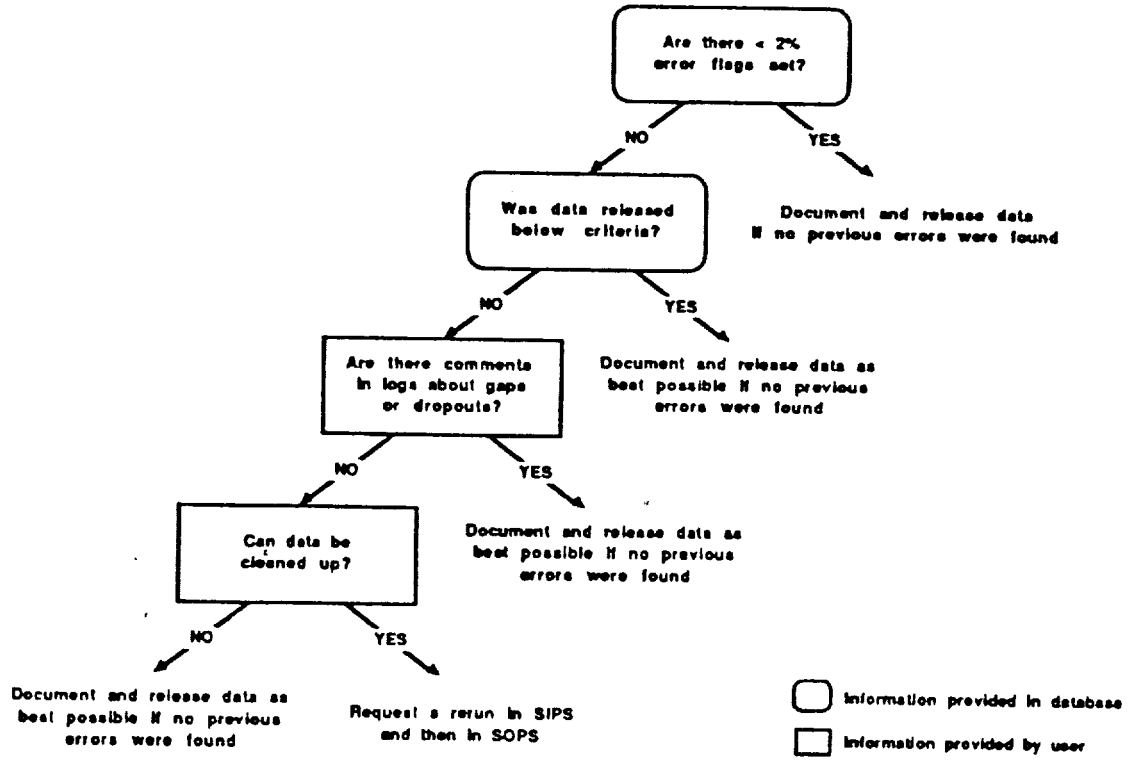


Figure 12d. Data Quality

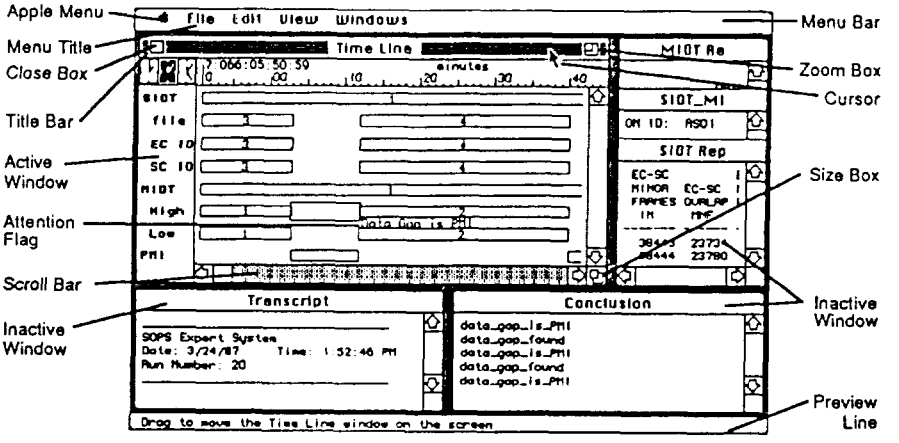


Figure 13 - Default Screen Layout

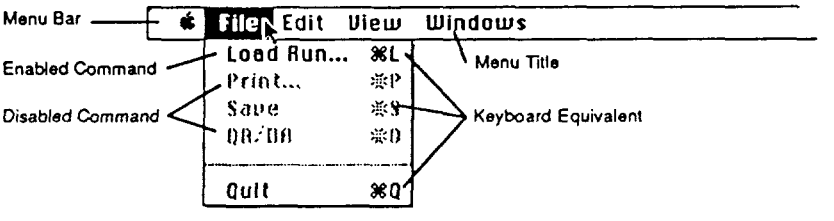
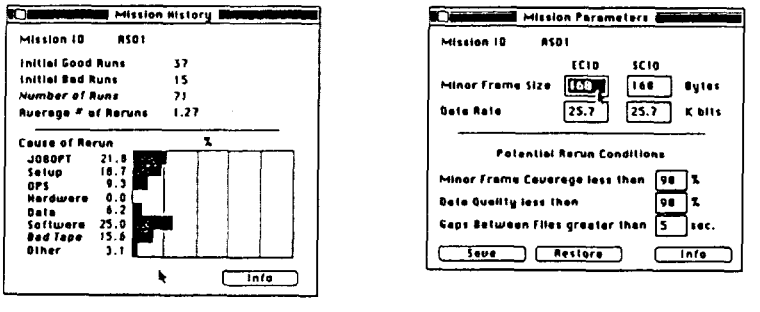


Figure 14 - Example Pull-Down Menu



a) Mission History b) Mission Parameters

Figure 15 - Mission Windows

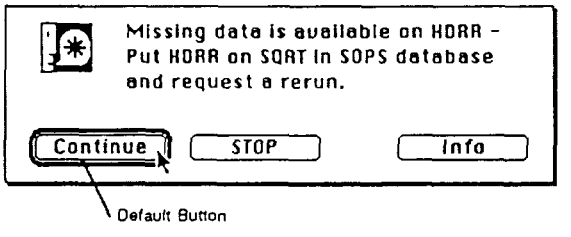


Figure 16 - Recommendation Dialog Box

session that can be printed upon completion. This log will contain all questions asked by the prototype ES and the analyst's responses, all recommendations from the ES, and any comments the analyst wishes to add. The Time Line window displays the run in a graphical format with the expert system's current focus of attention flagged. The Conclusion window displays the conclusions reached (rules fired) by the ES.

The screen also has the SIDT Report, SIDT/MIDT Report, MIDT Report, and preview windows. The report windows contain detailed data about the run being evaluated. The preview line is a special window that displays help information related to the current position of the mouse.

The QA analyst can customize the window arrangement on the screen with the mouse by positioning the cursor on the title bar of a window, pressing and holding the mouse button down, moving the mouse to a new position (the window will follow), and releasing the button (this is referred to as "dragging" an object).

Many of the windows used in the prototype have scroll bars that the analyst can use to change the current view of the contents in the window. Scrolling can only be accomplished in an active window with the scroll bars visible. To make an inactive window active, the analyst positions the cursor in the window and presses and releases the mouse button (referred to as "clicking" on an object). In addition, some of the windows contain a size box and a zoom box for resizing an active window. The analyst simply drags the size box with the mouse to reshape the window, or clicks in the zoom box to expand the window to the full size of the screen. Clicking in the zoom box again will return the window to its original size and position. The QA analyst can use this feature to get a more comprehensive view of a window and then return without disrupting the layout of the screen.

2.2.3.2 Menus

Displayed at the top of the prototype screen is the menu bar (see Figure 14). It contains the titles of the menus available. To choose a command from the menu, the analyst positions the cursor over the menu title and holds the mouse button down. While holding down the mouse button, the analyst moves the cursor down the displayed menu. As the cursor moves to each enabled command, the command is highlighted. When the analyst releases the button on a highlighted command, that command is selected. A shortcut for selecting some commands is holding down the Command key in combination with another key called the keyboard equivalent. Commands that have keyboard equivalents list them in the menu.

The (Apple) menu contains up to 15 desk accessories such as a calculator or a clock that the analyst can use during an evaluation run. Choosing any of the desk accessories causes that acceessory to appear on the screen. The analyst can use the Edit menu to cut, copy, and paste the information in most desk accesories.

The File menu contains the following commands for processing a run file:

Load Run... - prompts the analyst for a run number and load the run into the ES;
Print... - prints the results of the ES run evaluation (not implemented in the prototype);
Save - saves the results (not implemented in the prototype);
QA/DA - starts or resumes the evaluation of a run; if the analyst has stopped the evaluation before completion, this command in the menu will be Resume QA/DA; and
Quit - quits the SOPS ES.

The Edit menu allows the analyst to perform the standard Macintosh cut, copy, paste, and clear commands on text windows and desk accessories. The analyst can copy and paste data from the Report windows or conclusions from the Conclusion window into the Transcript window.

The View menu contains the following commands:

PMI List - displays the PMI List window and allows the analyst to add a PMI;
Run History - displays the processing history of a run (not implemented in the prototype); This might take the form of the last Transcript file or a summary of previous ES evaluations;
Mission History - displays a summary window of statistics such as the number of good and bad runs over the length of the mission (see Figure 15a), and
Mission Parameters - displays the Mission Parameters window that allows the analyst to change mission specific parameters or evaluation criteria before starting the ES evaluation (see Figure 15b).

The Windows menu contains a list of all the windows on the screen. A window can be selected from this list to make it active and redrawn as the front window. The Windows menu also contains a Clean Up command which will restore the default layout of the windows on the screen.

2.2.3.3 Dialog Boxes

The prototype ES uses dialog boxes to prompt the QA analyst for more information or to display a recommendation. In the prototype ES, dialog boxes are used in two forms: modal and modeless. A modal dialog box is one that the analyst must acknowledge before doing anything else. Since modal dialog boxes restrict the analyst's options, the prototype only uses them for messages requiring the attention of the analyst. Figure 16 is an example of a modal recommendation dialog box. A modeless dialog box allows the analyst to perform other operations before responding to the dialog box.

The information in a dialog box is designed to be as concise as possible so an experienced QA analyst is not burdened with lengthy messages and explanations. For this reason, an Info button is available for less experienced analysts. The analyst can click on the Info button to get an additional page or pages of information if needed. The additional information might include a more detailed

explanation, or in the case of a question dialog box, the information might specify where to get the data to complete the dialog box.

Many of the dialog boxes in the prototype have a bold outlined default button. The default button is always selected if the analyst presses the Return or Enter key on the keyboard. The advantage of a default button is that the analyst does not have to move his hands off the keyboard to respond to the dialog box. If a dialog box does not have a default button (bold outlined button), pressing the Return or Enter key does not have any effect.

3.0 BENEFITS OF PROTOTYPES

The Spacelab expert system prototypes offer many benefits. They are fast. They are consistent. They make the expertise of the most experienced staff members available to all. The prototypes can act as training tools when refined to a detailed level. As they are developed, they identify ways in which current procedures could be further automated to increase accessibility to information and improve processing speed as well as to decrease the monotony of repetitious tasks. They also identify areas in their own operation that should be streamlined to make the expert system concept not only workable but practical.

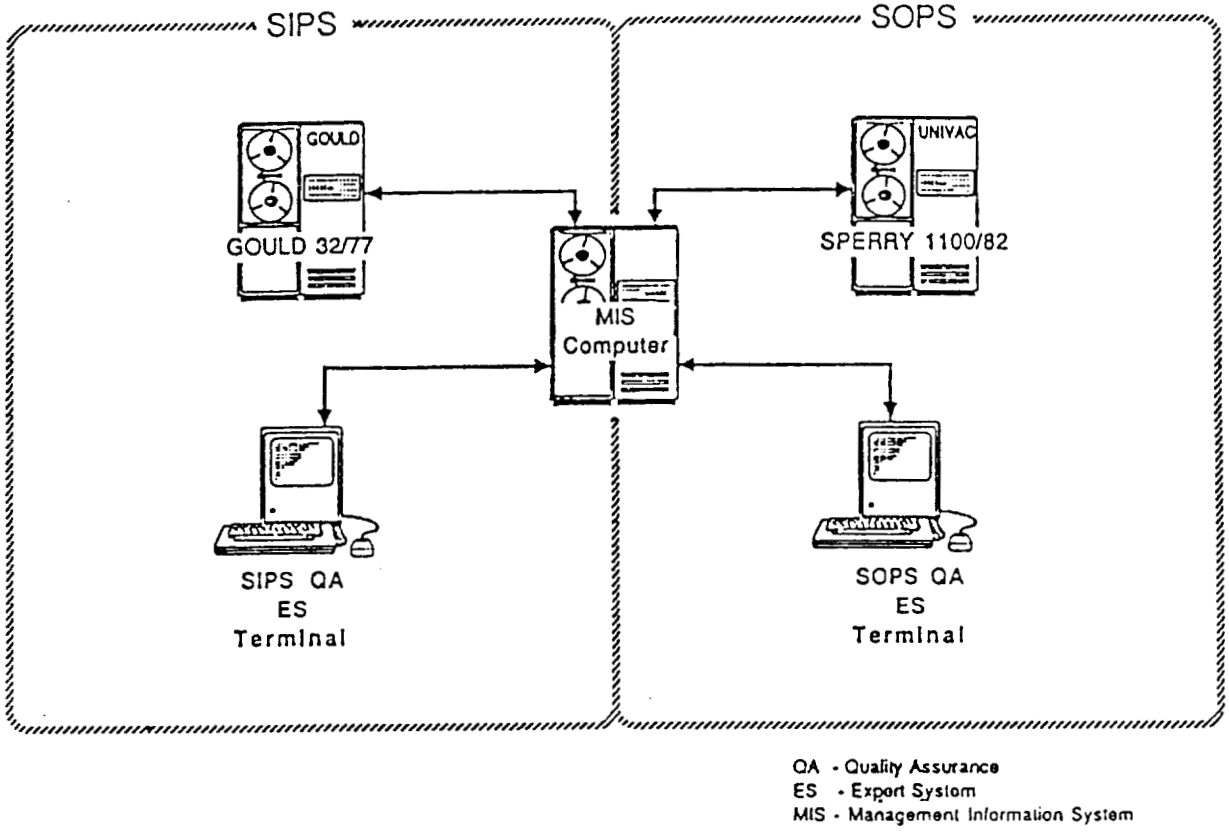
4.0 OPERATIONAL CONFIGURATION

The goal of the Spacelab prototype expert systems is to define the design and the configuration for expert systems in the mission environment. These new operational expert systems will be larger, more efficient, and more automatic, incorporating the capabilities indicated by but not present in the prototypes. Both the SIPS and the SOPS operational expert system configurations will make use of the same hardware and software for consistency (see Figure 17). It is planned that the initial configuration will be operational by July 1988, in time to support ASTRO-1, the first of several scheduled SLDPF missions in the post-Challenger period.

Acknowledgements

Michael Alvarez, Franz Berlin, Warren Case, Michael Garner, James Pizzola, and Beth Pumphrey (all of Lockheed) provided the Spacelab QA expertise for the knowledge bases. This project could not have been successful without their contributions.

FINAL EXPERT SYSTEM CONFIGURATION



QA - Quality Assurance
ES - Expert System
MIS - Management Information System

Figure 17. Operational Configuration

AN EXPERT SYSTEM APPROACH TO ASTRONOMICAL DATA ANALYSIS

Mark D. Johnston

Space Telescope Science Institute[1]
3700 San Martin Drive
Baltimore, MD 21218

ABSTRACT: *Expert System technology has much to offer to the problem of astronomical data analysis, where large data volumes and sophisticated analysis goals have caused a variety of interesting problems to arise. This paper reports the construction of a prototype expert system whose target domain is CCD image calibration. The prototype is designed to be extensible to different and more complex problems in a straightforward way, and to be largely independent of the details of the specific data analysis system which executes the plan it generates.*

I. INTRODUCTION

In recent years there has been an enormous increase in astronomical data volume and variety. A large fraction of astronomical data analysis is now automated, due to the increasing use of large format digital detectors on groundbased telescopes as well as data from satellite observatories. This trend will accelerate in the future with the launch of Hubble Space Telescope and the others in the "Great Observatories" series of satellites. At the same time, data analysis goals and methods have become increasingly sophisticated. New data reduction tools and techniques have been developed and are now in common use.

In response to the flood of data at all wavelengths, and to advances in display and computational hardware technology, astronomical data analysis systems have grown in number and functionality. The general philosophy of these systems is typically similar to the philosophy of the major computer operating systems: there is a command language (CL) which serves as the user interface in a "command/prompt" mode. The CL executes either single commands interactively, or scripts (procedures) of sets of commands (generally with a choice of interactive or batch/background execution). CL commands reduce to the execution of mod-

[1]Operated by the Association of Universities for Research in Astronomy for the National Aeronautics and Space Administration

ular operators which work on standardized types of data files. The advantages of this philosophy are clear:

- great flexibility for the user: individual commands can be combined to construct powerful tailored procedures
- ease of development: it is often relatively straightforward to add new modules, following a recipe that varies from system to system. Many programmers may thus independently contribute to the growth of a system.

There are, however, some serious drawbacks. Learning a system is not easy: commands are often complex with many parameters, and experts don't know all parts of even one system. To compound the problem, users often have to learn more than one system depending on where and how they obtain their data. It has also proven very difficult to capture and make available expert knowledge. Users can obtain assistance from manuals (often of enormous volume, hard to use and maintain), online help (often either irrelevant or a deluge of details), or by befriending the local expert on a particular topic. The same standardization that makes it easy for an expert to add new programs is often daunting to a casual user, who therefore writes "throwaway" code to accomplish some specific task. In some sense, the general philosophy breaks down when the size of a system becomes too large. In contrast to computer operating systems, where a novice need only learn a few commands to successfully navigate a system, there is no avoiding the complexity of a powerful analysis system when a difficult reduction is undertaken.

In response to these problems it is appropriate to consider alternative approaches to the general problem of astronomical data analysis. The most promising of these alternatives centers on the use of Expert System (ES) technology, which has matured dramatically in the past decade. Expert System languages and environments offer significant advantages over classical ones. They facilitate the manipulation of symbolic data, in contrast to languages that are primarily numerical in emphasis. They offer new data representation and programming paradigms that have proven successful on a wide variety of problems. Among these are:

- production (rule) systems: collections of IF...THEN... rules
- object-oriented programming: data representation in terms of modular "objects" (data structures) with control by "message passing" instead of procedure calls
- frames and inheritance: hierarchical data structures with properties inherited from parent classes
- new techniques for searching large problem spaces
- natural language capabilities: input/output in English-like syntax
- nonprocedural program specification: what to do rather than how to do it

These new programming methodologies allow the construction of programs that would not have been attempted before, and at the same time they increase programmer productivity. ES technology is maturing

and becoming commercial: in recent years the field has developed from an area of computer science research into a variety of commercial products. The most important of these is the "Expert System Shell", which provides a general collection of inference mechanisms, data structures, and development tools that can be easily adapted for different applications. Examples now exist of applications of expert systems to a wide variety of problem types (see, e.g., [1,2]).

Other S/W engineering advances are integrated with current ES shells but not yet in most other computing environments. These advances include rapid prototyping as a system development methodology and facilities for easy user interaction with the system. In current expert system shells, advanced development tools and user interface features are integral. These include extensive graphics for display of system status and data, mouse/menu user interaction, multiple screen windows, and natural language command structures. Capabilities such as these make these systems ideal for interactive applications.

What can ES offer to development of astronomical data analysis?

- new programming methodologies to apply to current problems
- a way to capture and disseminate expert knowledge
- a powerful means to attack new problems

To date, application of ES technology to astronomical data have been limited to a few types of classification problems [3,4], but it is clear that the potential of ES methodologies will allow much more powerful and general applications.

The concept of an expert "Data Analysis Assistant" suggests vastly different things to different astronomers, ranging from sophisticated online help to what essentially amounts to an astronomical research assistant. For the present project, the following guiding principles were adopted:

- take advantage of the power of current data analysis systems wherever possible ("don't reinvent the wheel")
- take advantage of the development and user interface capabilities of ES shells running on the current generation of AI workstations. This leads to an architecture (discussed more fully below) where the ES resides on a workstation networked to a host; the latter can execute data analysis and display operations at the request of the workstation.
- deal first with the simpler problem of reasoning about data descriptions (rather than about data contents). This allows "loose coupling" of the workstation and host.
- extend the scope of the system to –

  (1) reason about data contents,

(2) fully incorporate the user and host in the plan/execute/analyze cycle, and

(3) reason about astronomical objects where this can productively help to guide the analysis process

This paper describes the construction and operation of a prototype system following the approach outlined above. The primary goals of the prototype were:

- develop a general means for representing knowledge about different kinds of data, instrument modes, and data analysis operations
- demonstrate the ability to construct a plan to achieve high-level analysis goals in terms of low-level operations, independent of any specific analysis system or language
- demonstrate the ability to efficiently recognize and eliminate redundant operations
- demonstrate the ability to automatically generate a command procedure to execute the plan in one of several specific data analysis languages
- include intrinsic extensibility to handle increasingly high-level goals and alternative plan generation strategies
- provide an easy-to-use mouse/graphics interface to the system user

Section II provides a description of the example problem chosen as a test domain, followed by a description of the design and operation of the prototype system. Plans for extension of the prototype are discussed in Section III, and some general conclusions are presented in Section IV.

II. THE PROTOTYPE

A. The Problem Domain

The problem chosen as the test domain was that of CCD calibration (sometimes called "pre-reduction") which consists of removing the gross instrumental signature from the astronomical CCD images. This is a tedious but relatively routine (at least in its basic form) preliminary step in the overall reduction of CCD images. This domain was chosen not because it represents an outstanding problem in astronomical data reduction but because it provides a simple test case for the methodology.

As implemented in the demonstration system there are four basic steps:

1. extraction of a subimage representing valid data
2. bias subtraction

3. dark current correction
4. division by a flat field to correct for spatial nonuniformities in the CCD response

The first two steps depend only on characteristics of the instrument mode. The last two are more complicated, since dark and flat images are typically taken before and after science images and must therefore be identified and averaged to derive appropriate calibration images.

The operation of the prototype proceeds as follows:

1. The user provides a description of all relevant images (dark images, flat field images, and science images) to the system. In principle, most of this descriptive data could come from headers recorded with the image data, but there are some problems with this in practice.

2. The system analyzes the data descriptions to determine which dark and flat field images should be used to calibrate which science images. This step may identify science images for which no calibration data can be found: these are simply marked as problem images and ignored in subsequent processing.

3. The system then generates a plan network to calibrate each science image. This network consists of "tasks" representing image processing operations to be performed on the calibration and science images, along with the specification of any required ordering of the tasks. Any problems encountered in generating the plan are recorded and presented to the user.

4. Following plan generation, the user selects a specific command language for representing the plan. In the prototype two choices are offered: SDAS/IRAF (developed at Space Telescope Science Institute and National Optical Astronomy Observatories [5,6]) or MIDAS (developed at European Southern Observatory [7]). The system then turns the plan into a specific sequence of image processing commands (essentially a command procedure) in the chosen language. This procedure could simply be shipped via the network to the host for execution, although this was not implemented in the prototype.

B. Architecture and Implementation Hardware and Software

The prototype was constructed at the Space Telescope European Coordinating Facility at the European Southern Observatory in Munich. The system was implemented using the KEE Expert System shell from Intellicorp Inc., running on a Symbolics 3620 lisp machine; the latter will eventually be networked to a cluster of VAX 8600s which will serve as the host for image processing operations. A schematic view of the system architecture is shown in Figure 1. The user interacts only with the ES processor except for image display and manipulation which will be handled by the host processor. From the point of view of the

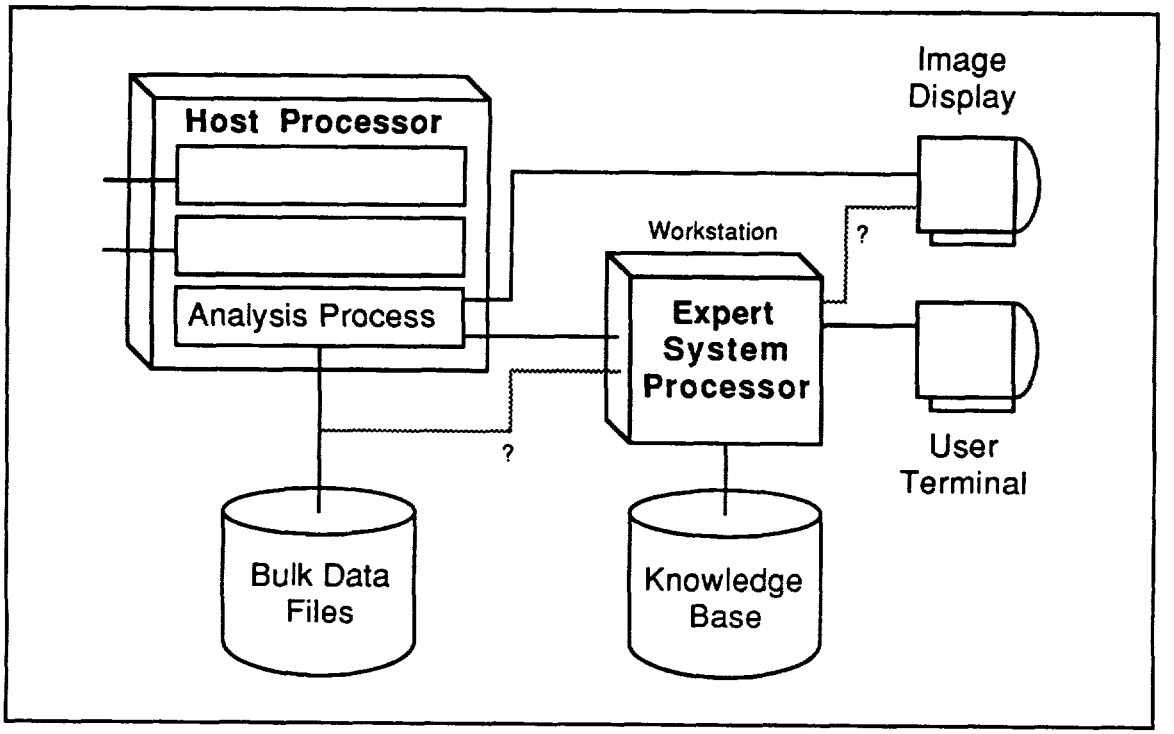


Figure 1—Architecture Overview. The host processor executes data analysis operations at the request of the ES processor. Bulk data files remain on the host, while the knowledge base is maintained on the ES workstation. Gray lines represent possible future connections whereby the ES could access bulk data files via a network file server, and directly control the image display.

host, the ES is simply another user accessing the system over the network. The gray lines represent possible future connections that would allow the ES processor to directly access bulk data files via a network file server, and to directly control graphics and image display devices. These are of particular interest if the ES can run efficiently on a general purpose workstation as well as on a lisp machine. This possibility will be evaluated on a Sun 3/160 workstation.

KEE provides a wide range of programming tools and techniques for implementing expert systems (indeed, one problem is to decide which of several possible representations or approaches are "best" for any particular problem.) Those of most relevance to the design of the prototype are the following:

production rules: rule classes may be defined and invoked in forward or backward chaining mode, or a combination of both

frame system: this provides for the definition of classes of data objects which inherit properties from their parent classes. Frames have "slots" which can hold data items, references to other frames, procedures, etc. Of particular utility are "methods", which provide for object-oriented programming (procedures are invoked by sending messages from one object to another), and "active values", which are methods automatically invoked when a slot is set or referenced.

graphics interface: various kinds of graphical display or control images can be attached to frames and slots. These provide for method invocation when the user clicks with the mouse on an active region, graphical displays of system status, etc.

These features are well integrated, allowing for rapid construction and modification of a prototype.

C. System Design

The design of the prototype may be logically divided into the following areas:

1. knowledge representation (data, instrument modes, analysis tasks)
2. control structure
3. reasoning about the data
4. generation of the analysis plan
5. conversion of the plan into a language-specific procedure

Each of these areas is discussed in more detail in this section.

(1) Knowledge representation

A diagram representing the structure of the knowledge base is shown in Figure 2. Only those frames required for the operation of the prototype on the CCD calibration domain are shown, along with an indication of where extension would be possible to handle other types of problems. There are three major classes of objects:

Data: this class holds descriptions of various types of data. Only a few classes of CCD data are defined in the prototype: moving down the hierarchy, each class is a specialization of its parent class and contains slots that characterize that specific type of data. So, for example, the CCD-data class defines properties common to all CCD images (such as number of rows and columns), while the CCD-science class defines properties not shared by CCD-darks and CCD-flats (such as target name). Specific image files are made known to the system by defining a frame as a *member* of the appropri-

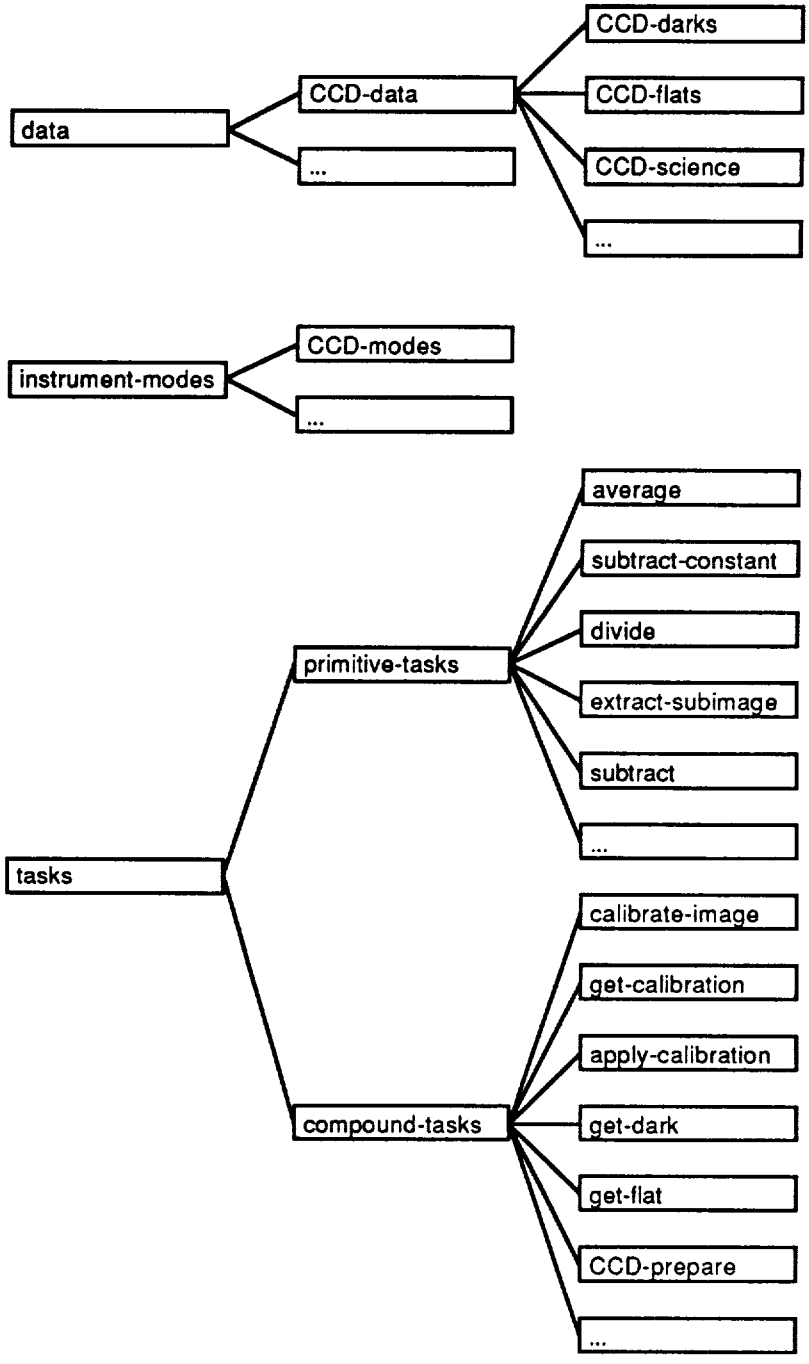


Figure 2.— Diagram representing the major classes defined in the prototype knowledge base (see §II C)

ate class and by filling in its particular descriptive slots with values which characterize the image.

Instrument Modes: only a single subclass is defined in the prototype, representing a "pseudo-CCD" mode. Specific modes are defined as members of this class. Their slots hold such information as the number of valid rows and columns, the bias value to subtract, and any other properties which are common to all data taken in that mode. Members of the CCD-data class must reference a member of the CCD-modes class in order for the system to know mode-dependent data characteristics. This eliminates the need to repetitively specify mode characteristics in the frames which represent data.

Analysis Tasks: each task *class* represents a "generic" analysis operation (e.g. *calibrate-science-image*). A *member* of task classes represents a "specific" operation, i.e. with input and output files fully specified (e.g. *calibrate-frame-1*). The construction of the analysis plan involves the definition of these task members and linking them in the proper order, as described below.

Tasks are divided into **primitive** and **compound** tasks as follows:

primitive tasks are "atomic" in the sense that they can be implemented with a single command (or simple series of commands) in any specific data analysis system. Basic image arithmetic operations fall in this category, along with operations such as extracting a subimage of a larger image. Only primitive tasks need have any knowledge of how they are represented in a specific data analysis language. To accomplish this, each primitive task has defined a "format" method which can generate a command statement implementing the task. This method generates different command statements depending on the language currently selected by the user. It is important to note that this "format" method is the *only* place in the system where languange-dependent information is required.

compound tasks represent "high-level" operations. Compound tasks cannot be directly converted into analysis commands, but instead must be "expanded" into a network of subtasks, each of which can be either compound or primitive. Ultimately, all compound tasks must expand into primitive tasks for the plan generation process to successfully complete. For example, the *calibrate-image* task expands into *get-calibration* and *apply-calibration*; the former expands into *get-dark* and *get-flat*, etc. An important design feature is that compound tasks need only specify tasks into which they immediately expand. This makes it easy to add new high-level tasks by using those already defined as building blocks.

(2) Control strategy

The operation of the prototype is controlled by the user by clicking in active regions of a graphical "control panel" (see Figure 3 for a copy of the screen immediately after a "reset" operation has completed). In the prototype the user can only take four actions (once the data descriptions have been entered):

1. reset: initialize the system
2. run: invoke the plan generation step
3. choose language: one of SDAS/IRAF or MIDAS must be selected
4. format: convert the plan generated by step (3) into a command procedure in the selected language

The plan generation step is controlled by a set of forward-chaining production rules. Conflict resolution is controlled by grouping rules by "salience" or weight; within a group, recency of rule instantiation determines which rule to fire next. The major salience groups are (in priority order):

1. reason about data properties
2. check for problems with data
3. establish top-level goals
4. merge redundant tasks
5. expand compound tasks
6. check for problems with tasks

Each of these groups is discussed below with the exception of (3): this group contains only a single rule to generate a *calibrate-science-image* task for each uncalibrated science image. In a more general system, the goal (or goals) would be specified by the user rather than by a specific rule of this type.

The plan generation process constructs a directed acyclic graph of tasks representing the analysis plan. Each task in this network contains references to any tasks that must immediately precede or follow it. These task-to-task "links" are used to ensure that any required task orderings are preserved as the plan evolves, without committing early to a linear sequence of tasks that would be hard to reorder later. While it is possible to manage these links with a rather complex set of rules, it is easiest to manage them procedurally via methods and message-passing. This approach greatly simplifies the rulebase and speeds up system operation, and also illustrates nicely how the integration of object-oriented and rule approaches can simplify system design and implementation.

(3) Reasoning about the data

The purpose of this rule class is:

- to infer any unspecified data properties that might be required when plan generation starts, and

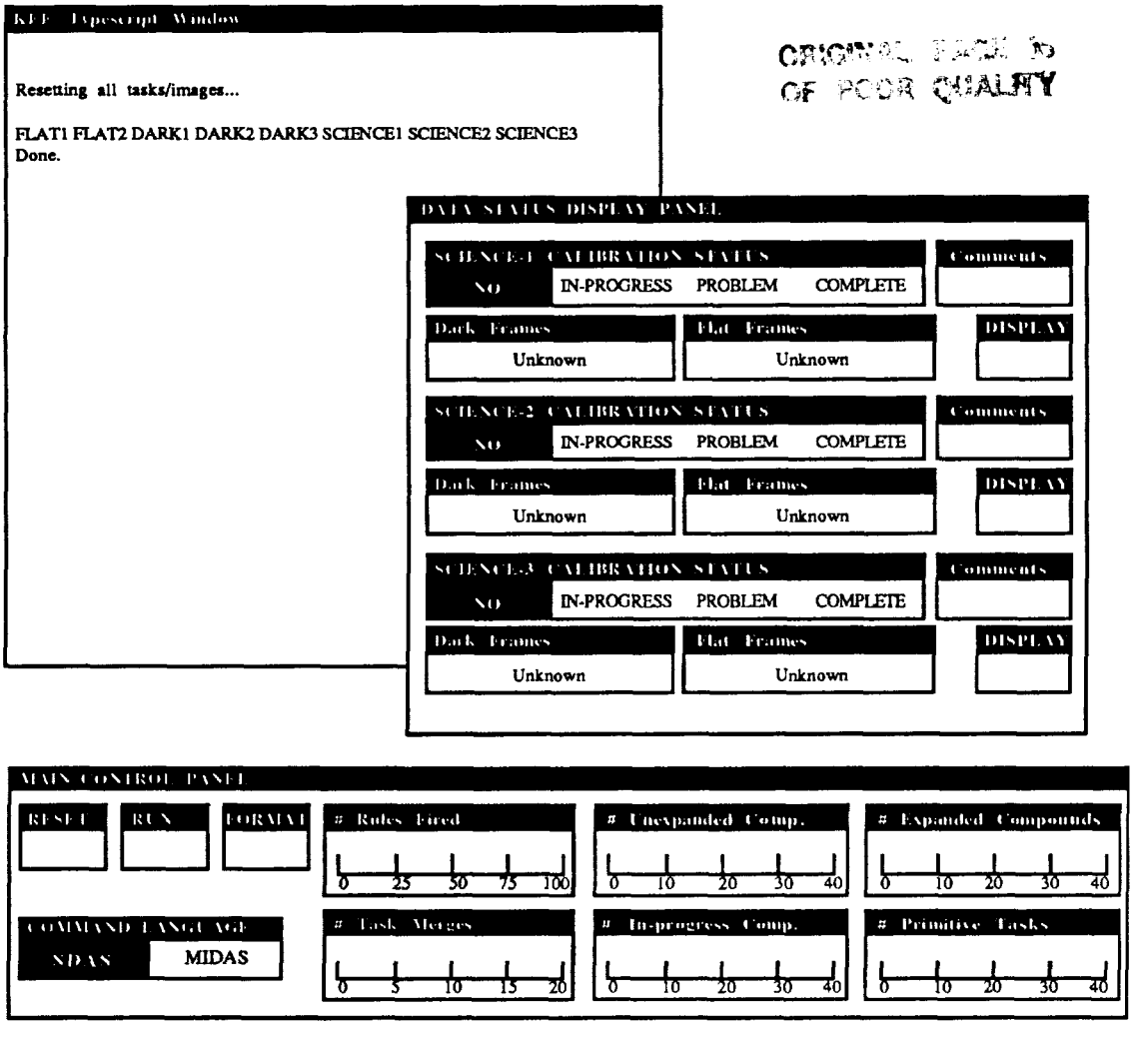


Figure 3—What the screen looks like immediately after execution of a "reset" command. The lower panel is used for overall system control and status monitoring. The upper right panel displays the calibration status of three science images as the system runs. Informative messages appear in the upper left text window. The user clicks with the mouse on an active region (e.g. RESET*) to control operation.*

- to associate calibration images (darks and flats) with science images.

In the present prototype this association is very simple (e.g., flat fields need only be taken on the same night as the science image with the same instrument mode and filters). There are also rules to detect and annotate problems, such as an inability to identify any calibration images for a particular science image. Extension of this rule class to handle more complex cases is straightforward.

(4) Generation of the analysis plan

Plan generation consists of two competing processes: **expansion** of compound tasks and **merging** of redundant tasks.

Expansion of compound tasks requires at least one rule per task; alternative expansion strategies would be implemented by including multiple rules which look for appropriate preconditions. An expanded compound task is marked as such, and all predecessor/successor links in which it took part are removed and attached to its newly created subtasks (see Figure 4(a) for an illustration). Link manipulation is handled procedurally, by sending a message to a task that is should propagate all of its current links to its subtasks. Intermediate data files are automatically given unique names. Expanded tasks are not deleted: they are used for identifying mergeable tasks as described below.

Some tasks expand into a fixed number of subtasks, while others can generate an arbitrary number depending on the circumstances. For example, the *get-dark* task expands into an unordered series of *CCD-prepare* tasks (one for each input image) followed by a single *average* task. A minimum of three rules were found necessary for this situation: one to initiate expansion and create any fixed tasks (*average* in the *get-dark* case); one to create and link each of the variable number of subtasks (*CCD-prepare* in the current example), and a final rule which notes that expansion is complete.

Merging of redundant tasks is accomplished by one generic rule, which essentially says that if two tasks of the same type are found with identical input, then mark one as redundant and change the predecessor/successor links of all affected tasks to reflect the required ordering (see Figure 4(b,c) for an illustration of this process). A note is made in a table that the output files of the redundant task and its replacement are to be identified. As with task expansion, the maintenance of task link information is performed procedurally. The task merging rule has higher priority than any task expansion rule, in order to catch redundancies at as high a level as possible. In the prototype, two of the example science images use exactly the same set of dark and flat field images for calibration: as a result of the merging rule, the tasks which compute the average dark and flat field images will only be planned once.

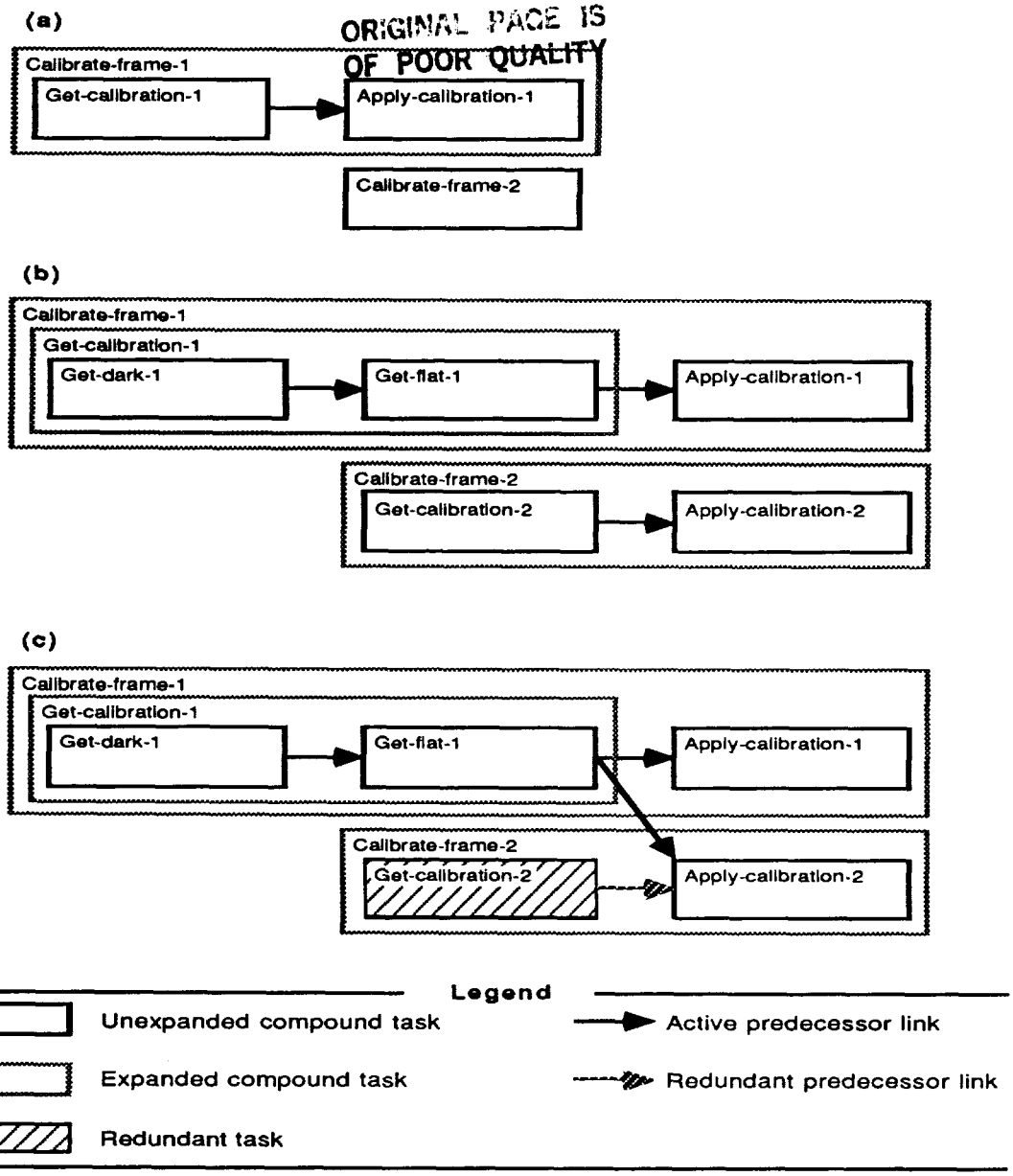


Figure 4— Illustration of the expansion and merging of tasks as the task network is built. (a) calibrate-frame-1 *is expanded (b)* calibrate-frame-2 *is expanded, and the subtask* get-calibration-1 *of* calibrate-frame-1 *is expanded (c) the system recognizes* get-calibration-2 *as identical to* get-calibration-1*: it is therefore marked as redundant and the predecessor links are changed to ensure that the tasks remain properly ordered. See §II C(4).*

At the conclusion of expand/merge rule processing, any unexpanded compound tasks are marked as errors and displayed to the user. Their presence indicates that the rulebase is not complete and that no knowledge about how to accomplish these tasks resides in the system.

(5) Conversion of the plan into a language-specific procedure

Once created, the task network representing the analysis plan resides in the knowledge base until the user clicks on the "reset" active region. This network may be converted into a command procedure in one of the two currently supported languages, SDAS/IRAF or MIDAS. The conversion process is procedural and is initiated on user request: the procedure simply traverses the network and sends each primitive task a "format" message after ensuring that all of its predecessors have already been processed. No optimizing of the traversal is performed, but this would certainly be possible (for example, to minimize the disk space required at any one time for storage of intermediate files).

III. FUTURE DIRECTIONS

Although the prototype system accomplishes the goals listed in Section I, there are a number of areas which require extension before a useful system could be considered operational even in the limited domain of CCD calibration. This section first addresses the shortcomings of the current prototype, then discusses directions in which the prototype should be extended.

- The current system does not represent the effects of analysis operations on the data. Intermediate files are represented simply as symbolic names. This means that it is not possible to write rules that reference the *properties* of partially processed data in their preconditions. Adding this capability will require the creation of frames with slots that describe how a particular analysis task modifies the data in an image file. Each intermediate data file would be represented by such a frame.
- The current rules for associating calibration images with science images are trivial. However, it would be straightforward to incorporate more realistic criteria for this process.
- There are no tasks defined to fix cold columns and perform other data repair operations. These can, however, be straightforwardly added to the current structure.
- Graphics displays or system status are currently limited to the specific science frames that are part of the test dataset. It would be useful and not difficult to allow the creation of monitoring graphics "on the fly" at user request.

The current prototype will be extended by incorporating rules representing more realistic calibration criteria, and by excercising the system on a real collection of science and calibration images.

There are several topics that must be addressed for the prototype to be useful beyond its current limited domain:

- **reasoning about data contents**: at present the system is based entirely on reasoning based on a *description* of data files, but this is clearly inadequate for more involved problems (and even to handle certain subtleties associated with the CCD calibration domain). It is essential to include a capability to obtain information about the *contents* of data files to proceed further in this direction. This will require one of two approaches: most desirable is the ability of the ES to initiate realtime tasks on the host processor to compute and return information about the data to the ES, such as statistics on pixel values, the results of fitting trial point-spread functions, etc. Alternatively, it may be possible to iterate with the host by preparing a batch procedure to generate this information; the ES would then have to preserve its current state until the results return.
- **user interaction**: the experienced eye of an astronomer is undoubtedly essential in many aspects of astronomical data reduction. While this is related to the problem of reasoning about data contents, it adds the further complication of ensuring that the user may, upon demand, display raw and intermediate files and provide input to be used by the ES in subsequent processing.
- **goal specification**: planning an analysis process is strongly driven by the ultimate goals of the analysis, in contrast to general CCD calibration domain where the same steps are followed for virtually all planned uses of the data. A facility must be provided to allow the astronomer to specify high level goals which will then be used by the ES to guide further planning.
- **reasoning about astronomical objects**: the prototype system contains no knowledge of the properties of astronomical objects, but this is clearly an area that would be be extremely fruitful to pursue.
- **practical issues**: certain practical problems must be solved before productive use of ES can be expected to become routine. For example, the speed of rule processing tends to be slower than procedural code and may become a serious limitation for a large operational system. Another problem is that certain data descriptors that are required to automate processing are not recorded in a standardized way at all telescope sites (this problem is not peculiar to ES processing!).

IV. CONCLUSIONS

The prototype described in this paper demonstrates the feasibility of applying expert system technology to the problem of astronomical data analysis. The major goals of the prototype were accomplished in a rela-

tively short time by taking advantage of the unique capabilities of current commercially available hardware and software systems. It is clear that the approach holds great promise for the future. Major problems to be addressed by the continuation of this project include:

- processing of large volumes of data via intelligent automated analysis; of particular interest is the integration of ES with modern pattern recognition and object classification techniques
- exploiting the expected properties of data to guide the development and execution of reduction procedures
- automation of tedious but necessary steps in data reduction
- minimizing the problem of proliferation of analysis systems and languages by providing a language-independent high-level user interface
- capturing techniques used by experts and making them available to less experienced users

---

The author is grateful for the hospitality and support of the Space Telescope European Coordinating Facility (ST-ECF) and the European Southern Observatory (ESO) where this work was conducted. Particular mention should be made of Rudi Albrecht, Hans-Martin Adorf, Roberto Rampazzo, and Dietrich Baade (ST-ECF), Glenn Miller and Mike Shara (Space Telescope Science Institute), and Don Rosenthal (STScI and NASA Ames Research Center, Intelligent Systems Division) who have provided many useful comments and suggestions on this work. Intellicorp GmbH (Munich) provided prompt technical support for the KEE system, and Symbolics GmbH (Eschborn/Ts) provided access to the lisp machine.

REFERENCES

1. *Building Expert Systems*, ed. Hayes-Roth, F., Waterman, D., and Lenat, D., (Addison-Wesley: 1983)

2. *Rule-Based Expert Systems*, ed. Buchanan, B., and Shortliffe, E., (Addison-Wesley: 1985)

3. Thonnat, M., Granger, C., and Berthod, M., IEEE Transactions on Computer Vision and Pattern Recognition, 206-208 (1985)

4. Bernat, A., and McGraw, J., Steward Observatory Preprint No. 694 (1986)

5. *Science Data Analysis Software User's Manual*, Space Telescope Science Institute 86053A (1986)

6. Shames, P., and Tody, D., "A User's Guide to the IRAF Command Language", Space Telescope Science Institute SDAS/IRAF Workshop (December 1986)

7. Crane, P., Banse, K., Grosbøl, P., Ounnas, C., and Ponz, D., "MIDAS", in *Data Analysis in Astronomy*, ed. di Gesú et al. (Plenum Press: 1985)

Section D

Expert System Tools/Techniques

Expert Systems Tools for Hubble Space Telescope Observation Scheduling

Glenn Miller [1]
Astronomy Programs, Computer Sciences Corporation

Don Rosenthal
NASA Ames Research Center

William Cohen[1]
Astronomy Programs, Computer Sciences Corporation
and

Mark Johnston
Space Telescope Science Institute [2]
3700 San Martin Drive
Baltimore, MD 21218

Abstract

Construction of an efficient year-long observing program for the Hubble Space Telescope (HST) requires the ordering of tens of thousands of proposer-specified exposures on a time-line while satisfying numerous coupled constraints. Although manually optimized planning can be performed for short time periods, routine operations will clearly require that most of the planning be done by software. This paper discusses the utility of expert systems techniques for HST planning and scheduling and describes a plan for development of expert system tools which will augment the existing ground system. Additional capabilities provided by these tools will include graphics oriented plan evaluation, long-range analysis of the observation pool, analysis of optimal scheduling time intervals, constructing sequences of spacecraft activities which minimize operational overhead, and optimization of linkages between observations. Initial prototyping of a scheduler used the Automated Reasoning Tool (ART) running on a Texas Instruments Explorer Lisp workstation.

[1]Staff Member of the Space Telescope Science Institute

[2]Operated by the Association of Universities for Research in Astronomy for the National Aeronautics and Space Administration

1 Introduction

Scheduled for launch by the Shuttle in late 1988, the Hubble Space Telescope (HST) is an observatory of unprecedented capabilities. From a vantage above the bulk of the Earth's atmosphere, its scientific instruments will be able to observe farther and over a wider spectral range than any other telescope. During the design lifetime of 15 years, its complement of six scientific instruments should dramatically expand knowledge in essentially every area of astronomy. The Space Telescope Science Institute (STScI) is responsible for conducting the science operations of the HST, ranging from proposal solicitation, through planning and scheduling, realtime operations, data processing, archiving and user support [1].

Astronomers throughout the world will use the HST. A year's observing program for the observatory will consist of about 30,000 exposures on approximately 3000 celestial targets. In executing these exposures, a large number of constraints (scientific, hardware, orbital, thermal, etc.) must be satisfied. Additionally, it is crucial to maximize the scientific return by having an efficient schedule of observations. These factors make HST planning and scheduling a challenging problem.

Several aspects of expert systems are attractive for the construction of tools to aide scheduling, and the purpose of this paper is to describe a plan for the development of expert systems tools which would augment the existing ground system software. The next section presents an introduction to the HST planning and scheduling problem, including the major constraints and efficiency issues. Section 3 describes the tools and their planned development, including a justification of an expert systems approach.

2 The Problem of HST Planning and Scheduling

In order to use the HST, an astronomer submits a scientific observing proposal to the STScI. The proposal forms are "astronomer-friendly" in that they allow the proposer to describe what data must be obtained without becoming needlessly involved in the details of how the spacecraft and ground systems will implement the observations [2].

Based on the advice of a peer review committee of experts in a range of astronomical disciplines (the Time Allocation Committee), the Director of the STScI selects which proposals are to be awarded HST observing time [3]. Competition for HST time will be keen as the oversubscription ratio (number of submitted to accepted proposals) is expected to be at least three (typical for large, ground-based telescopes) and may approach a factor of ten. Of the 1000-2000 proposals submitted yearly, only about 200-300 will be accepted for execution.

While scientific merit is the most important selection criterion, the selection process must take into account various *resources* which are in limited supply, e.g. unocculted viewing time, power, communications, etc. In other words, a mixture of proposals which can actually be implemented by the spacecraft and ground systems must be chosen. It is important to note that *no scheduling* of proposals is performed at this stage: selection is based on estimates of the resources used by each proposal in comparison to the total estimated resources available in the coming year. Calculation of resource consumption by a proposal is uncertain at this stage since it is a function of both the time of observation and what other observations

are on the timeline (refer to the constraints listed in the next section). Resource usage estimates are calculated using an expert system described in [6]. Likewise, the total amount of resources available is uncertain since it depends on the activities to be scheduled and the possible carryover of high priority observations from the preceeding cycle of observations. The decision process for proposal selection is aided by a natural language database query system [7].

The result of this selection process is a set of proposals to be executed in the coming year, and is therefore the input to the HST planning and scheduling process. Accepted proposals are called *programs* and are allocated to three scheduling priorities: high, medium and supplemental. Barring unforseen technical difficulties, all high and medium programs will be executed, and together they account for approximately 70% of the estimated available observing time. The essential difference between high and medium is that greater emphasis is placed on completion of high priority observations (e.g. medium observations may be rescheduled to accomodate rescheduling of a high priority observation). The supplemental programs comprise a pool used to fill out the schedule; the choice of a particular supplemental program is likely to be based on operational constraints. Exposures in the supplemental pool oversubscribe the available time (and thus there is only a moderate probability that any particular supplemental program will actually be executed).

Following the selection process, proposers supply additional details required for scheduling and make any modifications imposed during selection (e.g. a decrease in the amount of observing time or number of targets). Next, the observing programs are transformed from the proposal format into the parameters required by the planning and scheduling system, effecting the translation from scientific objectives ("what") to hardware and software implementation ("how").

The processing of HST observing proposals is aided by the Proposal Entry Processor (PEP), which includes several systems utilizing AI techniques: Transformation from scientific proposal format into planning and scheduling system parameters is accomplished using an expert system [4], [5], as is the calculation of resource usage [6]. The selection process is supported by a natural language database query system [7]. Examination of observations for scientific duplication also makes use of an expert system [6].

At this point, the scheduling process begins with a pool of 200-300 programs encompassing tens of thousands of exposures on a few thousand targets. The overall goal of this process is to execute all the high and medium priority observations and as many supplemental observations as possible. Many observatories schedule by allocating blocks of time to observers, who then perform their own scheduling within that time (often scheduling in real time). HST scheduling takes a different approach: in the absence of scientific constraints to the contrary, exposures will be scheduled at times which increase the overall efficiency of the observatory. As a result, observations from any particular program may be spread over several months.

Science scheduling for HST is a two step process:

1. A time ordered sequence of exposures (called a *calendar* or *timeline*) is created from the program pool. The generation of timelines is currently envisioned to be a iterative process of increasing detail and density. High priority and time critical observations will be scheduled on a 6 month to 1 year timeline. Next, month long timelines will

be identified and populated with more observations, followed by week long timelines, etc.

2. Given a timeline, high level spacecraft instructions are attached to the activities on the timeline. The output of this process is a Science Mission Specification (SMS), and can be thought of as the "assembly language" which drives the HST. From the standpoint of the HST ground system, the purpose of the STScI is to produce the SMS.

To avoid confusion, it should be noted that for the HST domain, the terms "planning" and "scheduling" have switched meanings compared to their usual meanings in AI literature. HST "planning" refers to the process of scheduling activities on a timeline, while HST "scheduling" refers to the process of ordering spacecraft instructions to accomplish activities on the timeline. In practice, these terms are often used interchangably.

The SMS is sent from the STScI to the Payload Operations Control Center (POCC) at Goddard Spaceflight Center where it is checked for errors and constraint violations which would affect the health or safety of HST or the instruments. From the SMS, the POCC prepares the actual binary command loads for the two onboard computers which control HST. Some iteration of the SMS occurs between the STScI and the POCC. The principal reason for this is the process of obtaining communications links. The POCC takes requests for Tracking and Data Relay Satellite (TDRS) links from the SMS and passes them onto the TDRS Network Control Center. Some links will not be available due to higher priority users (e.g. the Shuttle or other satellites). The POCC notifies the STScI of unobtainable links, and the timeline must be modified by the STScI, either by use of an onboard tape recorder or by rescheduling the observation.

2.1 Constraints and Operational Ground Rules

There are a number of considerations which influence the planning and scheduling process. These range from hard constraints, which if violated, may result in damage to the spacecraft, to operational ground rules which result in increased efficiency or flexibility.

Proposer specified constraints: In order to satisfy the scientific objectives of the observing program, astronomers can specify various relationships between exposures, for example:

- Time of observation: Although most exposures can be accomplished at any time, others must be accomplished within a certain time interval. Exposures with a narrow time window are referred to as *time critical.* Observations of periodic celestial phenomena (e.g. variable stars) may be constrained to certain phases.
- Precedence: before and after links between exposures
- Grouping: exposures which must be executed as a group, not necessarily in a particular order and without interruption by other activities.
- Priority and completion levels: In addition to the overall priority of a program (set by the Time Allocation Committee), a proposer may prioritize exposures within a proposal. Additionally, a level of completion may be specified, for example, 25% of

the targets must be observed for any to be useful, coverage of 50% of the targets will be optimal, but coverage of more than 75% may not significantly improve the results. This capability is especially important for supplemental priority and multi-year programs.

- Conditionals and selects: The HST observing proposal forms contain two constructs which allow the proposing astronomer considerable flexibility in specifying an observing program: "conditional" and "select". The first marks exposures which are contingent upon some condition, e.g. on the results obtained from some other exposure in the observing program or perhaps the results obtained from a ground based observation. Conditional exposures will not be scheduled until the proposer notifies the STScI that the condition has been satisfied. (This is in contrast to real time decisions which are handled by another mechanism). "Select" identifies alternative sets of exposures from which the proposer will select one or more for actual execution. As with conditional exposures, exposures contained in a select set will be placed on a timeline only after the proposer makes a final decision.

- Dark time: some exposures can only be executed when the HST is behind the Earth's shadow, shielded from the glare of the Sun.

- Orientation: certain observations require a particular orientation of HST in order to align a spectroscopic slit or polarization filter with features of a target. This factor is closely tied to power and thermal balance discussed below.

Realtime interactions: HST and the ground systems are designed to operately largely in a preplanned mode, e.g. the SMS must be complete three days before observations begin. However, the system is designed to support a certain level of realtime interaction. Examples include changing a filter in an instrument, a small angle maneuver for target acquisition or choosing among fully preplanned alternative observations. Realtime commands which would result in unplanned slews or major changes in instrument modes are not allowed. In general, realtime interaction places a large demand on spacecraft, communications and ground system resources, and its use must be carefully planned.

Orbital constraints: Many orbital factors exert a strong influence on the observing schedule. HST will occupy a low earth orbit (500 km), so a target on the orbital equator is occulted (blocked) by the Earth for about 39 minutes out of each 95 minute orbit. Long exposures will typically be implemented as a series of shorter exposures separated by Earth occultations. Targets within a few degrees of the orbital poles are not occulted by the earth, so this ***continuous viewing zone*** may be used for long observations which cannot be interrupted (if the target lies within this zone).

To avoid damage to the spacecraft and instruments, the HST cannot normally point to within 50 degrees of the Sun, nor can certain instruments view the bright Moon or Earth. In contrast, some instruments will use the bright Earth for calibration of the instrumental signature.

Another orbital factor is the South Atlantic Anomaly (SAA), a region where the Van Allen radiation belt dips into the orbit of HST. Noise induced by the charged particle radiation will prevent observations with most instrument modes in the SAA. However, one instrument (the High Speed Photometer) will be used to observe and map the extent of the SAA.

Power and thermal balance: Electrical power and a controlled distribution of temperature within the spacecraft are two closely related constraints. Power is generated on HST by a set of solar cells located on the "wings", and is stored in batteries. Instruments and other equipment can be damaged by extremes in heat or cold, and a proper thermal balance is accomplished by passive insulation, and active heating and cooling elements. In order to keep the solar cells pointed toward the Sun and to maintain the proper thermal balance, the V1-V3 plane of ST must normally be within 5 degrees of the Sun (V1 is the line of sight of the telescope, the V2 axis contains the solar arrays, while V3 is directed outward from the top of HST). Excursions as far as 30 degrees off this nominal roll are allowed as long as the batteries are allowed to properly recondition afterwards. Although most scientific observations will not require a particular orientation of HST relative to the sky (and thus a particular roll angle relative to the Sun), observations with certain instruments will (e.g. slit spectroscopy and polarimetry). As the solar cells and batteries age, their capacities will diminish and power constraints may become even more severe.

Guide stars: The HST uses Fine Guidance Sensors to lock onto two *guide stars* in order to compensate for long period drifts in the guidance system's gyroscopes. Although ample guide star pairs are expected to be available for most regions of the sky, certain regions will contain very few stars (and will restrict scheduling opportunities). Additional constraints arise when one pair of guide stars must serve two or more instruments (e.g. a target acquisition using a camera followed by an obervations with a spectrograph). Guide star acquisition and lock requires several minutes, so guide star acquisitions should be minimized.

Scientific instruments: Cycling the scientific instruments from a standby to operate mode will require careful planning. Power constraints limit the number of instruments which can be collecting data simultaneously and the time to bring an instrument from standby to operate can be as long as 24 hours. Certain instruments and modes will require a set of calibration observations each time they are brought to operate mode.

Slews: Changing the orientation of HST to point to a new celestial target (called *slewing*), is a relatively slow operation. HST is only slightly faster than the minute hand on a watch, accomplishing a 90 degree slew in about 13 minutes. Note that optimization of slews alone is an NP-complete problem and is only a subset of the HST planning and scheduling problem.

Communications: All communications with HST (command uplinks and data readouts) is via the Tracking and Data Relay Satellite System (TDRS) which serves multiple users. As a consequence, HST planners must negotiate communications contacts two weeks in advance, and not all requested contacts may be available. Additionally, the HST orbit is low enough that during a portion of each orbit the Earth blocks one or both TDRS satellites. In each orbit, HST is limited to 20 minutes of high speed downlink contact. When a TDRS is not available for readout, onboard tape recorders can save science and engineering data for later playback. However the tape recorders have limited storage and lifetime, so their usage must be optimized.

Calibrations: As with any scientific instrument, HST instruments require calibration observations in order to produce meaningful scientific results, e.g. flat-field observations, dark count determination, wavelength calibrations. Although some calibrations will be routinely performed, others are dependent upon which exposures will actually be executed (e.g. high accuracy calibrations or calibration of seldom used modes). Some calibrations can be performed during slews (e.g. observations of internal light sources), while other will

require observations of standard reference targets. Most calibrations must be accomplished within a certain time of the science observation. Routine instrument calibration is the responsibility of the STScI.

Straylight and exposure times: Since many HST observations will be of extremely faint objects, contamination by straylight can be an important factor. Sources of straylight are time variable and include the Sun, Moon and Earth, and sunlight scattered by dust in the solar system (zodiacal light). Any of these sources may drastically increase the exposure time required to reach a specified signal to noise ratio.

Adjustment of exposure times: Given a fixed amount of straylight, in most instances, it is scientifically acceptable to adjust exposure times by small amounts (typically 10%) to fit within an available space (shorter or longer).

Schedule disruptions: Although HST operates largely in a preplanned mode, disruptions to the schedule will occur for a variety of reasons. The most welcome disruptions are *targets of opportunity*, which are rare, important astronomical phenomena requiring immediate attention (e.g. a supernova). The ground system should be able to respond to targets of opportunity as often as once a month, and be able to begin observations within a few hours of notification. Other schedule disruptions will result from equipment failures, spacecraft anomalies or loss of communications contacts. These will occur with little or no advance warning. It is important to be able to build schedules which minimize the sensitivity to disruptions (perhaps placing the HST in a checkpoint state at periodic intervals) and to be able to re-plan or patch schedules rapidly.

Insight into the planning process: It is important that the STScI operations staff have an understanding of the planning process, even in the case of automatically generated schedules. This includes explanations of why a particular observation was scheduled at a particular time and why it cannot be scheduled at another time.

The above enumeration of the constraints should make it clear that there are numerous constraints which have complex interactions, and that the number of feasible alternative timelines is so enormous that human planners cannot reasonably evaluate even a few hundred within the time limitations imposed by HST operations.

2.2 Current Ground System

HST planning and scheduling utilizes the Science Operations Ground System (SOGS) Science Planning and Scheduling System (SPSS), which was developed by TRW, Inc.

Within SPSS, the proposal data is represented by the following data structure:

- An "Exposure" is a single instrument operation, usually resulting in the acquisition of a single data set, e.g. a camera frame or a spectrum.
- An "Alignment" is a set of exposures that can be taken without moving the telescope (usually a single instrument and a single target, sometimes multiple instruments and multiple targets).
- An "Observation Set" is a set of alignments that can be performed without affecting the guidance system (that is, without reacquiring guide stars).

- A "Scheduling Unit" is a set of observation sets and is the smallest schedulable entity. Scheduling units can draw observation sets from any proposal (within an observation set, all alignments and exposures must come from the same proposal).
- Scheduling units may be linked (via before/after time intervals).

Note that this representation imposes a certain structure on the observations, generating constraints in their own right.

The first step in using SPSS is to populate the scheduling unit hierarchy. For most proposals this is handled automatically by PEP Transformation. Special cases can be populated manually either using PEP or SPSS functions. Next, the planner creates a *candidate and calendar (C&C) list.* The calendar is a time interval to be populated, while the candidates are scheduling units available to be placed on the timeline. Planners can manually add or remove scheduling units (with constraint checking performed by SPSS). SPSS provides functions which, given a candidate, find the best time to schedule it, or given a time, find the best candidate for that time. ("Best" is evaluated by a cost function which takes into account factors such as scheduling priority and slew time). In addition to the manual planning capabilities, an automatic scheduler is under development. Based on a greedy algorithm, it will find the candidate which best fits the next time on the calendar.

Once a timeline is populated with activities (observations, instrument reconfigurations, slews, etc.), high level spacecraft instructions are attached to the activities and then an SMS is generated for transmission to the POCC.

As a result of preliminary operations and testing of SPSS and increased experience with the planning and scheduling problem, STScI staff have identified a number of enhancements needed to make effective use of HST. Performance of the system is a major concern. In the operational era it must be possible to generate a day's SMS in less than one day of effort, averaged over all aspects of planning and scheduling, staff and computer resources. Current performance falls significantly short of this goal. Automation of labor-intensive and routine tasks will clearly benefit performance.

Currently there exist no tools to help planners in matching candidate scheduling units with calendars. Given the large pool of programs, tools are needed to select candidates from the pool which fit a specific calendar and to select calendars which would be appropriate for a specific program (or portion of a program).

Scheduling units must be created before they can be placed on a timeline, including the sequencing of individual exposures and spacecraft activities. Currently, SPSS places the activities on a calendar in the order specified with no attempt at re-ordering exposures to better fit the orbital events at that time (e.g. occultation, day/night, etc.). Such a fixed sequence will be non-optimal in all but the most fortuitious of circumstances and will therefore decrease the efficiency of HST. The current system does allow the planner to iteratively "hand craft" a scheduling unit and its components based on its place in a timeline, however this has an obvious impact on performance, and if the SU is ever rescheduled, the results of the effort are wasted.

Several of the proposer specified constraints can be implemented only by manual procedures, including proposer priority, completion levels, conditionals and selects. The current system also provides no assistance in determining what calibrations are required for a particular

timeline. Automatic placement of proper calibrations when scheduling observations, and avoidance of redundant calibrations is highly desirable.

Straylight and variable exposure times are also difficult to handle in the current system. Observations can be flagged as requiring orbital day or night execution and it is possible to make manual adjustment of the Sun, Moon and Earth avoidance limits, but a more automatic method with a finer degree of control is required. Expanding or trimming exposures by small amounts to fit within an available time slot can only be accomodated by a manual trial and error process.

3 Development of Tools for Planning and Scheduling

The previous section sketched the problem of HST scheduling and highlighted capabilities which are lacking from the current ground system. This section presents an approach to solving these problems using AI techniques.

Work towards ground system enhancement is directed along two lines: 1. increasing the performance, reliability, maintainability and functionality of existing SPSS software, and 2. creating new tools to augment the existing software. The former effort is largely directed at science instrument instruction management and SMS generation, while the latter is directed at scheduling and is the focus of the present paper. These two approaches will be carefully integrated to provide a coordinated effort for ground systems enhancement.

3.1 The Environment

Experience with Transformation and other rule-based software in PEP [4], [5], [6] has shown the advantages of a rule-based expert systems approach, especially with regard to rapid development, functionality, performance, adaptability of code to changing requirements and quick turnaround time for changes and enhancements. It is natural then that an expert system approach be utilized in the development of the proposed planning tools. It is important to note however, that expert systems are not a panacea for this problem. In particular, judicious use of procedural algorithms will be extremely useful in pruning alternatives before application of expert system rules.

OPS5, the computer language used for implementation of PEP rule-based software, is a language with which we have had great success in the past. However, prototypes in OPS5 along the lines of the proposed planning tools have revealed limitations in the language for such tasks, additionally, the Vax OPS5 environment provides no direct support for graphics output and lacks program development tools.

Preliminary investigations into planning tools have shown that a powerful knowledge-based development system which supports *hypothetical reasoning*, a *combination of forward and backward chaining* rules, and *frame-based data representation* which incorporates inheritance is needed for such a task. In addition, strong support for *graphics-oriented* programmer and user interface is required.

Forward chaining inference systems are appropriate for problems where there are many equivalently acceptable solutions (as in Transformation, design problems, and planning

problems in general). Forward chaining rulebased systems are very strictly data-driven: given a starting state, conclusions are drawn, and actions taken. Backward chaining allows the program to reason from desirable consequences to the causes which produce them.

Frame-based representation is an extremely powerful method of representing relationships between data. Many of the important characteristics of planning data are relationships, for example, exposures related in time, position, or due to membership in a scheduling hierarchy. A frame can be used to define a class of data, and another frame to define a subclass or refinement of that data. Subclasses automatically inherit the representations of the parent classes, with additions or changes as specified by the programmer. For example, one class might define exposures. A subclass of exposures with the Wide Field/Planetary Camera (WFPC), would inherit all characteristics of exposures, with specialized characteristics of that camera (e.g. power requirements). A sub-subclass might define types of WFPC exposures (e.g. data collection or target acquisition) which would inherit characteristics of exposures, WFPC exposures, and add characteristics such as realtime link requirements. Such expressiveness obviously speeds development, and aids maintenance and enhancement.

Another important requirement is the ability for hypothetical reasoning. This creates an alternate "world view" which is different from an existing set of facts in one or more ways. Hypotheticals have an obvious and natural application to scheduling problems in that they allow the evaluation of the effects of scheduling a proposal at different times. Rules can be written which check hypotheticals for contradictions, constraint violations, and inefficiencies, and which then mark that state as not worth further consideration. This limits the effort used in searching unprofitable alternatives, without the need for backtracking. Rules can also reason across multiple hypothetical states of the program, optionally merging several such states if appropriate (e.g. combining two partial timelines).

A fully integrated graphics interface is important for two reasons: First to support a rapid development effort (graphical browsing of the rulebase as well as the tracing of the program state during execution), and second, to provide a product with a powerful user interface. Graphic objects on the screen can be mouse sensitive, and changes to the display can automatically affect the rulebase and/or working memory. Thus, the user can play out "what-if" scenarios, e.g. by moving observations around on the timeline and having the program continue from the new state of the timeline data.

Development of an environment with the above capabilities is clearly a large task, so our approach was to look towards commercial products. A detailed survey of the market identified two advanced expert system environments which are suitable for initial investigations: ART (Advanced Reasoning Tool) from Inference Corporation, and KEE (Knowledge Engineering Environment) from Intellicorp. We have obtained a license for ART and have begun prototyping the tools described below; KEE is not yet available to us. A Texas Instruments Explorer Lisp workstation is the host for the development and is networked via TCP/IP over Ethernet to the DEC Vax computers which host the PEP and SOGS systems.

3.2 The Approach

As a first step towards evaluating the utility of AI tools to augment the ground system, a graphical plan evaluation environment is being developed. It will provide the basic functions of placing an activity on a timeline and removing an activity from a timeline. Calculation

of scheduling constraints will be fully integrated into the plan evaluator, including display of scheduling windows and display of constraint violations which prevent activities from being placed at a selected time. (Although calculation of constraints and scheduling windows is an algorithmic problem, application of constraints will benefit from a frame-based representation. Additionally, these constraints will play an important role in pruning the problem search space before application of expert systems rules.) Due to the complexity of the problem, considerable effort will be placed on the user interface, e.g. activities will be mouse sensitive to allow display and editing of their parameters, and users will be able to zoom and pan on the timeline (see [9] for a description of a related system).

The graphical plan evaluator is an important tool for both the software developers and operations staff. It will aid in capturing the basic domain knowledge needed by the developers in determining high-level approaches to scheduling and it will also serve as a testbed to try different scheduling algorithms and heuristics. For operations staff, even a prototype plan evaluator which allows the ability to rapidly develop alternative schedules will aid in the development of schedules and operational procedures. In particular, the plan evaluator will be useful in development of long range plans and in the determination of calibration requirements.

Although STScI operations staff have many years experience with spacecraft scheduling, our understanding of the problems associated with HST is not yet complete. An important part of the development of these tools will be an approach which allows the continuing experience of the operations staff to be reflected in the tools development.

After the development of the plan evaluator, the tools will be extended to handle:

- evaluation of exposures to identify preferred execution times (including such factors as sensitivity to background light)
- evaluation of "clumping" exposures that should be scheduled together
- introduction of plan evaluation measures that can be used to compare alternative timelines for efficiency.

This extension will allow operations staff to aggregate exposures into Scheduling Units, and recommended times for execution.

As experienced is gained in the implementation and use of these tools, the emphasis of the work will focus on integration of the tools into the operational environment. This includes integration with PEP transformation and the P&S software and data structures, e.g. generation of SPSS data records and scheduling commands to place them on the C&C list at the appropriate times. The tools will also be extended to include a fully automatic mode, based on guidelines and heuristics discovered as a result of working with the interactive system.

To conclude this section, we describe an initial scheduler prototype which has already been implemented in ART. The prototype handled multiple constraints, including guide star acquisition, Earth, Moon and Sun occultations, SAA avoidance, variable slew times, instrument usage (including scheduling a transition from hold to operate), exposure precedence links and time critical exposures. The input exposures were taken from the Design Reference Mission ([10], a manual exercise in HST scheduling), and are therefore realistic set of

science operations. The prototype scheduled the DRM's first week of observations (total of 75 exposures) in 45 mintues. The prototype consisted of 19 ART rules, supported by 9 Lisp functions. (Calculation of the orbital events and target visibility windows was performed using a separate package of Fortran programs developed previously.) Development of the prototype took one person two weeks. This exercise clearly demonstrated the power of the expert systems approach for HST scheduling: development was rapid, the language is expressive and powerful and well suited to constraint checking and hypothetical reasoning.

4 Conclusions

In this paper we have described the problem of planning and scheduling science observations for the Hubble Space Telescope and how the numerous, coupled constraints make for a difficult problem. Several aspects of expert system development environments are attractive for the construction of tools which will augment existing ground system capabilities, including the rapid development cycle, adaptability of code to changing requirements and powerful methods for representing and reasoning with knowledge. Additional capabilities provided by these tools will include graphics oriented plan evaluation, long-range analysis of the observation pool, analysis of optimal scheduling time intervals, constructing sequences of spacecraft activities which minimize operational overhead, and optimization of linkages between observations. A plan for the development of enhancements was discussed and the results of initial prototyping was presented.

References

[1] *SO-07 Science Operations Concept*, 1983, Space Telescope Science Institute, SCI 82012B

[2] *Call for Proposals and Proposal Instructions*, 1985, Space Telescope Science Institute, STScI SC-02

[3] *SO-05 Proposal Solicitation and Review*, 1985, Space Telescope Science Institute, STScI 85051A.

[4] Rosenthal, D., Monger, P., Miller, G., and Johnston, M. 1986, "An Expert System for Ground Support of the Hubble Space Telescope", *Proceedings of the 1986 Conference on Artificial Intelligence Applications*, NASA Goddard Space Flight Center

[5] Lindenmayer, K., Vick, S. and Rosenthal, D. 1987, "The Operational Phase of an Expert System for the Hubble Space Telescope Ground Support", *Proceedings of the 1987 Conference on Artificial Intelligence Applications*, NASA Goddard Space Flight Center

[6] Jackson, R. 1987, "Expert Systems Built By The 'Expert': An Evaluation of OPS5", *Proceedings of the 1987 Conference on Artificial Intelligence Applications*, NASA Goddard Space Flight Center

[7] Hornick, T., Cohen, W., and Miller, G. 1987, "A Natural Language Query System for Hubble Space Telescope Proposal Selection", *Proceedings of the 1987 Conference on Artificial Intelligence Applications*, NASA Goddard Space Flight Center

[8] *SOGS Science Planning and Scheduling System (SPSS) Functional Requirements Specification, STE-17*, 1983, NASA Goddard Spaceflight Center.

[9] Taylor, W. 1987, Ames Scheduler and Planner User Interface Manual, NASA Ames internal document

[10] Sherrill, T.J. 1983, *Design Reference Mission*, Lockheed Missles and Space Corporation.

Toward an Expert Project Management System

by

Barry G. Silverman*+

Arthur Murray*

Coty Diakite+

Kostas Feggos*

*- Institute for Artificial Intelligence,

The George Washington University,

Washington, D.C. 20052

+- IntelliTek, Inc., Rockville, Md.

1.0) Introduction and Overview

The purpose of the research effort presented here is to prescribe a generic reusable shell that any project office can install and customize for the purposes of advising, guiding, and supporting project managers in that office. The prescribed shell is intended to provide both: (1) a component that generates prescriptive guidance for project planning and monitoring activities, and (2) an analogy (intuition) component that generates descriptive insights of previous experience of successful project mangers. The latter component is especially significant in that it has the potential to: (a) retrieve insights, not just data, and (b) provide a vehicle for expert PMs to easily transcribe their current experiences in the course of each new project they manage (i.e. to act as the Corporate Memory).

For the past several years the principal author has conducted psychological, behavioral, and cognitive studies of expert project managers' thought processes for the purposes of deriving a model suitable for translation into an expert system. The model is based on the process of diagnosis and analogical reasoning as described above and in sections of this paper. This model is based on the study of 21 employees of NASA, numerous employees of the U.S. military, historical case studies from the Space station and Space Telescope Programs and papers of 16 famous inventors (e.g., Ben Franklin's diaries, to mention one) as documented in earlier reports. It is expected that the successful implementation of the model and the integration of the analytical and analogical components will result in many new innovations including special-purpose expert system generators, which would represent a new phase in the maturation of Expert Systems technology for project management applications.

1.1) Technical Objectives

The focus of this paper will be to report on the preparation, conduct and results of an experiment to prove/disprove the premise that an expert project management system can be configured that will improve/expand the ability of a manger to perform project planning and monitoring. This experiment has been designed with the intention of accomplishing the following three objectives:

(1) Construction of a Simplified Prototype containing a Project Management (PM) Subsystem, an analogical reasoning inferencing mechanism and the associated knowledge bases.

(2) Exploration of Eleven Key Research Questions relating to the nature of an expert project management system (EPMS) environment.

(3) Evaluation of the Prototype and Recommendation of Design Guidelines for EPMS Version 1.

The evaluation of the prototype will consist of a system performance evaluation based on snapshots and backtracing of actual EPMS runs, and on comments/suggestions by 17 experts presented with three exemplary EPMS user sessions. The insights obtained from these evaluations will be used to formulate design guidelines for a working Version 1 system, which is expected to perform beyond the current limits of expert system shells, and exhibit the characteristics of an expert system kernel or generator.

1.2) Report Organization

This report will present in succession, the framework and results of the activities aimed at the accomplishment of the three technical objectives. The next section deals with the knowledge elicitation process and the resulting framework for the EPMS generator. Section 3 contains a top-level description of the prototype, and the evaluation of the prototype will be presented in section 4. The last section presents the conclusions reached and outlines of planned future developments.

2.0) PM Knowledge Elicitation

This section describes the concept of EPMS that evolved over dozens of knowledge collection sessions. In each session, feedback from domain experts was solicited by giving demonstrations and/or functional descriptions of EPMS: i.e. its goals, its conceptual design, and the types of sessions a user would encounter.

A common observation among the experts was the need to implement EPMS within any given two-level hierarchy, in order to be compatible and supportive of existing organizational boundaries and lines of communication (see Figure 1). Figure 1 also reflects the perception that the manager probably will not be the principal user of EPMS and that is more realistic to expect an Executive Assistant to assume the user role, and to expect the manager and submanagers to use EPMS either indirectly through the Assistant or occasionally themselves.

A significant observation made during the knowledge elicitation sessions was the presence of a wide diversity of needs for stand-alone expert system-based project management support tools. One of the managers interviewed presented a list of some of the possible areas for ES support (see table 1) and indicated that this list was by no means exhaustive. Furthermore, there was found to be an overwhelming preponderance of existing subsystems, data bases, MIS, DSS, etc. which would require direct interfacing to an integrated EPMS Kernel, or would

Table 1: Summary of Possible Stand-alone PM Expert Systems

Estimating

Project Selection

Conceptual Estimating
- Parameter Estimating
- Life Cycle Costing
- Value Engineering
- Integration with CAD
- Historical Cost Data Management
  - Collection
  - Maintenance
  - Tailoring

Check List
- Contract Document
- Site Inspection

Start Procedures Manuals
- Quantity Survey
- Pricing
- Securing Material and Subcontractor Prices

Project Conceptual Planning
- Equipment Analysis
- Crew Analysis
- Cash Flow Analysis

Bid Preparation
- Sub Bid Analysis
- Sub Bid Scratch Estimates
- Material Price Analysis
- Markup Analysis
- Bid Adjustments

Change Order Estimating
- Impact Analysis
- Change Condition Pricing

General Conditions Development
- Item Selection
- Pricing

Insurance Analysis

Cost and Time Control

Planning
- Activity Breakdown
- Logic Definition
- Duration Analysis
- Contingency analysis

Input Data
- Collections
- Checking

Activity and Project Status
- Analysis
- Overrun Projections
- Problem Flagging
- Problem Diagnosis
- Remedy Recommendations

Changed Condition
- Identification
- Impact Evaluation
- Recommendations for Execution
- Notifications of all Parties Involved

Productivity Analysis

Progress Payment Application Preparation

Invoice Checking

Material Tracking

Penalty/Liquidted Damages Evaluation

Site Planning and Management

Site Layout

Materials Handling

Temporary Facility Requirements

Site Management Staffing

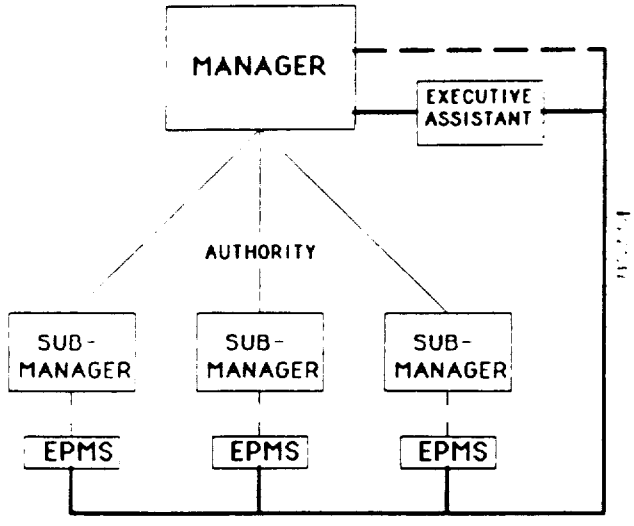
Access and Traffice Control

Security Systems

Safety Systems

Project Closeout
- Retention Reduction and Final Payment
- Resource Reduction
- Closeout Documentation Checklist
- Subcontract Closeout
- Lien Release
- Project Debriefing
  - Historic Cost Acquisition
  - Productivity Analysis
  - Learning from the Project

Figure 1 : EPMS Two-Tier Hierarchy



require data/knowledge transfer in the case of a stand-alone EPMS. In either situation, compatibility with existing resources emerged as an important criterion that places unique flexibility demands on the expert system "shell".

In response to the need for this adaptability a concept for an EPMS generator having a four-ringed architecture was adopted (see Figure 2).

2.1) Ring Four: Site Specific Elements

The outermost ring is representative of a gateway to the manager's external information environment. Most of the managers interviewed indicated a strong dependence on the availability of reference information, historical data and other large data base management and information retrieval requirements. Access to the external environment is accomplished in many different ways including person-to-person communications, on-line retrieval via a computer terminal, customized research conducted by a services firm, or physically locating the information in a library or other repository. Most of the groups indicated that for an EPMS generator to be effective, this vast array of information resources had to be taken into account, either by direct interface (in the case of computer data bases) or at a minimum, by identifying the source, point of contact, and location where supplementary information can be obtained. In essence, the outer ring represents the various "hooks" of the EPMS generator to the outside world, including intelligent information retrieval, organizational knowledge, generation of copies/sessions for use on proliferated stand-alone machines, and numerous other extension utilities.

2.2) Ring Three: The PM Kernel

The next ring represents the next "layer" of project management activity that emerged as a result of the experts' discussions. PM activity was found to have two main modes: 1) planning, where specifications, budgets, milestones, etc. for a new project are formulated, and 2) monitoring, where the execution of the plans developed in (1) are carried out. Most participants indicated that, after they had researched and obtained the necessary (or at least the most available) reference and background material regarding a problem or decision, the next step involved a series of processes where the information was sorted, ordered, analyzed and presented. Performance of this type of activity was the basis for the design of the planning mode of the EPMS generator. This generator consists of a project management subsystem that contains heuristics and analytical techniques used by project managers in analyzing information, assessing problem situations and generating proposed responses. For the monitoring mode, it was necessary to make available a subsystem of customizing utilities, whereby a project manager could specify and create an "automated layer of information filtering" including parameter and alarm thresholds, milestones, quick-look summaries etc. Finally, a user interface subsystem that makes use of human factors and computer visual engineering (CVE) principles was identified as a requirement for both modes. The interface design feature most requested by the experts was the ability to choose from a list of presentational formats, depictions, or other customized user-generated displays.

2.3) Rings Two and One: Analogical Reasoning Applied

The analytical filtering and processing specified for Ring Three is intended to result in the generation and display of key indicators, barometers and other parameters that are important to project management decision making. Most managers agreed that it was at this point in the thought process that analogical reasoning was most frequently applied. This was evident in most manager's comments, which stated that comparisons with past similar experiences were keyed

Figure 2: Rings of the EPMS Generator

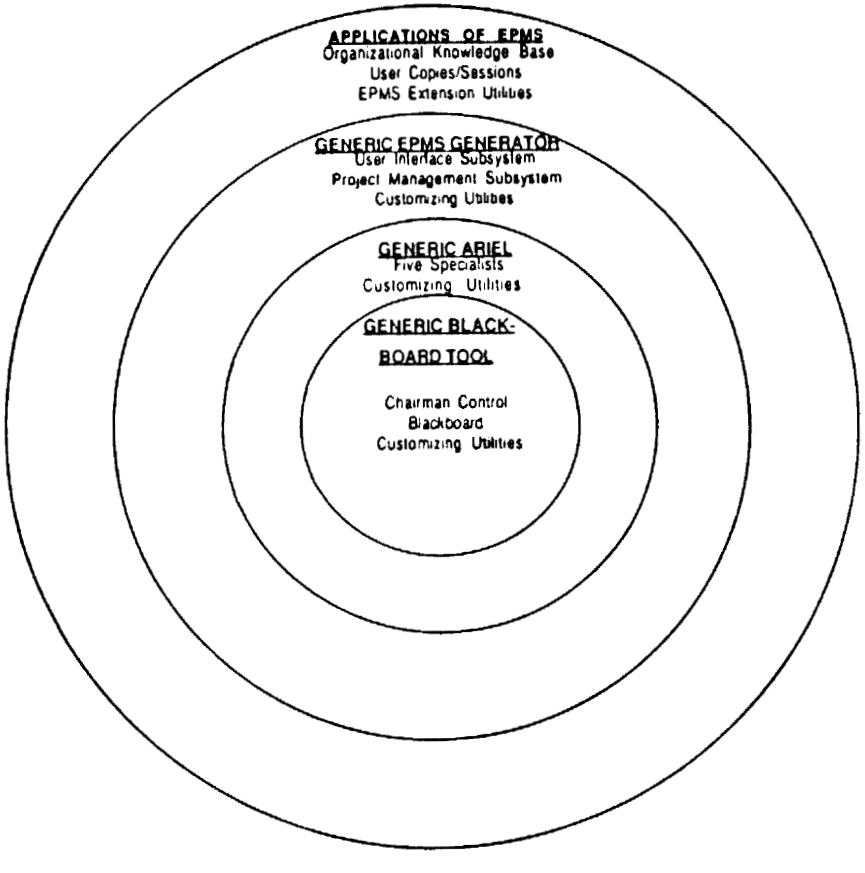
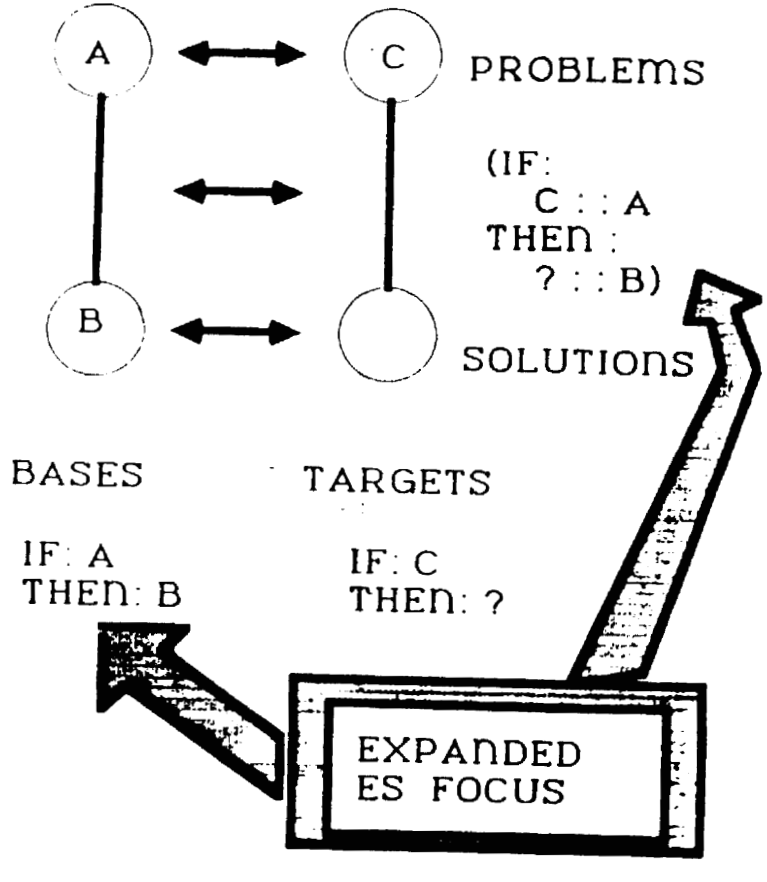


Figure 3: Contribution of ARIEL



on parameters resulting after extensive analysis of the data had been performed (ring three) rather than on the raw data elements themselves (ring four).

The process of analogical reasoning is best understood by referring to the classical analogy test question which is invariably written in the form: "some known problem, A, is to some known solution, B, as a new problem, C, is to which of several possible options (X,Y, or Z)"? Or more tersely, this can be written as A:B::C:(X,Y,Z,)? This is depicted in Figure 3, where A and B are the Base problem-solution pair (or pairs if more than one possible analog exists) and where C is the target problem statement. X,Y, or Z are the unknown target solution shown in Figure 3 as an empty circle.

A traditional stand-alone expert system operates on target problem-solution pairs generally via a series of productions of the form:

IF: C
THEN:?

That is, the traditional expert system isolates the conditions of the target problem space and attempts to directly infer the solution to be one of X,Y,or Z.

In order to make effective use of analogy, particularly in an automated analogical reasoning support environment, studies of the knowledge elicitation sessions showed the need to extend a traditional expert system in two principal ways: (1) facilitate identification of the target problem, C, by looking for similar problem statements in the set of bases, A, and (2) to help flush out the target solution space by suggesting past solutions from B that in part or in whole appear helpful and by assisting in adapting those solutions to the current problem (minimizing tendencies to rely too heavily on the past -- i.e., the "anchoring and adjustment" bias). This capability is illustrated as the "extended expert system focus" shown in Figure 3. Furthermore, this expanded focus must support situations in which the target problem, C, is initially ill-defined, rather than known a priori, as is the case with more conventional analogy programs. The same uncertainty must be manageable in determining analogous problem-solution pairs (A and B)s. For this reason, a major goal of this development effort was to gain the ability to scan a large set of possibilities and to generate and test ideas, with the best ideas being examined for merging, manipulation, transformation, and other disanalogy elimination heuristics.

3.0) EPMS Prototype

Figure 4 provides an overview of the portions of EPMS that were experimented with in the prototype. There are three major parts to Figure 4 -- the longer term aids, the core of EPMS, and the customizing utilities and user support functions. The prototype was an experiment upon the latter two parts which involved building just enough of each part to glean insights useful for next step development. The first part identifies the long term aids that are foreseen in order to integrate EPMS into an existing support structure, which would complete the kernel concept by bridging together the EPMS core with external data bases, tools, algorithms, sources of knowledge, workstations, personal computers, and other hardware.

The customizing utilities identified on the right-hand side of Figure 4 are geared toward making the EPMS kernel attractive to a wide variety of potential users. Hence the utilities support the tailoring of each and every knowledge base, analog, object, and PM subsystem module to the specific application of the individual user. Although the customizing utilities were not developed for the prototype, their design and scope was a major focus of the experiment. High level designs for many

Figure 4 EPMS Elements Exp mented With in the Prototype

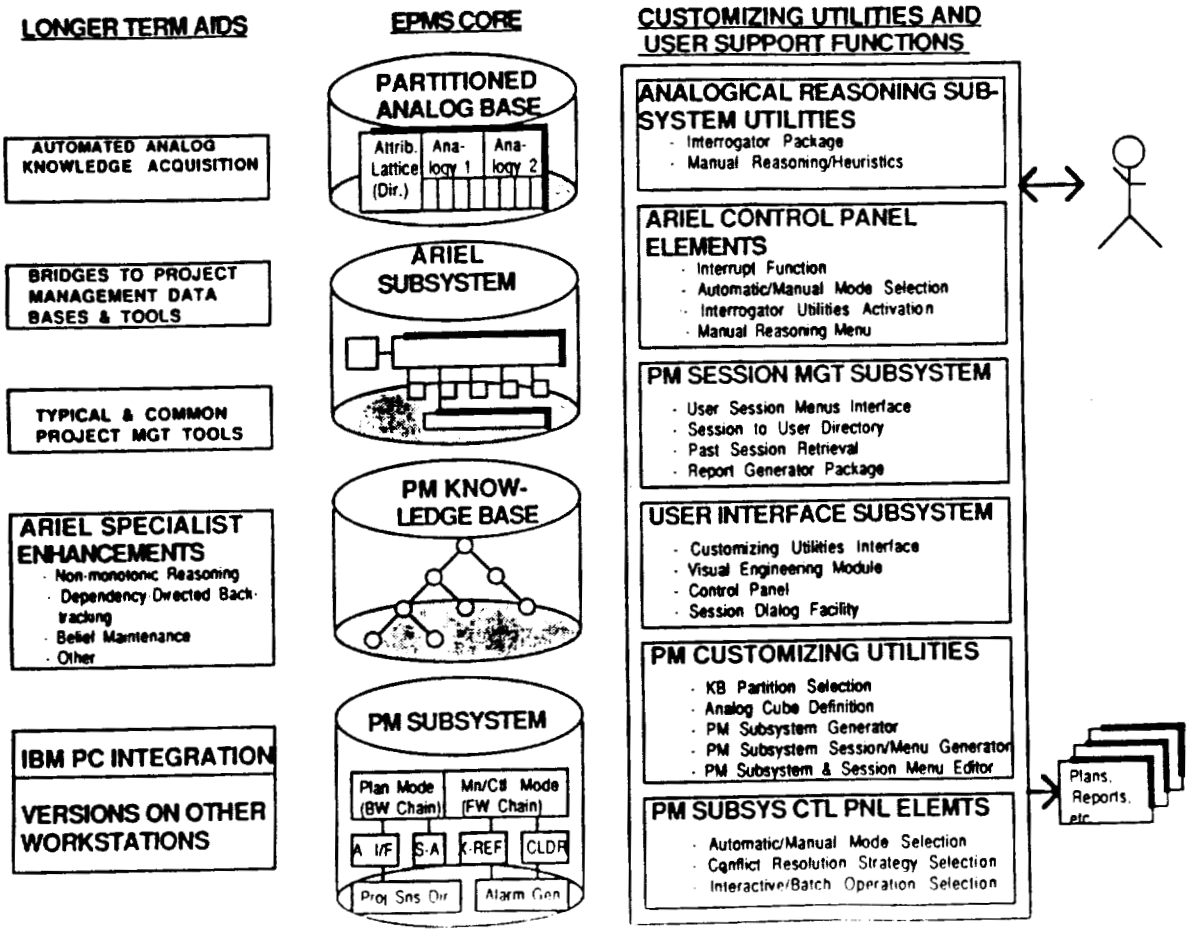


Figure 5 Project Management Subsystem Overview

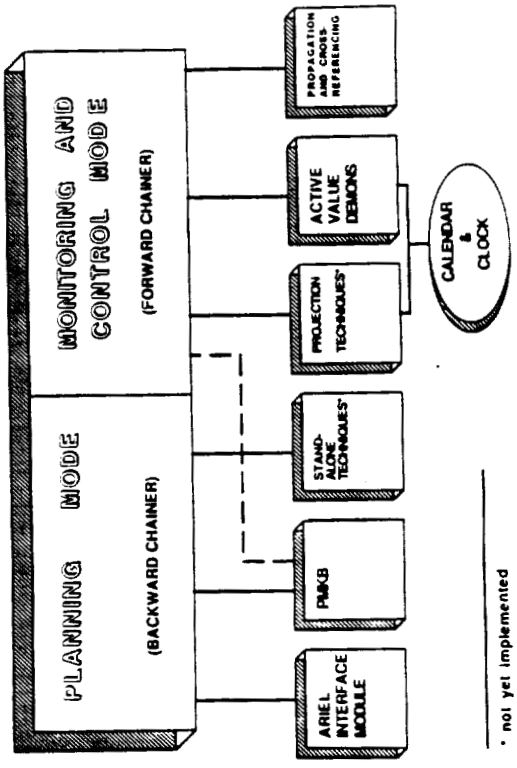


Figure 6 An Intelligent Subsystem Milestone Aspect Object of the PMKB

```
[Subsystem-Milestone-Aspect  Name:

        Definition:
        Comment:
        Objects Depended On:
        Objects Affectable:
        Techniques To Use For Inferring The Plan:
        Plan Value:
        Comment on Value:
        Technique Used:
        Date By Which Plan Value Is Due:
        Techniques For Determining Actual Value:
        Techniques For Projecting Actuals:
        Actual Value (Current Date):
        Actual Value (Projected To Due Date):
        Caution Threshold Level:
        Object Status (one of Completed, On Track,
                       Caution, Emergency):
        Actual Value Comment:                    ]
```

of these utilities were created and user evaluations of and reactions to these designs were elicited.

The remainder of this section will be devoted to providing descriptions of the four main components of the EPMS core, along with a discussion of the three exemplary user sessions that were run during the course of the experiment.

3.1) PM Subsystem

An overview of the PM subsystem is shown in Figure 5, which further illustrates the two-mode concept of operation. In the planning mode, a self-contained PM Knowledge Base (PMKB) works with a stand-alone backward-chaining inference engine to assist the user in planning all aspects of a new project. The inference engine "knows" how to draw on the ARIEL subsystem for assistance if its stand-alone techniques are unable to estimate a value required by the plan or if a search across a wider range of candidate analogs is required. That is, it chains through each subsystem, milestone, and aspect of the new plan for a new project. The aspects elicited include planned levels of manpower, dollars, etc. per time period and work package.

In the monitoring and control mode, a forward chaining inferencing technique is used in conjunction with active value "demons" to constantly monitor and test the deviations of actual values from planned values for the various subsystem-milestone-aspect objects. When cautionary (or emergency) alerts are detected, all related objects within the knowledge base are tagged with an alarm message, which allows the user to determine the source and the nature of any deviation that may affect overall project performance. This cross-referencing feature was cited as a major requirement currently lacking in most project monitoring systems. This mode was also equipped with a clock and calendar, in response to concerns expressed regarding the lack of the ability of current expert systems to adequately account for the effects of the passage of time on any given situation. As a result, all activities in EPMS are time-tagged in a Julian date format, as a means of keeping a record of the time of occurrence and duration of important events. The operation of each subsystem component including the control panel will be illustrated further in the description of the user sessions.

3.2) Project Management Knowledge Base (PMKB)

The EPMS PM Subsystem seeks to establish a new project plan and to monitor its progress. This is done cooperatively and interactively with a human participant, and a completed project plan ultimately becomes one more analog in the Analog Knowledge Base (AKB). The PMKB is thus a set of rules and objects designed to capture and hold the subsystem-milestones-aspect knowledge for the "target". Since this knowledge is unknown initially, the PMKB must hold both the expected or planned value for each subsystem-milestone-aspect and how that value was obtained.

The data structure for such a subsystem-milestone-aspect is shown illustratively in Figure 6. The slots for holding the important pieces of information are shown, however, the methods and other intelligence features are only implied by this Figure. The prototype EPMS implemented and tested most of these features with the exception of the projection capability.

3.3) Partitioned Analog Knowledge Base (AKB)

In order to facilitate search and to improve execution time the analogs are stored in a structure (Figure 7A) that defines two important characteristics: (1) the typology classification scheme, and (2) the progressive deepening levels. The typology classification scheme captures the sorting process that PMs use to classify

projects. For example, the statement "this project is a Class X, Type Y" is often encountered in analogical reasoning (see Silverman 1983, 1984, 1985, 1986). Hence the knowledge base offers a "typology plane" or surface. This classification scheme can be pursued at multiple levels of problem solving depending how deeply involved the problem solver is. Thus a progressive deepening scheme is also offered wherein the EPMS user could enter at the level at which he wishes to work (only three example levels are shown in Figure 7 A, more are possible).

Within each partition of Figure 7 A, are the analogs that correspond to that class-type of problem. Each analog is itself multi-dimensional as portrayed illustratively in Figure 7 B. The three dimensions shown are of variable length and capture the fact that most projects involve multiple subsystems each marching along a set of prescribed milestones. Attached to each milestone are each of the aspects listed vertically in Figure 7 B. The actual knowledge about each subsytem-milestone-aspect is stored in any of a number of possible representations (e.g., map, icon, graph, table, list, rule, etc.). Also stored with each subsytem-milestone-aspect are any relevant advice lessons learned, etc. for selected class-types of problems encountered.The prototype EPMS includes a 270 node object lattice of attributes associated with Figure 7 A and three analogs corresponding to Figure 7 B. The three analogs are the LANDSAT, SPACE TELESCOPE, and NIMBUS-G projects. In this lattice, each node represents a specific activity, system element, or organizational element of a given project and is organized in hierarchies and grouped into specialized project domain areas, so that the further the lattice is traversed, the more detailed the information about a specific project domain becomes. In this way, the lattice can be used for two purposes within the project management domain: (1) as a guide for selecting attributes to characterize new analogs being entered into the AKB, and (2) as a means of entering a description of the target problem/project to be planned/analyzed.

3.4) ARIEL Subsystem

The ARIEL subsystem physically implements the 5-part analogical reasoning process described in Silverman (1985) & Silverman & Moustakis (1986) within five specialists and a blackboard (see Figure 8). Each specialist consists of a short-term memory, local knowledge bases, and an interface to the main blackboard, which is the shared memory used by the other specialists. The local knowledge bases store the knowledge regarding the specialist's particular area of expertise, as well as knowledge related to planning and control. The local knowledge bases are used to formulate an approach by the specialist on the main blackboard. The local control knowledge base controls the flow of information between the specialist and the main blackboard, making sure that only relevant data/information is exchanged.

3.4 .1) The CHAIRMAN

The main blackboard is interfaced with a CHAIRMAN that controls the information flow among the five specialists. The CHAIRMAN has the same basic structure as a specialist. The primary purpose of the CHAIRMAN is to monitor the operation of the five specialists and to maintain an orderly flow of information to and from the blackboard. The CHAIRMAN also handles all inputs and requests from the user. The situation can best be compared to an individual (the user) presenting a problem to another individual (the CHAIRMAN). The CHAIRMAN then convenes a meeting of five specialists each of which is an expert in a particular aspect of the application of analogical reasoning to problem solving in general. These experts are sitting at a conference table (the Blackboard) and the input problem/request from the user is placed on the table by the CHAIRMAN. The CHAIRMAN directs each specialist to look at the information on the table. Each specialist is asked by the CHAIRMAN to prepare and submit a plan to solve or to help solve the problem, based on its own local knowledge about problem solving. The CHAIRMAN then

Figure 8 ARIEL ARCHITECTURE

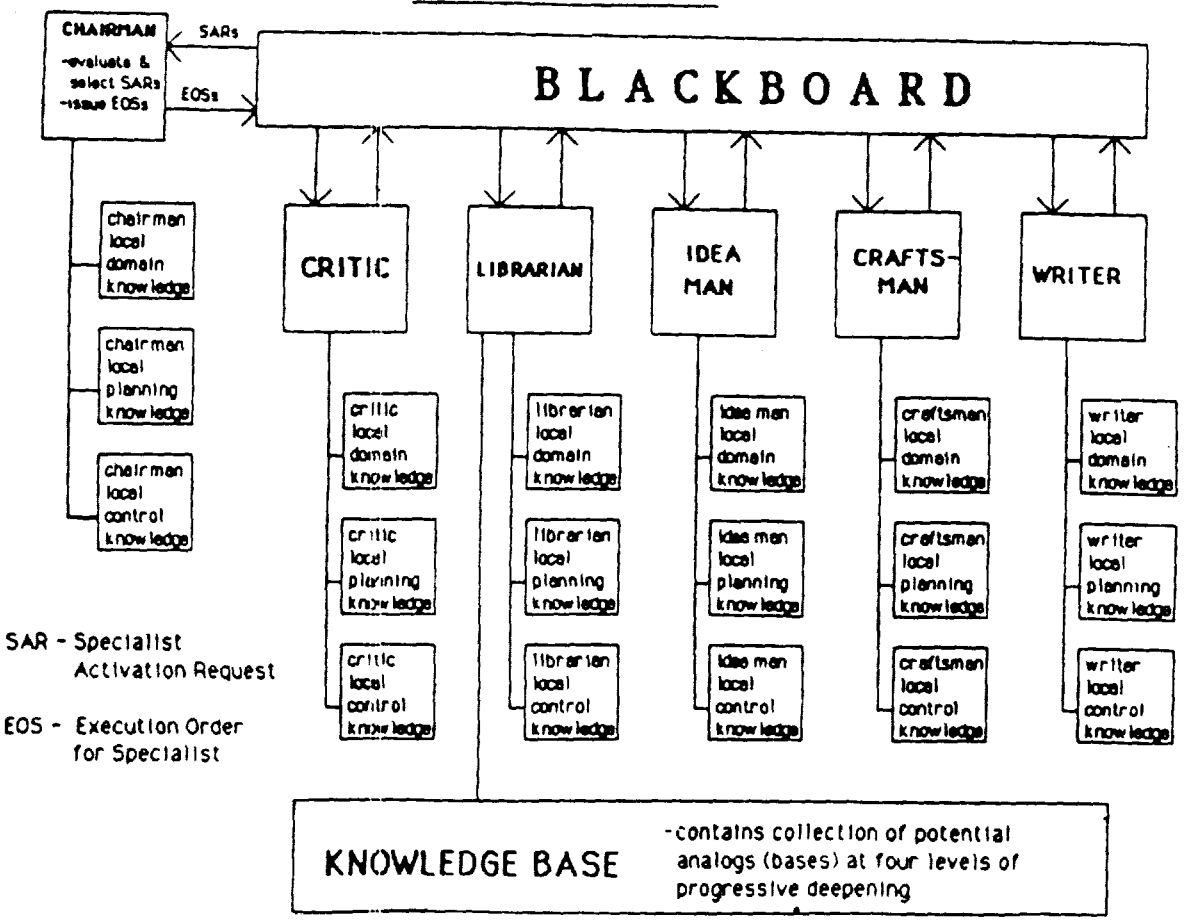


Table 2: Functions Supported a. . Current Status of EPMS Panel Elements

EPMS CONTROL PANEL ELEMENT	PROJECT MANAGEMENT FUNCTIONS SUPPORTED & STATUS							
	KNOWLEDGE ACQUISITION	PLANNING & DEFINITION	PLANNING ASSESSMENT	PERFORMANCE MONITORING	PERFORMANCE ASSESSMENT	COLLATERAL USES	IMPLEMENTED IN PROTOTYPE	APPEARS INACTIVE ON PROTOTYPE SCREEN
ARIEL ACTIVE IMAGES								
ARIEL INTERROGATOR								
CALENDAR								
CLOCK								
V E SCREENS								
TEXTUAL ADVICE								
TOP-LEVEL TYPESCRIPT WINDOW								
ARIEL QUERY/RESPONSE								
USER QUERY/RESPONSE								
ARIEL UTILITIES								
MODE SELECT SWITCH								

CURRENTLY IMPLEMENTED VIA PROMPT WINDOW

CURRENTLY IMPLEMENTED IN TYPESCRIPT WINDOW & VISUAL ENGINEERING MODULE

evaluates each plan and decides which plans are to be activated. The activation of one or more plans will result in new information being presented to the blackboard, which could conceivably affect the planning processes and results of the other specialists. The CHAIRMAN's main purpose, therefore, is to decide which specialists should be permitted to proceed and in what sequence in order to maintain an optimal and orderly progression toward the goal (target solution). For the purposes of EPMS, a plan submitted by a specialist to the CHAIRMAN is called a SAR (Specialist Activation Request), and the order issued by the CHAIRMAN to a specialist to proceed with that plan is called an Execution Order for Specialist (EOS). The SAR is similar (analogous) in nature to the KSAR employed in HEARSAY-II. (Erman, et.al 1980).

3.4.2) The Five Specialists

Each specialist is a self-contained expert system that opportunistically examines the contents of the blackboard and proposes an analogical reasoning related process or function (e.g., diagnose, classify, evaluate, scan, assimilate, etc.) to the CHAIRMAN as a means of contributing to the progression of the problem toward a final solution. The five specialists support the problem identification (PI), Knowledge Acquisition (KA), Analog Transfer (AT), Knowledge Transformation (KT), and Introduction into Use (I) steps outlined in earlier articles, and are called the CRITIC, LIBRARIAN, IDEAMAN, CRAFTSMAN and WRITER, respectively.

The main function of the CRITIC is to aid in the process of problem identification, problem formulation and requirements definition. To this end the CRITIC monitors the contents of the target problem definition space and determines what methods are to be employed in order to expand or refine the target problem definition. These methods usually entail the selection of an appropriate problem definition aid being presented to the user (via the Writer). The CRITIC is also charged with the overall responsibility of monitoring the target solution generation process as a whole. These tasks range from seeking additional information from the user or LIBRARIAN to invoking a "stopping rule" when either an optimal solution has been achieved or when successive iterations would produce little or no change in the entropy of the target solution. The purpose of the LIBRARIAN is to ensure that all possible building blocks within a certain threshold that could be used in constructing a target solution to the problem are being considered. In order to accomplish this, the LIBRARIAN conducts a search of the AKB by taking each attribute contained in the target problem definition space, searching for each new occurrence of that attribute, and returning to the blackboard all previously unconsidered bases exhibiting that particular attribute. Another major task of the LIBRARIAN is to ensure that the AKB is properly updated with new information generated either by the user or by the ARIEL system itself. Currently the LIBRARIAN is configured only to assimilate final results as a new base (analog) to be considered for subsequent problem solving sessions. In later versions of ARIEL it is planned to also incorporate intermediate results, including erroneous paths, etc., in order to increase the overall intelligence of the system and to promote maximum reuse of lessons learned during the problem solving session.

The primary responsibility of the IDEA MAN is to evaluate each candidate analog based on the value of the similarity metric [Silverman 1986] for that particular analog and the corresponding attributes contained in the target problem definition. Weighing factors to be used in calculating the similarity rating are provided by the user at the request of the IDEA MAN via the WRITER. The candidate analogs are ranked starting with the analog having the highest similarity rating, along with the value of the rating. This output represents a prioritized and valuated space of potential solutions for use by the CRAFTSMAN in generating a composite target solution.

The CRAFTSMAN has as its goal a means ends analysis that leads to the construction of an optimal solution to the target problem using to the greatest extent possible the existing analogs contained in the knowledge base and provided by the user. At this point in the process, all relevant analogs that have been identified have been evaluated and ranked. In constructing the target solution, the CRAFTSMAN starts with the highest ranked analog and checks for a similarly value of 1.0, in which case that analog becomes the final solution and the stopping rule is invoked by the CRITIC. If the similarity rating is less than 1.0, the CRAFTSMAN takes the analog with the next highest rating, and constructs a temporary target solution by combining it with the highest-ranked analog. At this point a new similarity rating is calculated and compared with the rating of the highest-ranked analog. If the new rating is lower the second-highest candidate is dropped from consideration and the third highest candidate is considered in a similar fashion. If the new rating is higher the temporary target solution now becomes the new basis for comparison and the process continues.

The WRITER is the only element through which ARIEL can send messages to the user. For this reason, most of the functions that are assigned to the WRITER involve the generation of prompts aimed at soliciting user input. The core structures of these prompts reside in the ARIEL knowledge base and are accessible via the LIBRARIAN. Once accessed, the WRITER fills in any additional information germane to the problem being worked, and outputs the resulting prompt directly to the user via the screen. These prompts contain hooks to specific data structures (lists) on the blackboard, and these lists are automatically updated as the user responds to the prompt. The other main function of the WRITER is to document ARIEL results and problem solving activity in a manner acceptable for inclusion by the LIBRARIAN into the AKB.

3.4.3) The Blackboard Problem-Solving Framework

The blackboard configuration provides an opportunistic search framework that is used to support an orderly progression from initial problem definition to final target solution formulation (see Figure 9). ARIEL differs from most conventional blackboards in that the user, at his option, can override the actions of any specialist at any stage in the problem solving process. This was considered necessary in order for ARIEL to truly function as an "extension" of the human analogical reasoning process. The arrows in Figure 9, which indicate lateral, forward or backward level transitions, show the user as being totally unconstrained. The CRITIC has the next greatest influence on effecting changes in the direction of the problem -to-solution progression. Note also that this process is iterative, and can be influenced by the activity of the other specialists.

3.5) User Session

The current prototype has the capability to run three exemplary user sessions that were developed with the intention of soliciting feedback from potential users and providing additional insights into the design of the overall system. In particular, the sessions were intended to address some of the EPMS Research Questions, especially with regard to the use of analogy, and the determination of what analytical techniques should be directly incorporated into EPMS. The three prototype sessions and the objectives of each are delineated as follows:

1) Session 1, to demonstrate the project requirements definition function and the use of the analogical reasoning extension.

2) Session 2, to demonstrate the budget planning function as supported by analytical techniques.

3) Session 3, to demonstrate the project monitoring function as supported by analytical techniques, including the cross referencing capability of the EPMS knowledge base.

A step by step summary of each of the sessions is provided in Table 2.

3.5.1) EPMS Control Panel

The purpose of the EPMS Control Panel is to provide a total multi-screen environment in a combination desk-top/pilots' console configuration from which the user can access any and all functions during an EPMS session. The control panel must also facilitate smooth-flowing user-machine dialogs. Use of the mouse/cursor is favored over the keyboard whenever feasible. Although the control panel is ultimately intended to be "tamper-proof", locking out all unauthorized access to the EPMS executive or resident software, the prototype version allows this access because of the developmental nature of the system.

The EPMS control panel is displayed in Figure 10. Some of the features were fully implemented in the prototype, others appear on the screen but are currently inactive (specifically the calendar, CVE screens and manual mode).

4.0) EPMS Prototype Evaluation

The purpose of this section is to document the results of the EPMS prototype evaluation sessions. Following a sequence of software IV & V (Initial Validation and Verification) tests, a series of expert evaluation sessions were conducted in which potential users were given the opportunity to run exemplary sessions and to provide reactions/comments on the overall system design and user interface. The comments were then used to create a composite summary of desired enhancements/improvements to be incorporated in EPMS Version 1.0, and to generate a list of 11 research questions mandating continued further exploration.

4.1) Expert Evaluation

This section describes the results of six separate knowledge collection sessions in which domain experts were given demonstrations and/or functional descriptions of EPMS: i.e., its goals, its conceptual design, and the types of sessions a user would encounter. The domain experts, in turn, each offered several types of feedback that are documented here including:

1) Evaluation of EPMS in terms of its goals, design and sessions. This feedback included suggestions for altering and improving EPMS.

2) "Deep Knowledge" was offered for EPM's knowledge banks. That is numerous ways for EPMS to utilize and/or "plug in" to existing handbooks, data bases, and other procedural aids was offered.

3) Heuristic Knowledge and rules of thumb used in project management were elaborated that could and should be incorporated into the EPMS knowledge base.

These three types of feedback--evaluation, deep knowledge, and heuristic knowledge are popularly thought to be collected in distinct sessions: a knowledge elicitation session with domain experts and an expert system evaluation session by potential users. While textbook descriptions of knowledge engineering invariably separate evaluation from elicitation, the fact is that both forms often are intermingled in any one interview session, particularly so during the conceptual

design period, as was the case here. A summary of the types of sessions and the experts participating in each is provided in Table 3.

4.1.1) Overview of 17 Experts' Comments

Table 4 provides a comprehensive summary of the major comments/suggestions received during the 6 sessions, and indicates whether the suggestion influenced the design of the prototype, the user sessions (interface), the planned version 1 system, or long-term enhancements. Not surprisingly, the three unanimous reactions favored: 1) the need for an environment to support and extend the human expert analogical reasoning process; 2) the need to structure domain-related knowledge in a 3-dimensional format that supports traversal across various functional characteristics and down various levels of granularity (progressive deepening); 3) the need to capture "lessons learned" and incorporate this knowledge in a manageable automated "Corporate Memory" structure, that could be called on to produce advice when a set of similar conditions are detected in real-time. It should also be noted that with the exception of the executive assistant concept arrived at in session #2, and the need for a separate AKB and PMKB from session #3, no other comments were unique to only one session. In fact, each session generated an average of about seven comments that were either actually incorporated in the prototype, or were entered as planned enhancements for either Version I or other long-term developments

4.2) Insights for an EPMS Generator

At the beginning of the EPMS prototype effort the investigators formulated 11 questions to be researched as stated in technical objectives of this report. The purpose of this section is to delineate the 11 research questions (see Table 5) and to discuss the answers arrived at during the course of the study. The first three answers indicate that there is a strong-felt need for analogical support; maintaining that support should not require much time or effort of the project team members, a dedicated assistant should be responsible for interfacing to EPMS (it may not be his only responsibility). The next two answers repeat the fact that there are an inumerable number of PM subsystems possible, most of which should be relegated to the longer term development period or to user development activity.

In terms of the Physical Model, answers to questions 6 through 8 were explored in describing the design of the Control Panel and sample user sessions and are summarized in Table 5, primarily in terms of utilities and packages needed to effect the desired results. These are not final answers but rather may be viewed as good starting points for future refinement. The answer to question 9 extends the scope of the utilities needed for effective user interrelation with EPMS. Finally the answer to question 10 points toward utility packages that help EPMS achieve flexibility and generality.

The very last question deals with what machine, environment and language to develop EPMS in. This is the same question numerous software vendors have yet to find the optimal answer to. The only solution seems to be to develop the product generically as possible on one machine and then gradually port it to other machines as time and funds permit.

5.0) Next Steps

Given the work done to date, the lessons learned, (partial) answers to the research questions, and the user interests and attitudes, a number of next steps are immediately obvious. These include:

1) Proceed With ARIEL Subsystem -- The ARIEL Subsystem should be flushed out as soon as possible and as originally designed and conceived. No user reactions indicated any concerns about ARIEL's design or heuristics. On the contrary, users

Table 4: Experts' Comments Summary

SESSION #						SUGGESTION	IMPACT					
							DESIGN			SESSION		
1	2	3	4	5	6		Prototype	Version 1.0	Long-Term	Prototype	Version 1.0	Long-Term
X	X	X	X	X	X	Analogical Reasoning Extension to Support Stand-Alone EPMS		X				
X	X		X			Integration of EPMS to existing DBMS/MIS/DSS Resources			X			
X	X					2-Tier Management Support Structure	X					
	X					Executive Assistant Concept (Recall Figure 3B)		X				
	X	X		X		KB Generation Aid			X			
	X					AKB Separate from PMKB	X			X		
	X		X	X		Separate Planning and Monitoring Modes	X			X		
X	X	X	X	X	X	Progressive Deepening and Typology Strategy	X			X		
		X		X	X	Alarm Condition Propagation and Cross-Referencing	X			X		
X	X	X	X	X	X	Management of "Lessons Learned" Knowledge to Support Advice Generation			X			X
		X		X		Schedule/Calendar Module		X			X	
		X		X	X	Projection Algorithms			X			X
				X	X	Computer Visual Engineering	X			X		

Table 3. Expert Knowledge Collection/System Evaluation Sessions Summary

SESSION #	EXPERT #	RANK/TITLE	ORGANIZATION	SESSION TYPE			
				KNOWLEDGE ELICITATION		SYSTEM & USER I/F EVALUATION	
				CONCEPT WRKSHOP	DESIGN BRIEFING	SESSION BRIEFING	SESSION DEMO
1	1	Division Manager	Goddard Space Flight Center (GSFC)		X	X	
2	2,3	Technical Manager	Jet Propulsion Laboratory (JPL)		X	X	
3	4-6	Project Manager	(JPL)	X			
4	7	Project Manager	Department of Energy (DOE)	X			
5	8-12	Deputy Under Secretary's Staff	Department of Defense (DUSDT&E)		X		X
6	13	Lieutenant Colonel, US Army	Chief of Staff, US Army		X		X
	14	Procurement Officer	US Army		X		X
	15	Professor	University of Oregon		X		X
	16	Major, US Air Force	Defense Systems Management College		X		X
	17	Captain, US Navy	Defense Systems Management College		X		X

Table 5: EPMS Research Questions

	Answers
Logical Model	
(1) Q. Is analogy truly important to project managers and if so what is its most important purpose? Is there a hierarchy of purposes and uses?	A. The progressive deepening concept was uniformly accepted as the correct way to organize each analog and hence, this hierarchy should include executive, management, and technical levels.
(2) Q. Given the rapid pace of most projects and the already heavy staff workloads, how should analogy information be collected for use in EPMS?	A. Some form of customizing utility for defining the structure/format of the user's analogs should be prepared.
(3) Q. Given that a project is a team of people should there be a single principal user of EPMS or should EPMS simultaneously support a number of members of the team?	A. EPMS should be oriented around two tiers of management (manager and sub-managers) with a dedicated executive assistant as the prime interface agent.
(4) Q. How exactly will these different users utilize EPMS? The analogical information? What types of sessions will they want to have?	A. The details of EPMS use will depend on the characteristics of the particular PM subsystem being installed, however it will most likely include both planning and monitoring modes.
(5) Q. Analogical reasoning is only one technique of project personnel. Which closely related techniques should be directly incorporated into EPMS vs. which should be relegated to user integration efforts?	A. There are an inumerable set of related techniques utilized in each stage of a project's life. It appears that all of these should be relegated to user development and integration, at least in the short term.
Physical Model	
(6) Q. A great many heuristics (i.e., computerized functions) appear necessary so that non-programmers can effectively retrieve and manipulate analogy and project data. Which of these are most important? In what order should they be developed?	A. In each of the demonstrations given, multi-windowed, visual feedback seemed popular and easy to comprehend, however, the exact details of what should be contained in each window/menu block require much greater user testing.
(7) Q. Man machine interfaces can make a large difference in the success of a software package. How should the user interface be designed? What parts of the EPMS should the user see on the screen and how should user access be effected?	A. A number of knowledge representation forms should be utilized including rules, nets, lattices, frames, tables, lists, and images. Tables, lists and images appeared most popular for output displays. Pop-up menus were the desired mode of user input, except a "lattice climber" was preferred for more complex domains.
(8) Q. Along the same lines, how should the user interface be designed from a knowledge representation and display perspective?	A. The user interface must support the generation of relatively numerous sets of simple typology categorizations.
(9) Q. What is needed to assist nonprogrammers in customizing EPMS to their site? What can be expected to be generic vs. what is site specific in the knowledge bases and ARIEL subsystems? What customizing/extending utilities are needed and in what order should they be developed?	A. The experts' feedback seems to indicate that every aspect of EPMS should be designed with a customizing aid -- from the screen elements, to the analog structure, to the PM subsystem interfaces.
(10) Q. What degree of effort needs to be expended to develop the project management subsystem? Will users generally have their own stand alone aids that they want to connect EPMS to or will they want EPMS to provide this capability? What is the essential kernel of this subsystem that many types of potential users will find most effective? What customizing utilities will be needed to	A. In the short term it may be more fruitful to concentrate all effort and energies on ARIEL, the utilities, and the interface subsystems.
Actual Computer Implementation	
(11) Q. Ideally, the EPMS generator should be readily available to users who own a variety of vendor workstations and computers. Given that the prototype is being developed on a Xerox Lisp machine is it reasonable to make maximum use of COMMON LISP and of IBM PC interfacing software or is some other language and environment (e.g., C on the SUN) more nearly optimal?	A. There is no easy answer to this question as numerous software vendors have discovered. For the PM world one would expect that purchasing a stand-alone workstation to run EPMS should not be a large drain on their budgets, especially with the price of workstations falling so rapidly. However, if EPMS is to be widely used, the requirement to buy a LISP machine seems prohibitive, particularly in view of the lesson learned that many users will wish to integrate EPMS with some custom designed stand-alone PM subsystem. The danger of locking in to the LISP machine world is that the overwhelming majority of PM offices have IBM PC's or any of a host of other workstations or minicomputers but few of them have LISP machines. In the short run this obstacle probably can't be overcome. In the mid term the solution might be to develop "bridging" software that creates virtual interfaces to a variety of other machines (e.g., IBM PC world) on which many stand alone PM subsystems are likely to be built. A longer term solution is to port EPMS to a number of different machines that support common LISP and/or to port it to a second language such as C which is fairly available.

want lots of simple heuristics, progressive deepening, typology and level selection, etc.
2) Flush Out Manual and Interrogator Mode Utilities -- These utilities have been defined and should now be built to permit users to both inspect all aspects of ARIEL reasoning chains and to permit advanced users to manually reason by analogy on the KB elements.
3) Develop Customizing Utilities -- Requirements for generality and flexibility can be satisfied with the development of numerous, relatively small utilities. Each utility can perform a single adaptation function (e.g., support individual analog feature generation tasks) that permits EPMS to be molded to the user's specific PM subsystem needs.
4) Select Field Test Site(s) -- Until users begin to actually try and apply EPMS to their site and to use it on a regular basis, there will be no way to accurately evaluate its man machine interface. For that purpose, one or more test sites should be selected as soon as possible and EPMS should be installed and adapted to their problem(s). It is most desirable to select a test site that either already has an existing stand alone PM subsystem or that does not want a very strong PM subsystem capability. These sites would provide useful MMI insights with a minimum of tangential PM subsystem development activity. A longer term goal will be to select sites with needs for greater PM subsystem help so as to focus in more clearly on what the customizing utilities and kernel elements should entail.

Acknowledgement

This work was funded in part by NAS7-949,GWU/EAD, and Intellitek, Inc.

Bibliography

1. Rounds, J.L. (1985), "Expert Systems Potential As a Cost Engineering Tool", presented at the 30th Annual Meeting of the American Association of Cost Engineers, Chicago (Preprint 86200).

2. Silverman, B.G., (1983)"Analogy in system management: A theoretical inquiry", IEEE Trans. Syst., Man. Cybern , vol. SMC-13 no. 6, pp1049-1075. (Dec).

3. _____(1984 a), "Toward An Integrated Cognitive Model of the Inventor/Engineer" R&D Management vol. 15, no. 2, pp. 151-158.

4. _____, (1984b) "INNOVATOR; An Expert System for Management of Modeling and Simulation", Proceedings of the International Test and Evaluation Conference, Nov. '84, pp._____

5. _____, (1985a) , "The Use of Analogs in the Innovation Process: A Software Engineering Protocol Analysis", IEEE Trans. Syst., Man. Cybern., vol. SMC-15, No. 1 Jan/Feb., pp. 30-44.

6. _____,(1985b), "Potential Software Management cost and Productivity Improvements,Computer, IEEE Computer Society, v. 18, n. 5, May 1985, pp.86-96.

7. Silverman, B.G., & Moustakis, V.S. (1986a) "INNOVATOR: Representations and Heuristics", in B.G. Silverman, Expert Systems for Business, Reading: Addison-Wesley (forthcoming)
8. Silverman, B.G., Murray, A., Feggos, K., (1986b) "Analogical Reasoning Integration and Extension Language (ARIEL)," The George Washington University Institute for Artificial Intelligence, Washington, D.C.*

9 Erman, Lee D. et.al, "The Hearsay - II Speech-Understanding System: Integrating Knowledge To Resolve Uncertainty, Computing Surveys, Vol. 12, No.2, June 1980

* Available from the authors

A Natural Language Query System for Hubble Space Telescope Proposal Selection

Thomas Hornick [1]

William Cohen[1]

and

Glenn Miller[1]

Astronomy Programs, Computer Sciences Corporation
Space Telescope Science Institute [2]
3700 San Martin Drive
Baltimore, MD 21218

Abstract

The proposal selection process for the Hubble Space Telescope is assisted by a robust and easy to use query program (TACOS). The system parses an "English subset" language sentence regardless of the order of the keyword phrases, allowing the user a greater flexibility than a standard command query language. Capabilities for macro and procedure definition are also integrated. The system was designed for flexibility in both use and maintenance. In addition, TACOS can be applied to any knowledge domain that can be expressed in terms of a single relation. The system was implemented mostly in Common LISP. The TACOS design is described in detail in this paper, with particular attention given to the implementation methods of sentence processing.

[1]Staff Member of the Space Telescope Science Institute

[2]Operated by the Association of Universities for Research in Astronomy for the National Aeronautics and Space Administration

1 Introduction

The Hubble Space Telescope (HST) will be launched aboard a space shuttle into low Earth orbit where it will function as a remotely controlled observatory for the next 15 years. Astronomers wishing to use HST submit observing proposals to the Space Telescope Science Institute (STScI). Due to the unprecedented capabilities of HST, a high oversubscription rate is anticipated; 1000-2000 proposals will be submitted each year, while only about 200-300 can be granted observing time. The evaluation of proposals will be accomplished via a peer-review process: An external scientific review committee (the Time Allocation Committee or TAC) advises the Director of the STScI in the selection of the HST observing program [1]. Although the scientific merit of a proposal is of prime importance, the selection process must take into account various limited resources such as unocculted viewing time, power, and communications links.

Due to the complexity and size of the proposal selection process, it was clear that software support was essential to assist in tracking proposal evaluation criteria. The Time Allocation Committee Operations Support (TACOS) system was developed to meet the following requirements:

- Flexibility: Although the basic procedures of the selection process are fixed, several detailed aspects are either uncertain or subject to change. Additionally, it was recognized that the first round of proposal selection would undoubtedly lead to several adjustment to the selection procedures. Therefore a prime consideration in the design of TACOS was to create a very flexible system in terms of the input data, query language and presentation of the output.
- Natural language interface: As the TAC members can devote little time for training in the use of the system, an easy to learn interface that allowed queries to be expressed in an English-like form was desirable. Tolerance to input errors and on-line help was also important. Available database query languages had complex and rigid syntax which require a significant amount of formal user training.

Several features of TACOS are worthy of note:

- Use of a bottom-up, shift reduce parser, rather than the more popular Augmented Transition Network (ATN). In the adopted approach, the grammar is maintained as data to the parser, whereas the grammar is embodied in the code of an ATN.
- Flexibility was not only incorporated into reporting, but also into the maintenance of the the system itself. Easy access to the initialization files allow the following to be modified if needed:
  - The TACOS database values and fields may be modified slightly or replaced completely, as long as the database has been generated in the proper input format. In fact, the database can be replaced by a new data and fields related to a completely different subject, which allows TACOS to be utilized in an unlimited number of domains.

- Changes to any of the field names or the security level of a field made be made within one initialization file.
- Keywords may be added, deleted, or renamed by making the proper changes to the lexicon initialization file.
- Grammar rules may be added, deleted, or modified by making the proper changes to the grammar initialization file.
- All keywords and phrases may be customized via macro definitions to accomodate the user's needs by either a initialization file or interactively by using the "define" phrase. The customization feature includes defining procedures, in which a series of commands may be executed with one predefined macro.

- The system was implemented in VAXLISP, a version of the popular Common LISP. Source code of the system may easily be ported to other hardware with little modification.

The first section of this paper provides an overview to the TACOS system. Sections 3-6 contain the details on the system design, featuring an in-depth look at the major components. The last section gives the conclusions of the paper.

2 Overview

The TACOS database consists of the all proposals being considered for selection. For each proposal, the database contains information relevant to selection such as Proposal ID, Principal Investigator information, total exposure time, dark time, etc. Each field can have three possible values:

- **original** - value input to TACOS; the original value is maintained to allow comparison to the limited value.
- **corrected** - In the event that an original value is in error, a TACOS user can fill the corrected slot.
- **limited** - The selection process may allow a proposal more or less of some quantity than was requested. This is kept in the limited slot.

The TACOS system will run interactively during the selection meetings (several sessions should be meeting simultaneously). Committee members have copies of each proposal and summary reports on criteria essential for selection, e.g. viewing time, number of targets, etc. During the deliberation process, the committee produces a ranked list of proposals and may also set limitations on resources used by a proposal. The TACOS system keeps track of these resources and produces a variety of repors from this information. A few examples will illustrate several features of TACOS.

```
define procedure "adjust limits" as
```

```
    define "scale" as
        "limited primary time / original primary time"

    define "resources" as
        "parallel time to scale*parallel time,
         parallel time used to scale*parallel time used,
         parallel time made to scale*parallel time made,
         time on target to scale*time on target,
         data volume to scale*data volume,
         number of uplinks to scale*number of uplinks,
         scheduling difficulty to scale*scheduling difficulty"

    limit resources

end procedure
```

Simply inputting the phrase "adjust limits" implements the procedure and sets the new limits.

To provide immediate feedback to the members of the meetings, requests can be made to display or print relevant data on the proposals. For example:

```
Tacos> display resources with primary time greater than 20
con't>
```

```
Tacos> output resources with primary time greater than 20 using "printer"
con't>
```

The ability to sort all proposals on field values is especially usefull when reviewers require a limit to be placed on a particular resource.

```
Tacos> display resources by largest primary time for all proposals
con't>
```

Due to the limited availability of HST resources, the committee may recommend the allocation of resources by a panel grade. If resources are allocated on this basis, committee would need to view all proposals in order of highest to lowest recommendation. They would also require a cumulative total of resource allocation to be shown according to the rank order, to allow them to determine when all resources have been allocated.

```
Tacos> display cumulative primary time by panel rank
con't>
```

TACOS is written mostly in VAX LISP, with some support routines written in VAX/VMS Digital Command Language (DCL), and editing and display capabilities provided using VAX/VMS Text Processing Utility (TPU).

3 Design

This section describes the high level design of the TACOS system. Figure 1 shows the architecture of the system.

The underlying structure of the TACOS database is rather complex, featuring a **objset**, a set of objects. Each **objset** contains a sorted list of the elements it contains and a hash table mapping the name of an element to the element itself. Each **element** is a hash table mapping field names to **fields**. These **fields** are actually property lists that maintain three values: original, corrected and limited. A special sort of the **objset** is maintained and is known as the **master set**.

Each sentence is processed in the following way. First, **read-input** reads the sentence from the terminal (or some other source.) **Read-input** reads until it reaches a blank line; it then concatenates the lines it has read into a string and hands this to **scan**. **Scan** then converts this string into a list of tokens: each token at this point is either a LISP atom, a number, a string, or a LISP list composed of numbers and/or strings. This *token list* is then checked for macros by **expand**. If any macros are present, then they are expanded in line. Finally, each token in the token list is classified by the **classify** module. **Classify** replaces each token with a simple parse tree.

Next, the expanded and classified token list is passed to the parser, where the list of trees is parsed into a parse tree. This tree is then in turn converted into a function which uses a restricted subset of LISP by the **codegen** module.

This function is then passed to the **backend** module. The first thing the **backend** does is to apply the function, via the **codeeval** module, to the **context** data structure, changing one or more fields of the **context** structure. The **context** structure is used for communication between frontend of TACOS and the backend. After various fields in the context structure have been set, the **backend** module examines the structure and uses the information in it to do what the user requested with his original command.

The **context** structure contains many fields; below is a list of the more important ones:

- The *verb indicator* which indicates what verb was used.
- The *consider set*, a set of proposals.
- The *change set*, a set of *change values*. A *change value* describes how some field of a proposal should be changed, as determined by the proposal selection committee.
- The *display set*, a set of values to display. Display values are structures that describe a value to display; they have these fields:
  - a *tag* to be used as a column heading;
  - a column *width*;
  - a *display function* which indicates what value to display, (This function is a LISP data structure which is actually a function of a proposal);
  - a *statistical function*, which indicates how to display the value - (e.g., "total" or "average").

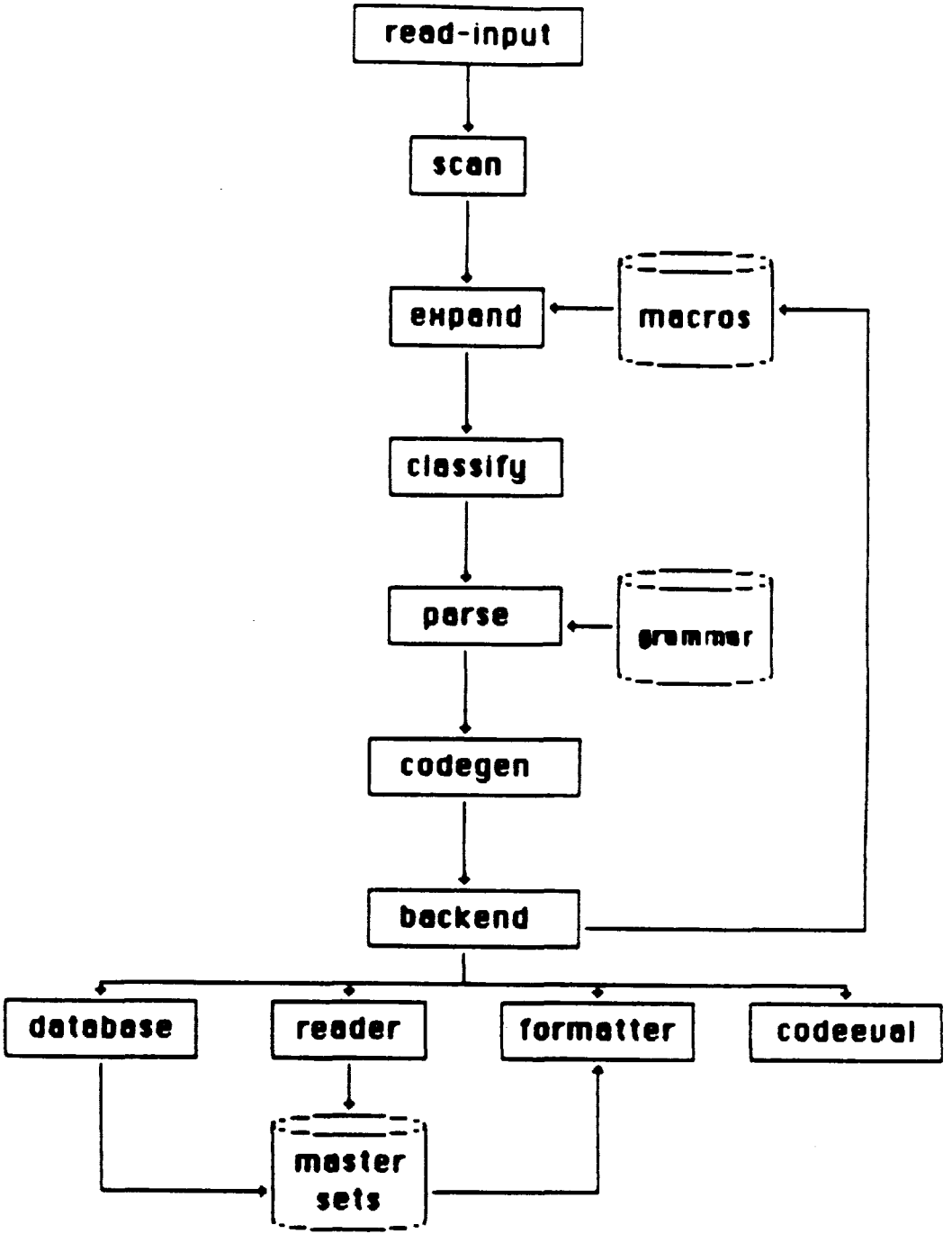


Figure 1: TACOS System

- The *order set*, which indicates how to sort the list of proposals. Each order value is a structure with these fields:
  - an *order function* which is just like the display function above, but is used differently; and
  - an *indicator* which indicates ascending or descending order, with default as descending.
- The *define-set*, which always contains a single pair of strings or is empty.

Other slots in the **context** structure are used internally, or for communicating specialized pieces of information to the **backend** module: for instance, there is a special slot for telling the **backend** module what procedure to call when the verb is **call-procedure**.

Most slots in the **context** structure are static; if **codeeval**ing a sentence doesn't change a slot, then the old value remains. TACOS will use these slot values as default values in the event that any of the set fields previously described are not specified in a single input sentence.

After the **readinput-...-codeeval** cycle is complete, most of the hard work is over. The set of proposals to operate on and other sets (such as sets of fields to display) have all been determined at this point. **Backend** now executes the sentence; this is nothing more than running the verb through a big **case** statement (actually a **cond** clause) and executing the appropriate procedure. Most of these procedures are, if not trivial, at least straightforward. **Backend** recognizes the following verbs:

- display - create a tabular display.
- edit - write the values to be modified into a table, invoke an editor so they can be changed, and then read the table back in.
- change, limit, correct - give a new set of values to some fields. The new values should be specified in the sentence, *e.g.*, "change requested primary time to 3 hours". **Change** changes the original value of a field; **limit** changes the limited value; and **correct** changes the corrected value of a field.
- define - define a macro.
- create partition - write a table representing a set of proposals (more generally, a set of objects) into a file.
- load partition - reads in a set of proposals from a given file. Appropriate error messages will be displayed if the set is disjointed from the current set of proposals, to prevent the database structure from being corrupted.
- define procedure, end procedure - these commands delimit a procedure definition.
- call procedure - call a predefined procedure
- help - call a help utility.
- hint - give a hint about the last error message.

- output, output using format - these are used to output scalar values (like "total dark-time"). The "using format" qualifier is meant to be used only in predefined procedures and is for formatted output.
- exit - quit the system.

4 Sentence Understanding - the TACOS Frontend

This section describes in detail the modules that comprise the TACOS frontend - *i.e.*, the part of TACOS dedicated to mapping the sentence typed in by the user into a LISP function that modifies the **context** structure. The cycle is as follows:

1. the **read-input** function reads a stream of characters from the terminal and produces a string;
2. the **scan** function turns that string into a list of tokens, terminated with a special end-of-sentence marker;
3. the **expand** function expands macros in that list;
4. the **classify** function maps these tokens to a list of parse trees;
5. the **parse** function parses this input, producing one parse tree for the non-terminal symbol "sentence";
6. the **codegen** function transforms this parse tree into a LISP function

4.1 Read-input

Read-input is responsible for reading a sentence from some input stream. Any leading blank lines are ignored, and a trailing blank line indicates the end of sentence input. The module operates in two modes, interactive and non-interactive. In interactive mode, a prompt is printed before each line is read: a primary prompt for the first line, and a secondary prompt for each line thereafter. In the example input sentence that was demonstrated in Section 2, the default prompts were shown, where "Tacos>" was the primary prompt, with "con't>" being displayed for each line of sentence thereafter.

A blank line was selected over the more commonly used period for the purpose of terminating input due mostly to our own experiences with natural language processors - it took quite a while for many people to get used to ending each sentence with a period.

4.2 Scan

The **scan** module performs lexical analysis of a string, returning a list of tokens. The tokens are either symbols, strings (corresponding to quoted strings in the input), numbers of some standard type, or lists of numbers and/or strings. **Scan** issues warnings about ill-formed

input (non-terminated quoted strings, non-terminated lists, illegal characters) but will never abort with an error.

Lexical analysis was done using a pseudo-readtable implemented with a hash table. This method was selected over a customized LISP readtable for two reasons; a pseudo-readtable has demonstrated a faster execution time and a customized LISP readtable is prone to changing output in generally unpredictable ways when modified.

All of the macro functions used by the **reader** are implemented at the user level. This was necessary for a variety of specific reasons, all of which are a manifestation of the problem that the standard LISP **read** function is *not* robust, and it was easy to draw **read** into a LISP error. As an example, if a control character is encountered when reading a symbol name, then LISP will signal an error. This is one of the reasons that we needed to map all alphabetic characters to a symbol-parsing routine.

4.3 Expand

Expand performs macro expansion on a list of tokens, in which any macros are immediately expanded in line. Implementation of this module is a different problem from parsing for one big reason: a bottom-up parser takes the shortest production it can, and the macro expander takes the longest. Cycles are avoided in this way: when defining *X* as *Y*, actually define *X* as *expand(Y)*. (Then when you perform the macro expansion by replacing *X* with *expand(Y)*, there is no need to also expand the replacement string, *expand(Y)*, since it has already been expanded. In other words, macros are always defined relative to previous macro definitions.)

The expand module uses a list of macro structures to do its work. Each macro contains two parts:

- The target. This is a list of tokens - when this list is encountered in the token list, then it can be superseded by the replacement.
- The replacement. This is another list of tokens, possibly empty.

Expand starts at the beginning of the token list, and begins matching targets to it, starting with the longest ones. When it finds a match, then the part of the sentence corresponding to the target is stripped off and superseded with the replacement, which is a parse tree, and **expand** continues expanding the remainder of the token list. If no match is found, then expand continues by trying to match macros that start at the second token, then the third, etc. and so on.

Expand works in time O(n)O(m) where n is the length of the sentence and m is the number of macros. There are more efficient methods to do this expansion, but they call for more complex ways of programming. The method implemented is more than sufficient for the macros we were concerned with expanding, since the problem is similar to regular expression matching.

4.4 Classify

Classify manipulates a list of tokens by mapping them to a list of simple parse trees. **Classify** issues a warning for each unknown symbol. If there are any unknown symbols in the sentence, then it aborts with an error message after the entire sentence is classified.

Normally, this classification process is carried out by a tokenizer, which also performs lexical analysis duties. The reasons for splitting **classify** off into a different module are:

- It makes the **scan** function more generally useful. It is used, for instance, in the table reader.
- It makes the **expand** function both generally useful (although not, as yet, generally *used*) and easier to write.

For implementation reasons, it's nice for the parser to deal with nothing except parse trees. Thus, **classify** outputs a list of "primitive" parse trees - trees without any subtrees. The "type" field of the generated tree indicates the part-of-speech/token-type of each token, and the "munger" field is a function that, when evaluated by **codegen**, will generate LISP code for the tree (see Section 4.6 for details.) These are obtained via lookup in the lexicon.

The lexicon is a hash table that maps symbols to the part-of-speech of that symbol, and parts-of-speech to the function needed by **codegen**.

4.5 Parse

Parse takes as input a list of "primitive parse trees" and outputs a single parse tree. The parser is data-driven, controlled by a set of grammar rules read in from a file.

The parser module is a bottom-up, shift-reduce parser; the end product of this method of parsing is a stack of parse trees. Each tree represents some recognized grammatical constituent (perhaps only a token.) The parse is deemed succesful if and only if the stack, after parsing is complete, contains a single parse tree representing a sentence constituent (sentence is thus hard-wired in as the distinguished symbol of the grammar.) If the parse is not successful, an error message is generated which is basically a dump of the parse stack; for example,

```
Error: I don't understand the sentence: if <comparison-list> then <sentence>
```

The heuristics for generating hints are described below.

The most usual way of parsing natural language is with an ATN parser - i.e., top-down with backup. For several reasons, have departed from this conventional wisdom and used a bottom-up parser without backup.

First, there is a maintainability issue. ATN grammars are programs, not data, and as programs they are less readable than most. Also, the ATN representation for a grammar doesn't look anything like the BNF for it, which is what most competent grammar-designers would think in. Writing ATN grammars requires an intimate understanding of how ATNs

work, which is not a common piece of knowledge to have. In contrast, a shift-reduce parser runs off reasonably comprehensible data.

Second, there is an efficiency issue. An ATN interpreter is essentially a branch-and-bound search program, looking for possible parses. This can get expensive when you have certain types of conjunctions. It gives one some extra power and flexibility, but at a cost.

Finally, top-down parsing with backup gives, at least in our experience, no useful information when a parse fails about why it failed or how to fix the problem. In our opinion, this is a grievous shortcoming. The stack resulting from a bottom-up parse contains a great deal of information about the sentence and why it didn't parse. Although not all of the information provided is used, this fact did influence the decision on which technique to implement.

Bottom-up parsing is a poor technique to integrate with backup and search. To compensate for the lack of backup, two techniques are used. First, any grammar rule can look ahead an arbitrary distance into the input stream. Second, the code generated for a parse tree can depend on the context in which it appears. For instance, the parse tree for a field may generate the name of the field in one context, and a function to access the field in another context.

This brings us to another decision: the close integration of the parser with the codegen module. Every grammar rule contains an associated function that will produce LISP code that corresponds to the parse tree that was input for that rule. Attaching this semantic information to the grammar has obvious advantage; it eliminates the necessity for a file parallel to the grammar file that would be used by the code generator.

The following heuristics are used to provide hints after an unsuccessful parse:

- First, the parser looks at the first token in the token list. If this is a verb or the keyword that marks a major clause, then a summary of the syntax for that clause is given.
- If this fails, then a list of allowable clauses, with a brief description of each, is given.

A special free list is kept of parse tree structures, rather than relying on the LISP garbage collection routines, thus increasing the speed of the parser by a large factor. The code for the parser itself is perhaps a little opaque; the subroutine structure was carefully chosen for efficiency rather than for simplicity.

Some future enhancements to make the parser module more effective would include:

- Some consistency checks on the grammar - at least enough to catch spelling mistakes, etc.
- Although this parser handles short sentences effectively, it probably would be much less effective on a program-sized parsing problem. To increase performance might require some sort of indexing scheme; for instance, using a modified digital search tree at a single level might increase parsing speed dramatically.
- Eventually replace the parser module with a LR(k) (right parser works deterministically, if allowed to look k input symbols ahead at each step) parser-generator like YACC. These types of parsers operate in strictly linear speed.

4.6 Codegen

Codegen takes as input a single parse tree, and generates a LISP structure that corresponds to that tree.

The munger functions used by **codegen** are modeled after YACC actions. The same special variables **$1**, **$2**, ... are used to represent the code generated by the first, second, ... subtrees of the tree being generated. In addition, these variables have been added:

- ***1**, ***2**, ... represent the first, second, ... subtrees themselves.
- ***tree** is the tree being **codegen**ned.
- **$subtrees** is a list containing **$1**, **$2**, ...
- ***subtrees** is a list containing ***1**, ***2**, ...
- ***env** is the *environment* (see below).
- ***len** is the number of subtrees.

All of these are used somewhere in the TACOS grammar. The YACC pseudo-variable "**$$**" isn't needed - LISP conventions for returned values are used instead.

The code generation process may be best explained with an example. Consider this grammar rule:

```
(expr --> expr '+ addend
   YIELDING '(+ ,$1 ,$2))
```

When this production is used, a parse tree T is produced; attached to T will be a pointer to a function somewhat like this:

```
(lambda (*subtrees *env *tree)
   (let (($1 <code generated for first subtree>)
         ($2 ...))
      '(+ ,$1 ,$2)))
```

When (**codegen** T) is called, this function will be called with the appropriate parameters. To find values for **$1** and **$2**, as required by the **let** special form, **codegen** is essentially called recursively (actually, a lower-level routine is called.) The body of the **let** statement is then executed, producing LISP code to evaluate the **expr**.

Code generation is thus done top-down. This is necessary for the following important reason. Often, the meaning of a non-terminal symbol (e.g., the code that must be produced for it) depends on the context on which the symbol appears. To take an example from the TACOS grammar,

```
<set-list> and <set-list>
```

is normally interpreted as a union of two sets. However, in this sentence

```
all proposals in <set-list> and <set-list>
```

the conjunction must be interpreted as an *intersection*.

The parser usually has no way of knowing, when it parses

```
<set-list> and <set-list>
```

what context the conjunction will eventually appear in. Sensitivity to context is then left up to the **codegen** module. Codegen routines can determine what context they are used in by checking the environment for signals passed by some higher-level function.

The environment is implemented as an association list. A routine can pass a signal to the n-th subtree with the **send** macro; instead of using $\$n$, it uses the macro call (**send** $*n$ **'signal**). The signal can be received by the called function with the macro (**receive**), which always returns the most recent signal left for that function.

5 The Context Structure

The **context** structure is a special structure that contains the "content" of a sentence. The **context** structure can be thought of as a frame representing a sentence. The slots in the frame are populated by evaluating the function that is created by the **codegen** module.

Part of what the context structure contains are a set of default values for what set of objects to operate on, what functions to display, and so on. These values are permanent; they are not changed unless they are explicitly overwritten by a command. Permanent context values include:

- The *consider set*, the last of proposals (more generally, objects) specified.
- The *display set*, which describes the columns to display in a table.
- The *change set*, which describes a list of fields to modify.
- The *order set*, which describes a list of functions to order by.
- The *last repeatable verb* used in a sentence. A TACOS command need not contain a verb; it may instead specify a new set of proposals, for instance. In this case, the last repeatable verb is used to complete the ellipsis. Repeatable verbs are **display**, **edit**, and **output**.

Other slots of the context structure hold temporary values. The values are like permanent values, but the grammar requires that a command populate them before they are used. They are in context not to ensure that they are preserved from one command to another, but as a convenience; the **context** structure is the only mechanism for communication with the backend module.

These **communication** slots are:

- The *partition* slot, which holds the file name of a partition to **load** or **create**.
- The *called-procedure* slot, which holds the number of the procedure to **call**.
- The *define set*, a set of values to define, or to retrieve the definitions of. Currently, the grammar only allows you to put a single value in this set, but software does support processing a list of define values.

Another type of slot is the temporary slot. Temporary slots need not be populated by the grammar, but are not permanent because the last value specified is not likely to be re-used. Temporary slots are reset to a default value before the function produced by **codegen** is evaluated. Temporary slots include:

- The *verb indicator*, which defaults to the last repeatable verb.
- The *display command*, a VMS command to display a table. This defaults to a command that invokes the TACOS examiner, which allows a user to view fields they have specified in a full screen format.
- The *field selector*, which defaults to **nil** (indicating to prompt the user for a value.) This determines what selector value of fields will be changed in an edit command.
- The *format string*, which indicates the format of an output statement. Its default value is **nil**, which means to use a default output reporting format.

A final type of slot in the context structure is the internal slot. Internal slots are used mostly for procedure definition and conditional execution.

Each of these types of slot is, of course, treated differently by the software. Together, the slots completely describe a sentence, and provide a clean and well-organized interface between the TACOS frontend and the TACOS backend.

6 Command Execution - the TACOS Backend

This section describes the **backend** of the TACOS on-line system. The backend takes as input a function produced by codegen, and then uses that function to modify the context structure. It then looks up, from the verb indicator in the context structure, what low-level routine to use to execute the user's command. Finally, that routine is executed, and **backend** returns.

The value that is returned by backend is one of:

- **'t**, to indicate normal execution
- **'exit** to indicate that an **exit** verb was processed and that the session should terminate.

Backend does a considerable amount of consistency checking of the context structure. If a check fails, then **backend** throws an appropriate error message.

Backend uses a number of other modules to do its work. Some of these modules are quite complex in their own right. These modules are:

- **Codeeval**, which evaluates the code produced by codegen and thus makes the necessary changes in the context structure.
- The **database** module, which handles access of the TACOS database.
- The **reader** and **formatter**, which handle input of tables and output of tables, respectively.
- The **procedures** module, which handles storage and retrieval of predefined procedures.
- The **logging** module, which maintains a log of changes to the database.
- The **field_prot** module, which provides some security to the database in the sense that it restricts write access to parts of certain fields.
- The **system** module, which makes calls on the operating system.
- The TACOS examiner (a full screen viewing facility) and the TACOS editor (provides full screen editing capabilities).

7 Conclusion

The TACOS system has been throughly exercised in a series of mock TAC meetings and the response from the users has been very favorable. The power and flexibility of the queries was demonstrated during these trials. Another feature that drew favorable comments from the users was the ability to customize the dialect via macros and predefined definitions. Up to six simultaneous TACOS systems were run during the trials on a VAX 8600, along with other STScI users. Performance was acceptable. Increased speed for initializing the system was obtained by increasing the priority of the TACOS process, yet this did not significantly degrade performance for non-TACOS users. We are planning to move the initialization procedures into a suspended version of the TACOS system, which will insure even greater time savings.

We have described how the TACOS system supports the HST proposal selection process in several ways:

- A flexible, easy-to-use, English-like command language
- A reliable on-line help facility
- A flexible grammar with free formatting of sentence input
- A quick and helpful diagnosis of erroneous input

The design of TACOS includes several innovative technologies which are useful in the creation of a natural command language, including:

- Using a bottom-up, shift reduce parser for sentence understanding.
- Modular development of functions to allow maintenance of all grammar rules, macro definitions, database field security, and keyword phrases to be made with requiring recompilation.
- Implementation of the system in a highly portable language.

The result of our efforts is a easy-to use, extensible command language system that is applicable to many other problems.

References

[1] *SO-05 Proposal Solicitation and Review, Volume 1*, STScI 85051A, Final, August 1985

[2] Aho, A. V. and Ullman, J. D., *Principles of Compiler Design*, Addison-Wesley Publishing Company, 1977

[3] Winograd, T., *Language is a Cognitive Process*, Addison-Wesley Publishing Company, 1983

Maintaining Consistency Between Planning Hierarchies: Techniques and Applications

David R. Zoch

Ford Aerospace & Communications Corporation
Space Missions Division
College Park, MD 20740

ABSTRACT

In many planning and scheduling environments, it is desirable to be able to view and manipulate plans at different levels of abstraction, allowing users the option of viewing and manipulating either a very detailed representation of the plan or a high-level more abstract version of the plan. Generating a detailed plan from a more abstract plan requires domain-specific planning/scheduling knowledge; the reverse process of generating a high-level plan from a detailed plan (Reverse Plan Maintenance, or RPM) requires having the system "remember" the actions it took based on its domain-specific knowledge and its reasons for taking those actions.

This paper describes this reverse plan maintenance process as implemented in a specific planning and scheduling tool, The Mission Operations Planning Assistant (MOPA), as well as the applications of RPM to other planning and scheduling problems, emphasizing the knowledge that is needed to maintain the correspondence between the different hierarchical planning levels.

PROBLEM

In many planning and scheduling environments for space applications (e.g., scheduling instrument operations on board a satellite, Space Station Payload Scheduling, etc.) a daily schedule may consist of a complicated mixture of hundreds/thousands of payload activities and operations activities. It is difficult to detect any high level organization or overall plan from this detailed

schedule. These complicated schedules, however, are usually generated from higher level plans, which are valuable to the mission planners since they make it possible to get a more abstract view of the mission objectives and instrument operations for the day.

If schedules remained static, there would be no problem; mission planners have access to both high level plans and detailed schedules. Unfortunately, this is often not the case: schedules change due to changing mission objectives, instrument failures, and targets of opportunity (a sun-observing instrument might want to reposition itself to observe a solar flare, for instance). It is often much simpler to make these changes at the detailed schedule level instead of making the changes to the higher level plan (re-generating the detailed schedule might take hours or days). This obviously causes a difference in the two representations, invalidating the more abstract plan. This, then, is the purpose of Reverse Plan Maintenance: to rectify the difference between the high level plan and the detailed schedule so that the high level plan "agrees" with the modified detailed schedule.

Benefits of Reverse Plan Maintenance

The benefits of RPM in a planning and scheduling application are:

1. A reduction of scheduling mishaps due to discrepancies in plan representations.

2. Immediate feedback on the high-level impacts of making changes to a schedule at a very detailed level.

3. Makes "what-if" scheduling practical, since the entire schedule does not need to be regenerated.

4. Saves computer time and manpower since entire schedules do not need to be regenerated.

An Appropriate Environment for RPM

Reverse Plan Maintenance is not a "necessity" in many planning problems. For instance, if it is only necessary to make changes to a plan at the more abstract planning level and if the detailed schedule generation process is simple, then there is no reason to support RPM. The following criteria are useful in deciding if Reverse Plan Maintenance is appropriate for a specific planning and scheduling problem:

1. A high level (condensed) representation of a plan is desirable and possible, due to some organization (patterns) in the resulting detailed schedule.

2. The detailed schedule resulting from the expansion of the plan is "too complicated," i.e., it is difficult to perceive any organization or overall strategy in the schedule (for instance, it may contain hundreds or thousands of activities, some of which interact with each other).

3. It is desirable to make changes to the more detailed schedule.

4. It is desirable to view the modified schedule in its condensed version so that it can be quickly assimilated by a human planner.

The following pages describe a specific application (for mission planning on the Upper Atmospheric Research Satellite, UARS) where Reverse Plan Maintenance is beneficial and has been implemented.

AN EXAMPLE APPLICATION: MOPA

Upper Atmospheric Research Satellite Background

The Mission Operations Planning Assistant (MOPA) is a knowledge-based system developed by Ford Aerospace for NASA Goddard Space Flight Center to support mission planning for the Upper Atmospheric Research Satellite (UARS), a multi-instrument orbiting observatory scheduled to be launched by the Space Shuttle in 1991. UARS will provide experimenters at remote locations with data on the temperature, composition, and dynamics of the earth's upper atmosphere.

Mission planning for a satellite such as UARS is a complex process since the activities of the ten instruments on board must be defined and coordinated with the actions of the satellite (for instance, most instruments must be put into a safe state whenever the satellite is firing its thrusters.) Additionally, there are "cooperative constraints" between instruments (for instance, two scientists may want to cooperate on an experiment which requires that each has his instrument in a certain mode at the same time) and operational

constraints that if violated could cause damage to an instrument (for instance, aiming a solar/stellar instrument at the sun while it is in one of its star viewing modes).

If the scientists in charge of the instruments on board UARS were all located at Goddard Space Flight Center, these problems could be worked out in daily meetings, as is often currently done for other missions. Since the instrument scientists are located throughout the country, however, this was not feasible, and a more automated approach, i.e., MOPA was taken. An additional motivation for this automated planning approach is that it is applicable to other situations where it is not feasible for the payload scientists to have daily planning meetings, e.g., Space Station.

UARS Mission Planning

The first planning step that the Mission Planning Group (MPG) must do is to create a Daily Science Plan, which is a description of UARS instrument functions to be performed during a day's operation. The DSP is created using the Long Term Science Plan (which is developed by the instrument scientists) as a guideline. After the creation of the "first cut" DSP, the scientists are allowed to review and request modifications to the plan. The MPG acts as a coordinator and negotiator of plan modification requests which will arise due to variances in instrument performance, changing scientific objectives, and the occurrence of targets of opportunity. The MPG must coordinate with the scientists and with the flight operations team to ensure that the Daily Science Plan does not violate current instrument and spacecraft capabilities and constraints.

After the DSP has been approved, the MPG develops detailed operating plans (known as Activity Plans) which are lists of activities for the UARS instruments and their corresponding times of execution. The activities themselves are predefined command sequences which configure an instrument to perform a certain operation.

MOPA Planning Support

MOPA supports the UARS planning process by providing a plan representation analogous to each level of planning in the MPG planning process, and facilities to manipulate these plans. In addition to these three types of plans, the other major data structures in MOPA are ACTIVITIES and EVENTS. The following paragraphs describe each of these

structures.

EVENTS in MOPA are anything that a scientist might want to use in order to trigger his instrument to enter a certain mode. The events may be satellite events (e.g., a yaw maneuver), orbital events (e.g., acquisition of sun), special observational events (e.g., a volcano), or pseudo-events such as a specific time or the beginning of a specific orbit. Figure 1 shows a part of a formatted EVENT file, which contains a list of events for a specific day. (EVENTS are represented and manipulated internally using schema, the data structure provided with the Automated Reasoning Tool (ART) by Inference Corporation. Generic Plans, Daily Science Plans, Activity Plans, and activities are also represented using ART schema. MOPA contains formatting programs to display these to the user in a more readable form. The internal representation used by MOPA is described later in the Knowledge Representation section.)

Start Time	Name	Duration	Priority
1:08	SLEW	:11	4
1:22	SUN	00:37	2
1:36	ORBIT-2	01:36	2
1:38	ASC-NODE	00:49	2
1:53	TDRS-EAST	00:15	2
1:59	NIGHT	01:01	2

Figure 1. Some Sample Events from an EVENT-FILE

ACTIVITIES in MOPA define command sequences that can be executed by an instrument's microprocessor or other hardware in order to perform some arbitrary function. Both Activity Plans and Daily Science Plans consist of sequences of activities, but the activities in each are at different levels of abstraction: the DSP activities are "compound" activities that represent high level instrument operations in terms that the scientists themselves have defined; AP activities are primitive activities that correspond to one instrument command.

There are three types of plans in MOPA: Generic Plans (GP's), which correspond roughly to the notion of a Long Term Science Plan, Daily Science Plans (DSP's), and Activity Plans (AP's). The planning process in MOPA begins with the Generic Plan, which is a high level

event-driven specification of the ACTIVITIES that an instrument is to perform based on the specified EVENT. Figure 2 shows a part of a sample Generic Plan. The part shown is for the SUSIM (Solar Ultraviolet Spectral Irradiance Monitor) instrument. (A complete Generic Plan has a plan for each of the instruments on board.) The plans are generic in that they (1) Describe the operation of each instrument under a wide variety of circumstances, many of which may not occur on a given day, and (2) Are possibly appropriate for many days (or even months), so they can be "reused," probably with some minor modifications, which are made via Reverse Plan Maintenance.

SUSIM plan of operations:

* On every YAW perform:
  1. SAFE-FOR-YAW 2 minutes before YAW until the end of YAW

* On SUN occurrences 2 10 perform:
  1. 10A-SPEC-SCAN for 36 minutes

* On SUN occurrences 4 12 perform:
  1. 1A-SPEC-SCAN for 36 minutes

* On every SUN Perform:
  1. CONTINUOUS-MONITOR for 37 minutes

* On every NIGHT Perform:
  1. ELECTRICAL CALIBRATION for 1 hour 35 minutes.

Figure 2. A Generic Plan for the SUSIM Instrument

The Generic Plan is used in conjunction with a daily EVENT schedule to create a Daily Science Plan for that day. Figure 3 shows part of a DSP. As previously mentioned, the activities in the DSP are actually "compound activities" in that they may represent many actual instrument commands. For instance, the "1A-SPEC-SCAN" activity (1 angstrom spectral scan) might actually consist of opening a viewing door, adjusting the wavelength by rotating a filter wheel, and then closing a door at the end of the viewing period. The instrument scientists are allowed to group arbitrary combinations of activities together in this way and define the resulting "compound" activity for use in Generic Plans (Actually,

"compound" activities can be parts of other compound activities, resulting in arbitrarily deep hierarchies of activities.)

Start Time	Instrument	Name	End Time
19:20	SUSIM	10A-SPEC-SCAN	19:56
19:56	SOLSTICE	SAFE-FOR-SLEW	20:06
19:57	SUSIM	ELECTRICAL-CALI*	20:58
20:06	SOLSTICE	STAR-OBSERVATION	20:22
20:22	SOLSTICE	SAFE-FOR-SLEW	20:28

Figure 3. Some Activities from a DSP

An Activity Plan is created from a DSP using the compound activity definitions. Figure 4 shows some activities from an Activity Plan. The formats of the DSP and AP are identical, the difference is a conceptual one: the DSP contains "compound" activities; the AP contains only "primitive" activities.

Start Time	Instrument	Name	End Time
19:57	SUSIM	ELECTRICAL-CALI*	20:58
20:06	SOLSTICE	OPEN-APERTU-DOOR	20:07
20:07	SOLSTICE	ADJ-WAVE-LENGTH	20:21
20:22	SOLSTICE	CLOSE-APER-DOOR	20:23
20:23	SOLSTICE	SOLSTICE-WAIT	20:28

Figure 4. Some Activities from an Activity Plan

MOPA provides a menu driven interface which makes it easy for the user to retrieve and save Generic Plans, create DSP's from a GP, and create AP's from the DSP. Additionally, MOPA provides a graphical representation (activity time lines) of both DSP's and AP's (Figure 5 shows the graphical representation of a DSP). The activities in the time lines are mouse-sensitive, which allows the user to replace, add, and delete activities simply by clicking the mouse on the appropriate activity and choosing the appropriate operation.

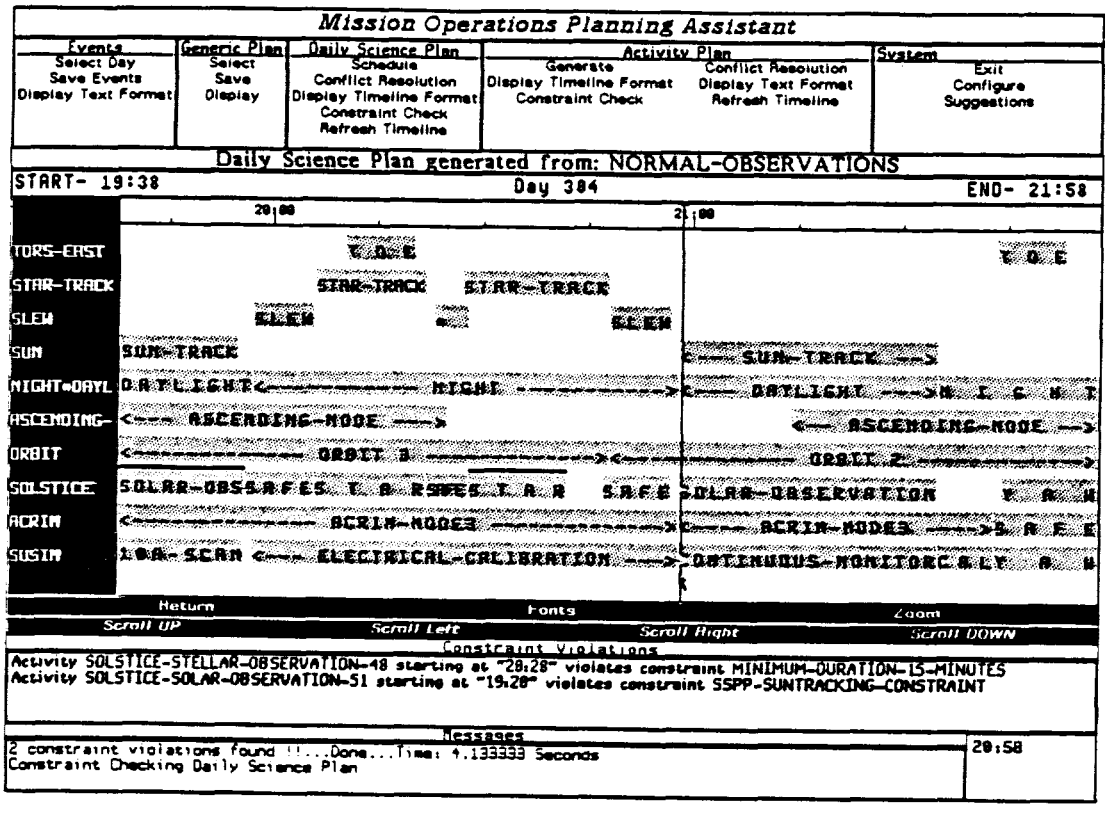


Figure 5. Graphic Representation of a DSP

KNOWLEDGE REPRESENTATION

Before tackling the internals of Reverse Plan Maintenance, it is important to understand the internal representations of events, activities, and generic plans. This section focuses on the representation of these entities and how the knowledge is used for generating Daily Science Plans and Activity Plans.

Events

As previously mentioned, the process of planning and scheduling instrument activities is driven by the nature and time of occurrence of different events. For example, the period of time during an orbit that the Sun is visible (spacecraft day) is an event which triggers the sun-viewing activities of several instruments on the UARS.

The event classes used in MOPA are represented in a classification hierarchy. At the root of this hierarchy

is the definition of the generic event as shown in Figure 6. The attributes of the generic event schema are inherited by the rest of the event classes.

Generic Event schema

```
(defschema event
  (name)
  (short-name)
  (has-activity)
  (priority)
  (start-time)
  (end-time)
  (duration))
```

Figure 6

The attributes name, short-name, priority, start-time, end-time, and duration are common to events and activities. The name and short-name attributes provide a printed representation of the event for use by the MOPA time line and textual displays. Values of these attributes may be string or symbolic types. The start-time, end-time, and duration attributes will contain the scheduled times (in numbers of seconds offset from the start of day) of the instances of the events. The has-activity relation relates an event schema to the activities which are triggered by it. This link is used during the RPM process and its values are activity schemata.

Activities

Central to the design of MOPA is the representation of UARS instrument activities. Activities are represented in MOPA as collections of ART schemata. This section describes the details of the activity representation scheme.

Basic Attributes-- All activity schemata have certain basic attributes associated with them. The generic activity schema is shown in Figure 7. The attributes name, short-name, priority, start-time, end-time, and duration have the same meaning as they do for events. There are several relations defined for activities that link them to other schema types. The activity-of relation provides a link between an activity and the instrument with which it is associated. The priority slot is used to resolve conflicts between activities that are scheduled to occur at the same time. The priority slot gets its value

from the event that triggered it. This has the effect of giving priority to activities that occur as a result of higher priority events. For example, a GP might specify for an instrument to respond to both a spacecraft yaw maneuver and the acquisition of sun. Since the instrument might be damaged if it does not put itself into a safe mode during the yaw maneuver, the YAW event has been assigned a higher priority than "acquisition of sun" and any activity triggered by the YAW event will have priority over any activity triggered by the "acquisition of sun" event.

Generic Activity Schema

```
(defschema activity
  (name)
  (short-name)
  (has-event)
  (activity-of)
  (priority 0)
  (start-time 0)
  (duration 0)
  (end-time 0)
  (has-constraint-violation)
  (has-sub-activity)
  (sub-activity-of)
  (fixed-duration)
  (has-pre-condition)
  (delta-time 0)
  (restriction))
```

Figure 7

Key Slots for RPM-- The has-event relation relates an activity to the event instance that caused its selection and scheduling. The restriction attribute value is passed down from the Generic Plan entry that specified the activity. This value is the "event restriction" value (EVERY, NUMBERS, or TIME) from the DSP entry. The value of the restriction attribute is used during the RPM process to determine the "triggering-event-specification" (refer to the BNF in Figure 8) in the Generic Plan that caused the activity to be scheduled. The has-sub-activity relation links an activity to its detailed components; the sub-activity-of relation provides a link in the opposite direction. These links are used when doing RPM from the AP to the DSP.

The delta-time attribute determines when an activity should start in relation to the event which triggered it.

This value is a negative or positive integer specifying the number of seconds before or after the event start-time that the activity should begin.

Activities also have several other slots; these are not described here since they are not important for the RPM process.

Generic Plans

The Generic Plan (GP) is an encoded description of the relationship between Events and planned instrument activities to occur based upon the Events (e.g.., during Spacecraft Night, Susim should perform Electrical Calibration). This event-driven specification of the daily operations of an instrument is encoded as an ART schemata.

Figure 9 illustrates a sample plan schema for the Susim instrument. The plan-of relation links the instrument-plan to the instrument which executes the plan. The triggering-event slots determine the activity that Susim will perform when the specified event occurs.

Triggering-event Syntax

```
<triggering-event-specification> :=
      triggering-event ( <event-name>
                         <event-restriction>
                        (<activity-specification>*)
<event-restriction> := EVERY |
                       <number-specification> |
                       <time-specification>
<number-specification> := (NUMBERS <numbers>+)
<time-specification> := (TIME <time>+)
<activity-specification> :=
          ( <activity-name> <attribute-value-pair>*
            [DURATION <duration-specification>]
<attribute-value-pair> := <activity-attribute-name>
                          <attribute-value>
<duration-specification> := <time-specification> |
                            EVENT | END-EVENT
```

Figure 8

For example, the "triggering-event (night-event ..." entry is interpreted to mean "the Susim instrument should perform a SUSIM-ELECTRICAL-CALIBRATION activity at every occurrence of the spacecraft NIGHT-EVENT". The "triggering-event (sun-track (numbers 4 12)" means that on the fourth and twelfth occurrences of the SUN-TRACK event,

Susim should perform the SUSIM-1A-SPECTRAL-SCAN activity. A "numbers" event-restriction entry overrides an "every" event-restriction entry. If both of these types of event-restriction are present (as is the case in the Susim plan) the "every" specification is interpreted to mean "on every event, except the occurrences specifically mentioned, perform ...".

The triggering-event entry for SPACECRAFT-YAW-EVENT in the Susim plan illustrates attribute/value pairs which can be used to override the default activity slot values for the activities created during the DSP generation process. In this case the delta-time attribute of the SUSIM-MAN-PREP activity is assigned the value -120 (i.e. 2 minutes before). The duration is specified to be "end-event" meaning the SUSIM-MAN-PREP activity should continue until the end of the SPACECRAFT-YAW-EVENT (a numeric duration could have been specified instead).

Susim Plan schema

```
(defschema susim-plan
  (instance-of instrument-plan)
  (plan-of susim)
  (triggering-event
    (night-event every
((susim-electrical-calibration))))
  (triggering-event
    (sun-track every
((susim-continuous-monitor))))
  (triggering-event
    (sun-track (numbers 4 12)
((susim-1a-spectral-scan))))
  (triggering-event
    (sun-track (numbers 2 10)
((susim-10a-spectral-scan))))
  (triggering-event
    (spacecraft-yaw-event every
      ((susim-man-prep delta-time -120
                       duration end-event)))))
```

Figure 9

REVERSE PLAN MAINTENANCE IN MOPA

As previously mentioned, RPM allows the user to edit the graphic displays provided by MOPA at either the activity plan level or the DSP level and "reflect" these changes back to the high level plan for the day, the

generic plan. This plan can then be saved for later use. This saves the user from having to edit the generic plan and regenerate both the DSP and the AP, allowing him to make small changes in a number of seconds and still have three consistent representations.

Currently, the MOPA prototype supports RPM only for what we think is the most common type of modification a user will make to a plan: replacing one activity with another. Activity Replacement with RPM is supported for activities at both the DSP and AP levels. Other types of RPM's such as activity deletions, additions, and modifications (such as increasing the duration of an activity) are not currently handled, but should be for an operational system.

An Example

Figure 10 shows a part of a generic plan for the ACRIM instrument. According to the plan, ACRIM should be in MODE-3 for every SUN-TRACK event except for the tenth one; in this case it will be in MODE-7.

```
(defschema acrim-plan " "
    ...
 (triggering-event
   (sun-track every ((acrim-mode3 duration 8000))))
 (triggering-event
   (sun-track (numbers 10) ((acrim-mode7 duration 8000))))
    ...)
```

Figure 10. A Part of a plan for the ACRIM instrument.

Suppose that the user has decided that he does not want ACRIM to be in MODE-3 during the second SUN-TRACK event; instead, he just wants to put the ACRIM instrument in a wait mode. To make this change, he simply brings up the graphic display of activities (either the DSP or AP), "clicks" the mouse on the ACRIM-MODE-3 that he wants to replace, and then "clicks" the mouse on the replacement activity, ACRIM-WAIT (a menu of possible replacement activities will appear when the user clicks on the "replace" option). The MODE-3 activity will be replaced in the graphics display by a WAIT activity of the same duration. As part of the replacement operation, the current plan is modified as shown in Figure 11.

```
(defschema acrim-plan
          " "
        ...
  (triggering-event
    (sun-track every ((acrim-mode3 duration 8000))))
  (triggering-event
    (sun-track (numbers 10) ((acrim-mode7 duration 8000))))
  (TRIGGERING-EVENT (SUN-TRACK (NUMBERS 2)
     ((ACRIM-WAIT DURATION 1000)))))
```

Figure 11. ACRIM plan after RPM.

This plan says to do something special, i.e., put ACRIM in the WAIT mode, on the second occurrence of SUN-TRACK (Changes made because of RPM are capitalized). If this plan is then saved, it can be used to precisely regenerate the current activities for the current day. Additionally, this might be an appropriate plan to use on other days.

Special Knowledge Needed for Reverse Translation

In order to be able to modify the generic plan based on changes made at other levels, it is necessary to know what part of the generic plan caused the creation of an activity. For instance, if an activity such as ACRIM-WAIT-1 is replaced with ACRIM-MODE-4, it is important to know that the line:

```
(triggering-event
   (sun-track (numbers 3) ((acrim-wait duration 1000))))
```

caused the generation of the WAIT activity, since this is the line that must be altered. The necessary information is the instance of the event that triggered the creation of the activity (stored on the HAS-EVENT slot), and the restriction from the generic plan (the RESTRICTION slot). These two "attributes" are assigned to each activity when it is created so that reverse translation will be possible.

Reverse Translation at the DSP level

The main parts of reverse translation when a change has been made at the DSP level are:

1. Find the entry in the generic plan that caused the replaced activity to be generated and determine whether to modify this entry, add a new entry, or both.

2. Figure out the slots of the new activity that should be included in the plan (i.e., don't explicitly include an attribute in the Generic Plan if its value is the same as the default value.

The following sections explain each of these in more detail.

Finding the Generic Plan Entry to Modify-- Since each activity knows the exact occurrence of the event that caused it to be generated (the HAS-EVENT slot) it is straightforward to find all of the entries that could have possibly caused the creation of the current activity. The three possible cases with their resolution strategies are:

1. There is only one entry with the proper triggering-event, and it has an EVERY restriction. (In this case, an extra entry is added to the generic plan; the extra entry will specify what to do in the special case and will have a restriction slot such as "(numbers 8)".)

2. An activity with a restriction slot with a value such as "(numbers 4)" is being modified; in this case, this entry is just modified to reflect the new activity. (There might also be an entry for the same triggering event with an EVERY restriction. This entry can be ignored since the more specific restriction has precedence.)

3. An activity is being replaced that has a restriction slot such as "(numbers 4 10 12)". In this case, this entry must be modified to something such as "(numbers 4 10)" (assuming that the activity that happens on occurrence "12" is being replaced), and a new entry with a restriction slot of "(numbers 12)" is added with the new activity.

Determining the Attributes of the new activity-- The actual attributes of the new activity (its duration, start time, priority, triggering event, etc.) are known, since they are identical to the attributes of the activity that it replaced; the only difference is the actual activity. The purpose of this part of the algorithm is to (1) minimize the amount of information that is copied back into the plan (it is not necessary to specify information that is the same as the default value), and (2) to specify the duration of a replacement event as "END-EVENT" or "EVENT," when appropriate, instead of just using a numeric value.

Reverse Translation at the Activity Plan level

There are several possible situations when doing replacement at the AP level. A user might replace an activity that is part of a known hierarchy with another activity, thus destroying the hierarchy; alternatively, he might replace an activity in a hierarchy with another one that causes a new hierarchy to exist. For instance, if A/B/C is a known hierarchy called A1, and A/X/C is a known hierarchy called AX, then replacing B with X causes AX to replace A1 at the Generic plan level. If A/X/C were not a known hierarchy, then A1 is replaced with the list of activities, A/X/C. A third possibility in doing replacements at the AP level is that an activity that is not a part of any hierarchy is being replaced: in this case, the processing is identical to that for processing a replacement at the DSP level.

OTHER APPLICATIONS OF RPM

Many AI systems are said to be "intelligent" systems or "expert" systems since they use the intelligence of an expert (often in the form of rules) to make the same decisions that an expert would under the same conditions. RPM is a step towards "intelligent" systems in a different sense: the system "knows" and "remembers" why it has done certain things and can use this knowledge for other purposes. Many rule-based systems can perform a back trace of rules fired in order to let the user know how it arrived at a certain conclusion. The concept in RPM is to have the system itself effectively use this knowledge to accomplish other goals; in MOPA, the knowledge is used to allow fast incremental changes to a schedule; other applications are also possible.

For example, in a scheduling system that consists of a number of activities competing for limited resources, a scheduler might remember that it could not schedule activity A2 because it conflicted with a higher priority activity, A1, which consumed the remaining resources available at that time. If A1 is later moved or deleted, the scheduler could instantly "know" that A2 can now be scheduled. Other activity-to-activity dependencies can similarly be remembered and used to accomplish quick rescheduling/replanning.

Missing Knowledge

In many cases it is just not possible to do the "inverse" of an operation as RPM does in MOPA; for instance, if an activity is added in MOPA, there is no way

for the system to "know" which activity "causes" the execution of this new activity. The system could certainly make some intelligent guesses, by looking at the events that usually trigger a certain type of activity, but it can never be sure. Prompting the user (or having the user validate the system's result) is probably the most reasonable solution in these cases.

SUMMARY AND CONCLUSIONS

The Reverse Plan Maintenance feature of MOPA is a time-saving feature that is implemented for the case of replacing one activity with another at either the DSP or AP levels. Other types of plan modifications are possible also, even though the user might have to be prompted for additional information.

The reverse plan maintenance process is an exemplary application to show what can be done with the knowledge that is typically available in a knowledge-based system. By remembering its decision paths and storing the information in a usable form, several seemingly difficult tasks can be easily accomplished.

Intelligent systems that use available knowledge can often exhibit their "intelligence" by not doing redundant re-computations; instead they can "know" what needs to be re-done and do only those calculations.

REFERENCES

1. M. Fox, "Constraint Directed Search: A Case Study of Job-Shop Scheduling". Ph.D. Thesis.,Computer Science Dept, Carnegie Mellon University, Pittsburg, PA 15213, 1983.

2. W. Gevarter, "Artificial Intelligence". Noyes Publications, 1984.

3. I. Goldstein and B. Roberts. "NUDGE, A Knowledge-based Scheduling System". The Fifth Internation Joint Conference on Artificial Intelligence, IJCAI, 1977.

4. Inference Corporation, "ART Reference Manual Version 2.0". Inference Corporation, 1986.

5. J. King, "RPMS: Resource Planning and Management System". JAI PCC 1984.

6. Symbolics Inc. "Reference Guide to Symbolics-Lisp". Symbolics Inc. 1985.

7. J. G. Schuetzle and D. R. Zoch, A Hierarchical Planning System For Mission Support. Expert Systems in Government Symposium, 1986

8. J. G. Schuetzle, The Mission Operations Planning Assistant, 1987 Goddard Conference on Space Applications of Artificial Intelligence and Robotics.

A Lisp-Ada Connection

Allan Jaworski, David Lavallee, David Zoch

Ford Aerospace & Communications Corporation
College Park, MD 20740

OVERVIEW

The Ford Lisp-Ada Connection (FLAC) is an expert system generation tool designed to support direct entry of knowledge by experts in a Lisp machine environment and downloading to an inference engine which has been implemented in the Ada programming language. FLAC consists of two subsystems, the Knowledge Editor Graphics System (KEGS) and the Ford Ada Inference Engine (FAIE).

Knowledge is entered through KEGS, an easily learned knowledge base CAD system which provides integrated features for rule development and knowledge base testing. An expert can use a set of menu- and mouse- driven resources to develop a knowledge base which is graphically represented. Tools are provided for the expert to rapidly enter, test, and debug knowledge base logic paths. The user interface is similar to those found in CAD systems for electrical circuit design.

The knowledge base can then be downloaded to FAIE, an extremely fast portable Ada-based inference engine which is capable of firing up to 700 rules per second on an MC68000 or 1500 rules per second on a VAX 11/780. The inference engine is written in the Ada programming language and supports both forward and backward chaining modes of inference. The FAIE run-time environment has been previously used in a prototype of the Space Station Operations Management System.

FLAC currently runs on a Symbolics 3640 and a VAX 11/780 VMS processor connected via DECnet/Ethernet. KEGS has been implemented in Symbolics ZetaLisp and FAIE was originally implemented in Telesoft Ada on an Intellimac MC68000-based system and then ported to DEC VAX/VMS Ada.

EXPERT SYSTEMS IN THE SPACE STATION INFORMATION SYSTEM

Future space information systems must be automated to the fullest extent possible both to support crew and operations staff efficiency and to allow us to maintain spacecraft and instrument safety in the face of increasingly complex systems and operational requirements. The Space Station Information System is a prime example of these enhanced requirements. One of the key engineering guidelines for the

Space Station is that it should be able to carry out normal operations for some finite period of time without contact with the ground. As pointed out in a NASA Technical Memorandum on Automation Technology For The Space Station,

> "Expert systems are needed to perform many monitoring and control functions requiring complex status analysis and automated decision making so that the Station is less dependent on ground support in these areas."

Also in the same document:

> "In emergency situations, automated systems which respond very rapidly to a crisis can bring the system to a fail-safe condition before extensive damage occurs... Without automation, humans may be placed more often in pressure-prone situations such as EVA and emergency maintenance in which there is an increased chance of error."

Expert systems could incorporate fault diagnosis, isolation, and recovery to enhance crew safety. Alarms could be triggered automatically to warn crew members of hazardous situations. In addition, many faults could be corrected before they pose any danger to the crew or spacecraft.

Lisp versus Ada

Ada has been baselined as the programming language for the configuration-managed software associated with the SSIS. A lively debate in the software community has centered on the respective strengths of Lisp and Ada as languages for the implementation of expert systems. Lisp is generally regarded as a "hacker's" language. Lisp and its development environments support highly interactive modes of software implementation where requirements specification, design, implementation, and testing are mixed in a process typically characterized as "prototyping". Ada stems from a heritage of disciplined development and configuration management practices of the sort usually practiced by both NASA and the Department of Defense. Neither mode is a totally satisfactory approach to the development of quality products which fully meet user needs. We cannot tolerate undisciplined development of mission critical systems, yet many of our systems are developed with an inflexible approach that does not fully meet user requirements.

At a more fundamental level Lisp and Ada differ as programming languages. Lisp is a weakly typed language with characteristics that even obscure the distinction between data and programs (both are trees of Lisp atoms). Ada on

the other hand requires compile-time checking of type compatibility and does not allow the passing of procedures as arguments. Lisp is commonly used as an interpreted language while the design of Ada (particularly the large volume of compile-time checking) strongly restricts it to compiled implementations.

Lisp and Ada do share certain common characteristics. Object-oriented design is a common theme and some of the best implementation work done in both languages uses object-oriented methodologies. Both languages support structures which facilitate the direct implementation of object oriented design. The Rational Ada development environment borrows many Lisp machine concepts to radically improve productivity of Ada programmers. The Common Lisp notion of a package provides Lisp programmers with modular capabilities similar to those found in the Ada language.

An Integrated Approach

Broadly speaking, Lisp environments are ideal for the prototyping and development of user interfaces. Ada environments are ideal for the development of large software systems with critical reliability and maintenance requirements. An approach currently being evolved at Ford Aerospace is the integrated use of both languages in a networked environment. Lisp is used to construct an extremely friendly knowledge editor which supports the development and testing of knowledge bases which are downloaded to an Ada inference engine that operates in real-time fully independently of the knowledge editor. This paper describes the features of the Ford Lisp-Ada Connection (FLAC), a prototype which combines both systems into a coherent real-time expert systems development environment.

THE FLAC ENVIRONMENT

Figure 1 shows the overall structure of FLAC. In its current implementation an expert uses a mouse-menu interface to enter knowledge through a graphics-oriented editor. He can exercise test cases and directly observe the behavior of the knowledge base which is graphically represented. He can trigger downloading of knowledge bases to a remote computer system and directly observe the execution of applications programs which use the real-time inference engine through remote terminal services. Windowing services on the Offline system allow simultaneous knowledge entry, debugging, and observation of remote applications. The current implementation environment of FLAC is a a Symbolics Lisp machine (Offline System) linked by DECnet/Ethernet to a VAX 11/780 (Online System). We describe the two major components of FLAC, the Ford Ada Inference Engine (FAIE) and the Knowledge Editor Graphics System (KEGS).

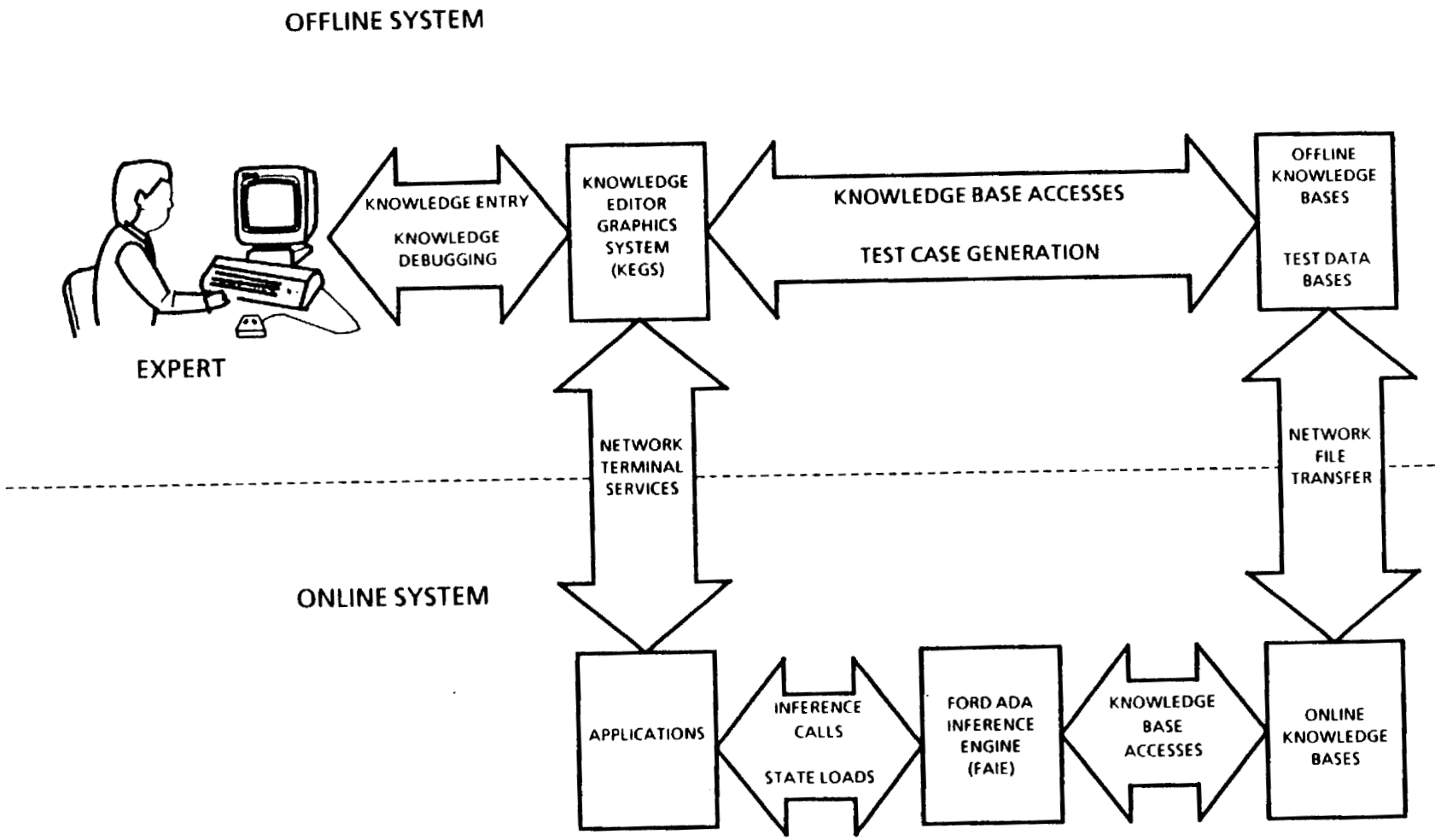


Figure 1. Ford Lisp - Ada Connection (FLAC)

The Ada Inference Engine

The Ford Ada Inference Engine (FAIE) is a prototype expert system inference engine designed to execute as an Ada task embedded in an expert system which could in turn be embedded in a larger program. The sample application discussed here involves using FAIE for fault diagnosis. A typical rule in this type of system might be:

> "IF temperature is above normal and
> heater output is above normal,
> THEN power off heater."

The knowledge base is structured as a directed acyclic graph. This can be thought of as a network of nodes with the links all pointing in the same direction. For the diagnostic system, the leaf nodes on one side of the graph represent the various sensor data measurements. Commands for corrective action are the goal nodes on the other side of the graph. The relationships between erroneous measurements are the intermediate nodes leading to a goal. Figure 2 shows a portion of a sample graph. Note: the dotted lines represent additional portions of the graph that are not shown.

The leaf nodes represent initial data points that must be provided to the inference engine. The nodes on the other side of the graph represent goal states that are sought when executing the inference engine. The nodes in between represent hypotheses or subgoals that will be tested. The links between the nodes are the "production rules" that the inference engine uses to traverse the graph.

Since we have a compiled, static knowledge base, all elements are present in the graph. Each node has a status which we will refer to as "flagged", "unflagged", or unknown. A "flagged" node is one that satisfies its associated IF-THEN rule. We must distinguish between an untested node (status equals unknown), and a node that was tested and does not satisfy the associated IF-THEN rule (status equals "unflagged"). A "flagged" node is one that will be used to traverse the graph. The path to a goal must be continuous through "flagged" nodes. An "unflagged" node represents a "dead end".

Status for all the leaf nodes is passed to the inference engine when a problem exists. Figure 3 shows the sample knowledge base with all the leaves (nodes 1-11) given an initial status. Nodes 2,3,10 and 11 are "flagged".

In an attempt to find a goal as quickly as possible, the successors of the first "flagged" leaf node are examined and the first one in the list is visited using Ada procedure FORWARD_CHAIN. Since the status of the successor node is initialized to unknown, its predecessors are examined along

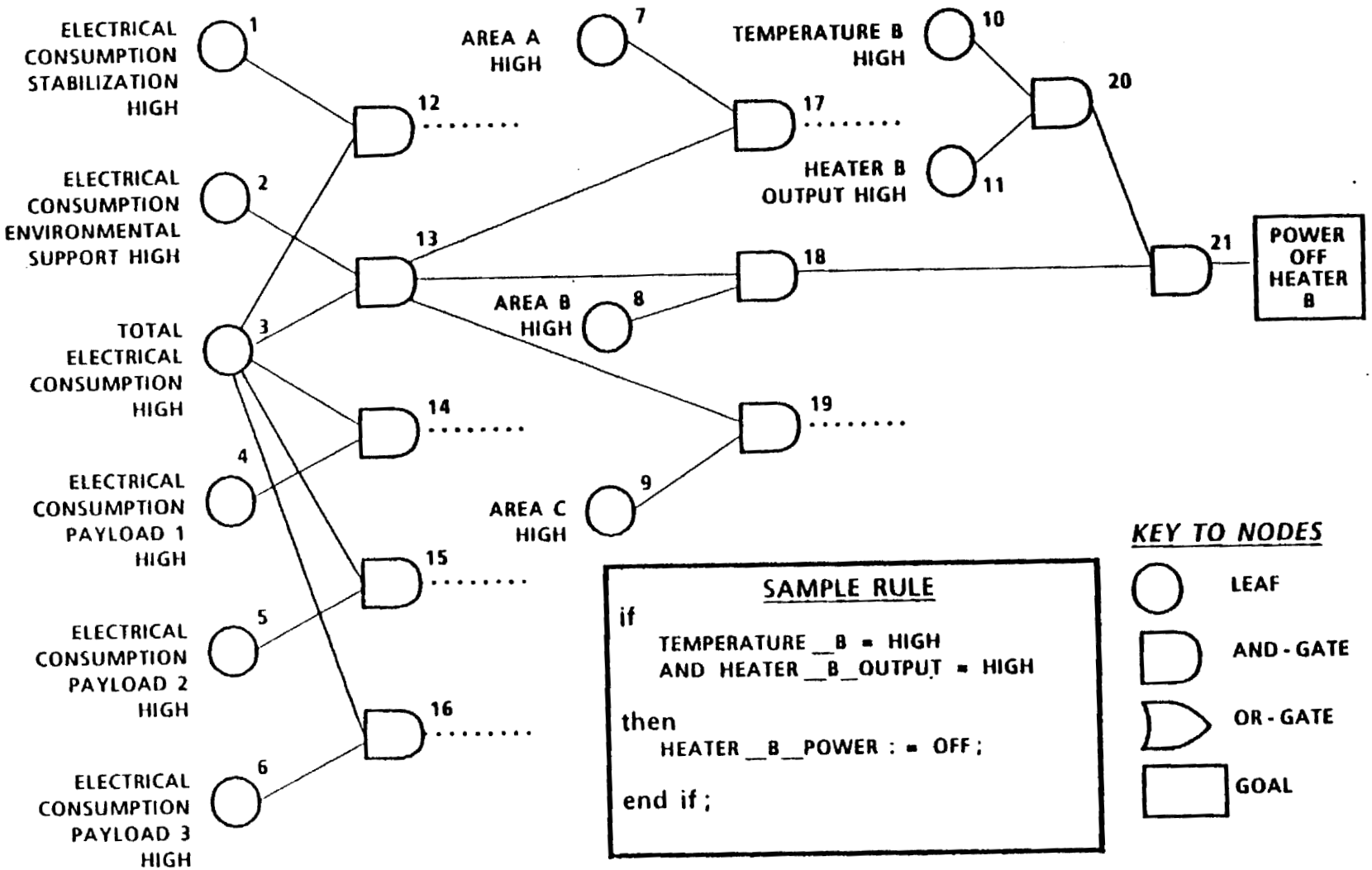


Figure 2. Sample Compiled Knowledge Base

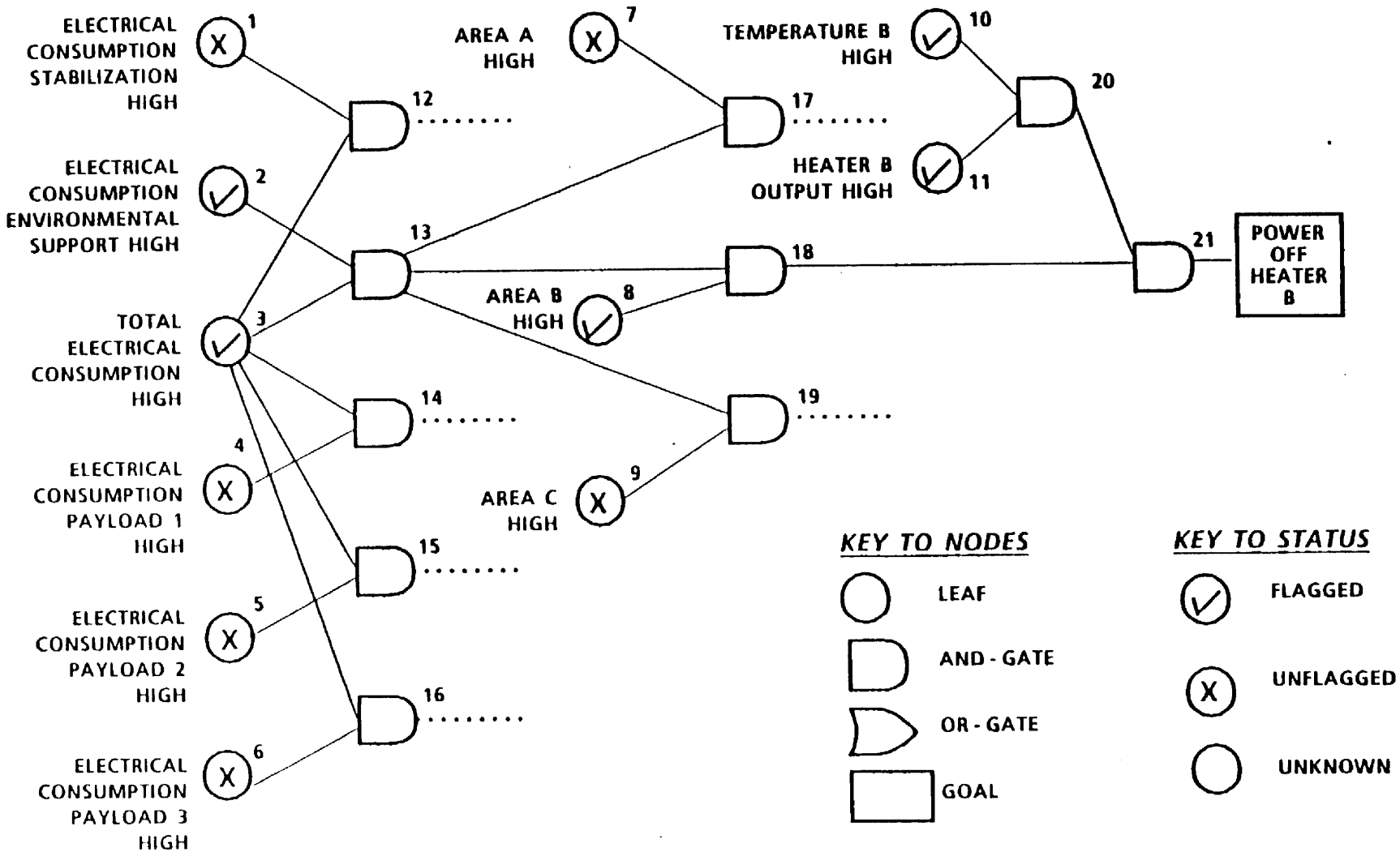


Figure 3. Knowledge Base - Initial Problem State

with its AND/OR flag to determine its status. If the status of this first successor to the first leaf node is found to be "flagged", then its first successor in its list is visited, and so on until a goal is found or a dead end is reached. If the status of this first successor is found to be "unflagged", then the next successor in the first leaf node's list is visited.

If the status of a predecessor node is unknown, then Ada function BACK_TRACK is invoked to return the status. Both subprograms FORWARD_CHAIN and BACK_TRACK are recursive.

Figure 4 shows the resulting status after running the inference engine. To get to Figure 4 from Figure 3 the following steps were taken:

1. Node 2's successor list is examined, and node 13 is passed in a call to FORWARD_CHAIN.
2. Since node 13 is an "and gate" and both its predecessors (2 and 3) are "flagged", node 13 becomes "flagged".
3. Node 13's successor list is examined, and node 17 is passed in a recursive call to FORWARD_CHAIN.
4. Since node 17 is an "and gate" and node 7 is "unflagged" node 17 becomes "unflagged".
5. FORWARD_CHAIN returns to visiting node 13, where the successor list is examined, and node 18 is passed in another recursive call to FORWARD_CHAIN.
6. Since node 18 is an "and gate" and both its predecessors (8 and 13) are "flagged", node 18 becomes "flagged".
7. Node 18's successor list is examined, and node 21 is passed in another recursive call to FORWARD_CHAIN.
8. Since the status of node 20 is unknown, node 20 is passed in a call to BACK_TRACK.
9. Since node 20 is an "and gate" and both its predecessors (10 and 11) are "flagged", node 20 is "flagged" and BACK_TRACK returns.
10. Since node 21 is an "and gate" and both its predecessors (18 and 20) are "flagged", node 21 is "flagged" and a goal has been found.
11. The recursive calls return and visit other successor nodes for additional goals.

In practice FAIE is capable of exceedingly fast performance. As implemented on a nonvalidated version of the Telesoft Ada compiler FAIE supported a rule firing rate of 700 rules per second on a Motorola MC68000 processor. On its current VAX 11/780 implementation rule-firing occurs at the rate of 1500 rules per second. For small expert systems this rate is more than adequate to achieve real-time performance. Moreover, the maximum search time for a goal can easily be computed from the characteristics of the knowledge tree. Future versions of FAIE which take advantage of multiprocessing implementations of Ada run-time systems will be even more powerful.

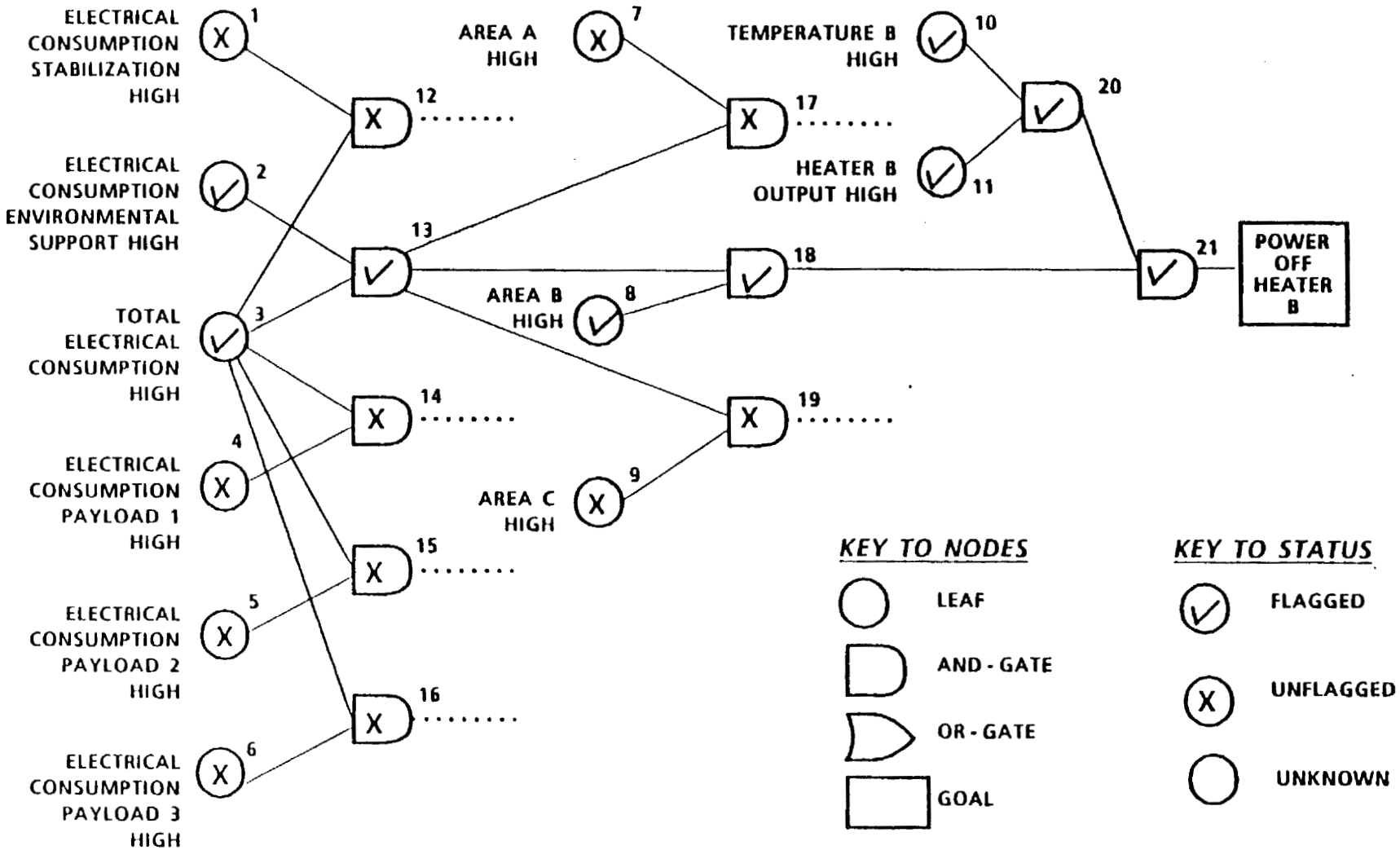


Figure 4. Knowledge Base - Problem Solution

The Knowledge Editor

The Knowledge Editor Graphics System (KEGS) is a simple CAD-oriented system for the direct entry of knowledge by experts. Knowledge entry is accomplished by using a mouse to draw logic gates, leaves, and goals. Figure 5 is a sample screen from a KEGS session. It can be seen that the screen representation of the knowledge base is nearly identical to the internal representation of the knowledge base. This greatly simplifies design and eliminates need for time-consuming conversions of knowledge base formats. Moreover, since the representation is identical to that used in electrical circuit design it is nonintimidating to a large class of experts, the hardware engineers who design spacecraft systems.

Operation of KEGS is intuitive. The expert simply touches the cursor to the symbol to be drawn and places as many as required on appropriate screen locations. Connections are drawn by touching the line icon and then touching pairs of symbols in sequence. To delete an icon or connection the expert touches the cursor to the circle with the diagonal line through it and then touches the symbol to be deleted. Connections to that symbol are then automatically deleted. Ada functions to test leaf conditions or execute goal conditions are attached through a simple fill-in-the-blanks menu. These functions can be either user-written functions or selected from a library of system-provided functions (e.g. test a variable for range, send a command). Current capabilities to test the data base are limited to manually marking the leaf nodes and observing goal firing. Future plans call for inclusion of automatic test and tracing functions. Once the data base is complete and tested, the expert system knowledge base can be downloaded through the network to be executed by the runtime environment.

RESEARCH DIRECTIONS

The primary focus of our work is the use of multitasking to improve performance. This will also solve the problem of reading dynamic data which is constantly being updated as inferencing is in progress. It seems reasonable to use Ada tasking to enhance the real-time performance of inference engines. Although true production-quality multiprocessing Ada compilers do not yet exist, it is now feasible to write tasking implementations of inference engines which will exhibit order-of-magnitude improvements in rule-firing rates when ported to true multiprocessing Ada environments.

Douglass lists five levels of potential parallelism in rule-based expert systems. They are: subrule level, rule level, search level, language level, and system level. These levels include different types within them. Douglass concentrates on rule level and various types of search level parallelism. He gives a range of quantitative results for

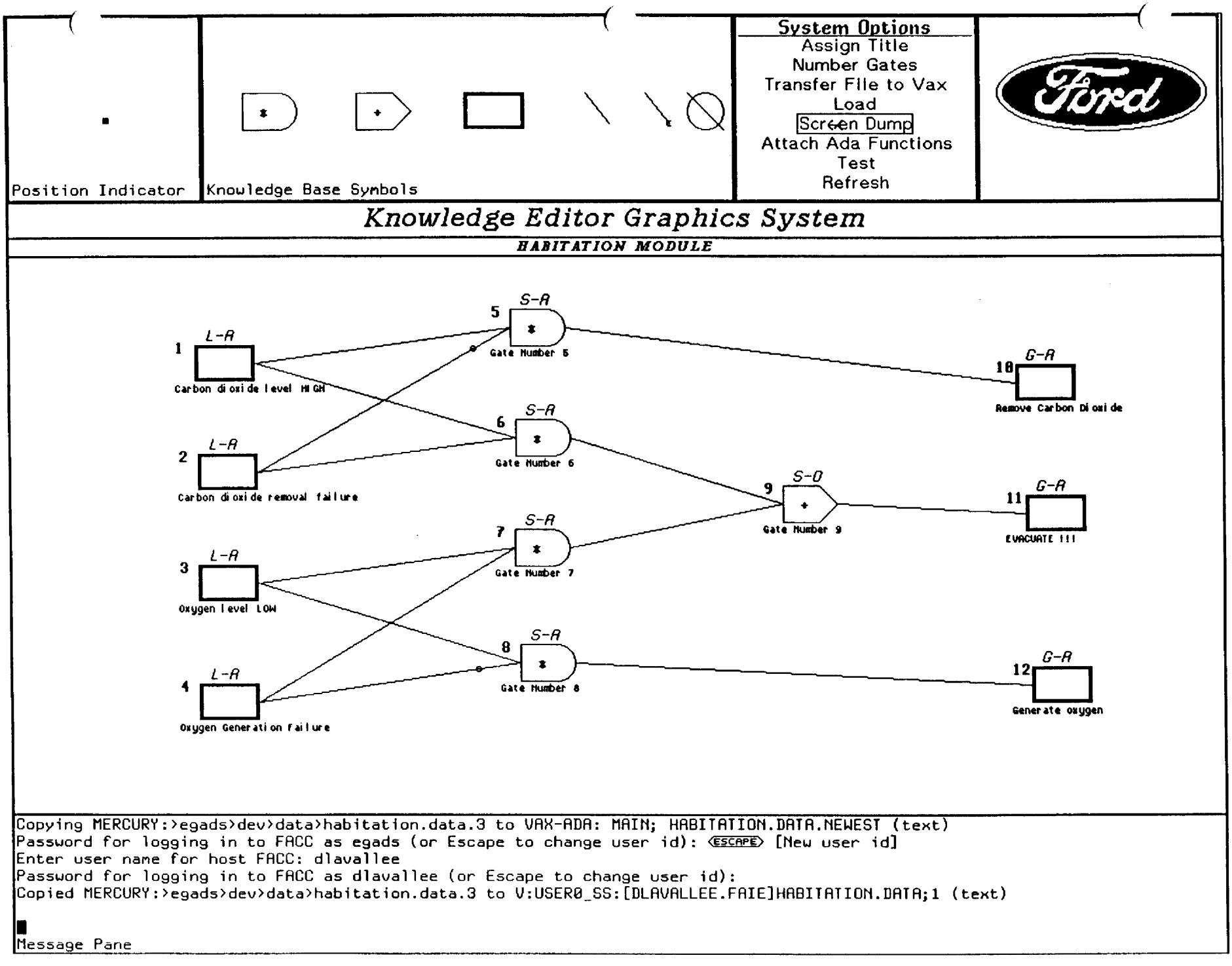


Figure 5. Sample KEGS Screen

these levels using mathematical models and concludes that combinations of subrule, rule and search level parallelism will yield better results than any single level when the characteristics of the specific system are taken into consideration. He also mentions that very little work has been implemented and tested on parallel computers.

Communication between processes is an important factor in the efficiency of parallel algorithms. Generally speaking, the more frequently that information is exchanged, the slower the computation is performed since processes spend a larger portion of their time communicating rather than computing. Researchers working on the DADO machine have developed some unique methods of communicating between parallel processors (e.g. a binary tree structure of processors with communication rules controlled by hierarchy).

In Ada, the task is the natural construct for parallel processing. However, multitasking involves considerable overhead in creating/activating tasks, communicating between them, and terminating them. This overhead must be compared with the amount of computation performed in parallel in order to determine the relative efficiency gained by various strategies of parallel processing. Gehani concurs, and goes on to say that in designing concurrent programs in Ada, one must avoid the polling bias in the communication mechanism. He also points out that multiprocessing programs will be more efficient if the underlying hardware offers genuine concurrency.

Deering also emphasizes that hardware considerations, especially processor speeds versus memory speeds, must be examined when designing the architecture of expert systems. He says one should "study hardware technology to determine at what grain sizes parallelism is feasible and then figure out how to make [the] compilers decompose programs into the appropriate-size pieces."

Granularity is the average amount of work done by a process between communication with other processes. It is inversely proportional to the frequency of communication. The five levels of parallelism mentioned by Douglass range from very finely grained to roughly grained. A fine grained approach was taken by Rude where each rule was itself declared as an Ada task with rendezvous for links to predecessors and successors. This concept has merit but is questionable for real-time applications. In the implementation of the PICON expert system for real-time process control, a roughly grained algorithm was chosen by segmenting parts of the knowledge base and applying priorities to searching the different portions. Our future investigations will include analyzing various strategies, including forward and backward chaining on individual rules in parallel, dividing the knowledge base, and combinations of the different

strategies.

For the knowledge editor we expect to investigate modes of enhancing the overall debugging interface, including the development of tools for automated testing which allow the expert to explore the more critical logic paths of the system, particularly those logic paths which might lead to actions or recommendations which affect overall spacecraft safety.

A significant problem which we have encountered in the KEGS implementation is the binding of Ada procedures and functions to leaves and goals. Since Ada does not provide the capability to dynamically define new procedures or functions and pass them as procedure parameters we have been forced to limit the expert to the use of a previously defined library of procedures and functions (e.g. test to see if a variable is in range, send a command string). A more flexible approach would give the expert access to the Ada program library in order to implement procedures and functions when necessary. We are currently investigating the alternatives and implications for implementing these bindings.

Conclusions

The prototype demonstrates the feasibility of using Ada for expert systems and the implementation of an expert-friendly interface which supports knowledge entry. In the FLAC system Lisp and Ada are used in ways which complement their respective capabilities. Future investigation will concentrate on the enhancement of the expert knowledge entry/debugging interface and on the issues associated with multitasking and real-time expert systems implementation in Ada.

BIBLIOGRAPHY

David C. Brauer, Patrick P. Roach, Michael S. Frank, and Richard P. Knackstedt, "Ada and Knowledge-Based Systems: A Prototype Combining the Best of Both Worlds", Expert Systems in Government Conference, McLean, VA, October 1986

NASA Advanced Technical Advisory Committee, Advancing Automation and Robotics Technology for the NASA Space Station and for the U.S. Economy, NASA Technical Memorandum 87566/, Volume II, March 1985 p. 5.

Deering, M. "Architectures for AI", Byte Magazine, April, 1985.

Douglass, R., "Characterizing the Parallelism in Rule-Based Expert Systems", Proc. Hawaii International Conference on Systems Science, HICSS-18, Jan. 1985.

Douglass, R., "A Qualitative Assessment of Parallelism in Expert Systems", IEEE Software, May 1985, pp. 70-81.

Gehani, N., Ada: Concurrent Programming, Prentice-Hall Inc., 1984.

David B. Lavallee, "An Ada Inference Engine for Expert Systems", Proceedings of the First International Conference on Ada Programming Language Applications for the NASA Space Station, Houston, TX, June 1986.

Moore, R., L. Hawkinson, C. Knickerbocker, L. Churchman, "A Real-Time Expert System for Process Control", First Conference on AI Applications, IEEE Computer Society Press, Dec. 1984.

Moore, R., "Adding Real-Time Expert System Capabilities to Large Distributed Control Systems", Control Engineering, April 1985.

Rude, A., "Translating a Research LISP Prototype to a Formal Ada Design Prototype", Proceedings of the Washington Ada Symposium, March 1985.

Stolfo, S., and D. Miranker, "DADO: A Parallel Processor for Expert Systems", Proceedings of the 1984 International Conference on Parallel Processing, IEEE Computer Society Press, August, 1984.

Stolfo, S., "Five Parallel Algorithms for Production System Execution on the DADO Machine", Proceedings of the NCAI, Austin, TX, 1984.

Expert Systems Built By The "Expert": An Evaluation of OPS5

May 13, 1987

Robert Jackson [1]
Astronomy Programs, Computer Sciences Corporation
Space Telescope Science Institute [2]
3700 San Martin Drive
Baltimore, MD 21218

Abstract

Two "expert systems" have been written in OPS5 by the "expert", a Ph.D. Astronomer with no prior experience in AI or Expert Systems, without the use of a "knowledge engineer". The first system was built from scratch and and uses 146 rules to check for duplication of scientific information within a pool of prospective observations. The second system was grafted onto another expert system and uses 149 additional rules to estimate the spacecraft and ground resources consumed by a set of prospective observations. The small vocabulary, the IF *this occurs* THEN *do that* logical structure of OPS5, and ability to follow program execution allowed the "expert" to design and implement these systems with only the data structures and rules of another OPS5 system as an example. The modularity of the rules in OPS5 allowed the second system to modify the rulebase of the system onto which it was grafted without changing the code or the operation of that system. These experiences show that "experts" are able to develop their own "expert systems" due to the ease of programming and code reusability in OPS5.

[1] Staff Member of the Space Telescope Science Institute

[2] Operated by the Association of Universities for Research in Astronomy for the National Aeronautics and Space Administration

1 Introduction

This paper describes the experiences of a "computer semi-illiterate domain expert" in writing two expert systems in OPS5 without the use of a Knowledge Engineer. The two systems, Duplication and Resource Usage, are described in some detail along with some of the strengths and limitations of the OPS5 language and environment.

2 Personnel Background

In order to put these "expert" systems into perspective it is useful to describe the people who created them.

The author's software experience, prior to beginning these projects, consisted of rather simple data analysis routines written in FOCAL running on a PDP 8/I and some initial exposure to the IQL database query language for a Britton-Lee IDM 500. There was no previous contact with modern programming languages, structured programming, or artificial intelligence.

The author did, however, work in the same group as a Computer Scientist who was writing a large system using OPS5 and who currently teaches OPS5 workshops. Aside from initial "hand-holding", this OPS5 expert's advice was used infrequently.

3 Scientific Duplication

The Space Telescope Science Institute is responsible for the scientific operations of the Hubble Space Telescope (HST). When launched by the Space Shuttle, the HST will allow astronomers to detect objects seven times more distant and with ten times higher resolution than from ground based telescopes. Astronomers who wish to use HST submit proposals describing the observations they wish to perform. Because of the unique capabilities of HST, much more observing time will be requested than is available. A Time Allocation Committee selects which of the proposals will be granted time on HST.

In this proposal selection process, the proposed exposures will be checked to see if any of them duplicate exposures already in the HST archives, exposures proposed by other scientists, or exposures which are "reserved" by the Guaranteed Time Observers (GTO's), the scientists who developed the HST scientific instruments. This duplication checking is intended to identify observations which waste spacecraft time or which violate the perogatives of the GTO's.

There will be about 40,000 exposures executed in each year of HST operations. With an expected oversubscription factor of about three for the observing time requested versus the time available, the duplication checking process clearly had to be automated. The author was charged with automating that process.

3.1 Specifics of the Problem

A single observation with HST contains information with the following characteristics:

- Field of View
- Spatial Resolution
- Wavelength Range
- Wavelength Resolution
- Minimum Detectable Intensity
- Time Duration
- Time Resolution

Two observations could be considered to be duplicates to the extent which the information obtained has the same values of these characteristics. Obviously, if two exposures use identical target positions, instruments, instrument settings, and exposure times, then the same data would be obtained. The complication comes from the fact that different instrument settings and even different instruments can produce similar, though not identical, data. The Duplication checker should be able to identify cases where similar data is being obtained.

The author in the capacity of "Domain expert" analyzed the instruments and the instrument settings and created a number of criteria for what constitutes a "HIGH", "MEDIUM", and "LOW" confidence duplication. The instruments and instrument settings are sufficiently different that several different criteria can produce the same confidence duplication. For example a "LOW" confidence duplication can be produced by both:

- Identical instruments and instrument settings, but drastically different exposures times.
- Completely different instruments which have similar spectral and spatial range and resolution.

The specific exposure data available for the duplication checking are:

- Target Position - Center of Field of View - 2 Coordinates
- Target Position Uncertainty - 2 Coordinates
- Configuration - The optical path, set mechanically or electronically.
- Mode - Spatial or Time parameters of the instrument
- Spectral Elements - Wavelength Range and Resolution
- Aperture - Field of View or Spatial Resolution
- Central Wavelength - Spectral Range
- Exposure Time - Minimum Detectable Intensity

From this data, for all the proposed exposures and all the past exposures, the three types of duplicate expsoures have to be identified.

3.2 Overall Approach

The problem can be decomposed into a spatial duplication and a instrumental duplication. There is no point in checking if two targets have duplicate exposures if they are 180 degrees apart on the sky. Note that this ignores the possibility that scientifically equivalent information can be obtained from different targets which are members of the same class of objects. Identifying objects as being members of the same class would require an AI system combining data from virtually all the available astronomical catalogs and literature and would be a major project in its own right. Using such an AI system would ignore the possibility that differences between objects in the same class is the very subject of the investigation.

The separation between spatial and instrumental duplication is not as clean as it might appear. Spatial duplication can depend on the spatial range of the instrument being used. For a camera with a very wide field of view, a given target could appear in two exposures at rather different positions. Additionally where the stated uncertainty in the target position may not be a good measure of the true errors in the target position, a very flexible measure of spatial duplication is needed.

Based on these considerations, the approach finally adopted consisted of:

1. Initial Search - Find groups of targets close together on the sky.
2. Instrument Matching - Find pairs of exposures in each group with "HIGH", "MEDIUM", and "LOW" confidence instrument matches.
3. Fine Position Check - Find pairs of exposures which are either
    - At statistically indistinguishable target positions.
    - Both within the instrument's spatial range.
    - Both within a user specified distance of each other.

The "Initial Search" drastically reduces the search space of the problem. It finds groups of targets that are closer than .1 degrees of each other, a distance which is much larger than the usual "Fine Position Check" distances of about 10 seconds of arc. The larger "Initial Search" distance thus requires the "Instrument Matching" be done on exposures which may not meet the "Fine Position Check". But by performing "Fine Position Check" after the "Instrument Matching", the criteria used in "Fine Position Checking" can be selected by user consistent with the user's estimate of the reliability of the target positions and uncertainties. The flexibility of the user specification of the "Fine Position Check" was deemed to be more important than the time wasted on finding irrelevant "Instrument Matching" duplications.

The other intentional limitation built into the duplication checking algorithms is fact that time variations of the target's properties are ignored. If more than one identical observation of a target is made, all the subsequent observations are ignored. While spurious duplications of time varying targets may be produced, the number of duplications found will be greatly reduced.

3.3 The Initial OPS5 Implementation

OPS5 seemed to be the appropriate tool to implement the "Instrument Matching" part of the duplication checker. The conditions for each degree of duplication can be described in English sentences and are easily expressed into a small number of OPS5 rules. Additionally, an OPS5 program could work on all the exposures in a group of targets in "parallel" without the need for looping mechanisms, thus simplifying the coding.

The computer scientist familiar with OPS5 gave the author the VAX OPS5 User's Guide [1], VAX OPS5 Reference Manual [2], and some examples of his OPS5 code. From these documents and the code samples, the general idea of data driven program flow and the **IF** *This Data Exists* **THEN** *Do These Actions* nature of OPS5 rules became apparent. The code samples illustrated the use of comments and white space and the use of English words and phrases to name rules, data structure fields, and variables.

The non-procedural nature of OPS5 was *not* a serious impediment for this particular novice. The notion of data-driven program flow is easily comprehended and implemented. The non-procedural aspect of OPS5 is actually an advantage where a function is to operate on all the available data, e.g., no DO LOOP control is needed. The OPS5 rules simply perform the operation on all the available data.

With this initial knowledge, the author started writing rules. After writing a single file with 4300 lines of OPS5 code, the author wandered over to the Computer Scientist and asked "Am I supposed to do something with this file?" Notions such as small modules and even compilation were still new to the author.

Upon compiling the file, there were only four syntax errors. The small syntax and the uniform structure of the rules allowed a neophyte to write code largely by cutting and pasting with the editor and putting different words in the same slots.

This first version relied almost solely on the data-driven notion of how OPS5 works. Not much reliance was placed on recency (testing for the most recent data) or specificity (testing for the more descriptive rule) as a way to control program execution. Instead, most of the rules tested the value of a control element which was set when the prior function has been completed. Equivalently, there was no use of the MEA strategy (placing extra weight on the first clause in a rule) and controlling program flow by chaining from one goal to another.

The OPS5 Duplication checking program performs the following operations on the data:

- Input the data from one group of exposures.
- Rename certain spectral elements and apertures to eliminate redundant names.
- Renormalize HRS Echelle central wavelength to aid in calculating the spectral range for this spectral element.
- Eliminate exposures which repeat identical exposures.
- Find "High" Instrument matches - identical instrument parameters.
- Find "Medium" Instrument matches - small differences in instrument parameters.

- Remove HRS Echelle matches with non-overlapping spectral ranges.
- Downgrade "High" or "Medium" duplications to "Medium" or "Low" if the exposure time ratios are too large.
- Find different type of "Low" Instrument matches.
- Calculate target separation for each Instrument match.
- Perform Fine Position Check for statistically indistinguishable target positions.
- Perform Fine Position Check for both targets in entrance aperture.
- Read out duplicate exposure data.

3.4 Subsequent Development

Since these humble beginnings, the code has been changed to:

- Use different criteria to determine the various degrees of duplication. The users of this Duplication checker preferred a somewhat different set of criteria than the author originally created.
- Use the MEA strategy, which gives extra weight to the first clause of the rule, with goal chaining for program control. This control method replaced the use of control flags and allowed much easier modification of the rulebase. It was no longer necessary to check which flag had been set, where it had been set, etc., when making changes to the rules or adding new rules.
- Be contained in several small files of related rules instead of one large file. Using several small files allowed more rapid compilation of the rule changes and provided a more obvious organization of the rules for subsequent maintainers.
- Replace several similar rules with one rule reading from a data table. There were many rules where the only difference was the value of certain constants in the *IF* clauses. These rules were replaced by one rule which went to a lookup table for the values of the constants. This compression of the rulebase made the code more compact, more easily modified, and more readable.
- Perform the third, "Fine Position Check", part of the duplication checking. This feature was required by the users and was easily added to the existing OPS5 rulebase.

In the debugging of this code, the following interpreter commands were extensively used:

- **@***filename* executed a file of OPS5 commands and was very useful for inputting test data.
- **RUN** *n* executed *n* rule firings and allowed single stepping through the program execution.

- **BACK** *n* stepped back *n* rule firings and allowed reinvestigating the internal state after one or more rule firings.
- **NEXT** displayed the name of the next rule set to fire.
- **CS** displayed the names of all the rules able to fire and ID number of the data which enabled them.
- **PBREAK** *rulename* halted rule firing before *rulename* fired and allowed quickly running the program to the point before a certain rule was to fire.
- **MATCHES** *rulename* displayed the ID number of the data which satisfied each clause of *rulename* and was very useful in finding why a certain rule did not fire.
- **WATCH 3** displayed all changes to the data and the rules which were enabled as each rule fired.
- **PPWM** *data description* displayed all data which met the *data description* and allowed searching the internal data.
- **SAVESTATE** *filename* saved the internal data state and the rule firing state at a point in time to *filename* and allowed restarting the program at a certain point without having to start from scratch.
- **RESTORESTATE** *filename* restored the internal data state and rule firing state to a certain condition.

3.5 Duplication Status

The duplication checker currently uses 146 rules and it is easy to modify the conditions defining the different levels of duplication. To search 52,000 exposures requires about six hours of CPU time on a VAX 8600. Extrapolating this performance to larger sets of data is difficult. Tests on subsets of the 52,000 exposures indicate that the CPU time increases more slowly than the number of duplications found and the number of exposures checked. In the mature phase of HST operations, the number of exposure which have to be checked will be so large that the duplication rulebase will have to be tuned to reduce the execution time. For now, the duplication check is performed only once each year and the execution time is acceptable.

4 Resource Usage

The committee which advises the Director of the Space Telescope Science Institute on which proposals to select is constrained by limits on the available spacecraft resources, i.e. time, data capacity, earth shadow time, etc. They need to know how much of these limited resources each proposal's observations consumes and how much of the available resources will be consumed by all the accepted proposals.

In the scientists' proposal for HST time, they specify the target positions, exposure times, instruments, instrument settings, absolute times, relative times, etc. of the observations.

The scientists do not specify, nor can they specify, the overhead times for spacecraft slewing, data readout, and internal mechanism changes or the time spent with the target occulted by the earth. The overhead times for spacecraft slewing and earth occultation are largely determined by how the planning and scheduling system combines the proposed exposures into the following hierarchy.

- Exposures → Alignments. An Alignment is a pointing of the spacecraft at a fixed position on the sky.
- Alignments → Obsets. An Obset is a collection of alignments done sequentially and which are sufficiently close together on the sky that the same pair of Guide Stars can be used.
- Obsets → Scheduling Units. A Scheduling Unit is a collection of obsets which must be done in a specific time sequence and is treated as a single unit by the scheduling software.

The model for the consumption of spacecraft time is that each obset will consist of:

- A slew from the previous obset's target position
- A guide star acquisition
- A series of alignments, each with a:
  - Exposure time
  - Data Readout time
  - Mechanism change time
  - A possible small angle maneuver time to get to the next alignment
- A number of occultation times and guide star reacquisition times which is a function of the total duration of all the alignments in the obset

Thus, if one knew how the planning and scheduling system combined exposures into alignments and obsets and knew the data readout and mechanism change times and small angle maneuver times, one could estimate the time spent in slewing or in occultation. With all this information, the total spacecraft time required by a scientist's proposal could be estimated.

Fortunately a system (called the *Transformation system* [3]) was being developed when this Resource Usage tool was being designed which:

- ordered the proposer's exposures into the Alignment-Obset-Scheduling unit hierarchy,
- determined the data readout and mechanism change times

This Transformation system is used to feed the HST Planning and Scheduling System.

Thus it seemed possible to use the Transformation system as the frontend for a Resource Usage calculator. The Resource Usage calculator would take the results of the Transformation system and then estimate the slew times, guide star acquisition times, small angle maneuver times, earth occultation times, and guide star reacquisition times. These times would be combined with the individual alignment times determined by the Transformation system to estimate the total spacecraft time required to execute the proposer's exposures.

Since the Transformation system also computes the data volume generated by the exposures and has the proposer's description of the observations in its database, the combined Transformation and Resource Usage systems could estimate the following constrained spacecraft resources:

- Total spacecraft time
- Data volume
- Spacecraft time which must be spent on the dark side of the earth
- Parallel observation time used
- Realtime uplinks required

as well as other quantities which the committee wishes to monitor.

By using the Transformation system as a frontend, the Resource Usage calculator would always use the same assumptions for how exposures were combined into alignments, obsets, and scheduling units; for the data readout times and mechanism change times; and for the data volumes as did the Planning and Scheduling system. For a single proposal considered in isolation, there would be no better way of estimating spacecraft resources.

4.1 Reusing The Transformation Rulebase

This Transformation system is written in OPS5, a language which the author was familiar after building the Duplication system. The program flow is controlled by a series of goals with the MEA strategy which gives extra importance to the first clause in the rule, usually the goal clause. The data structures, external declarations, rules for goal chaining, and the rules for each goal are all contained in separate small files.

The modular nature of the Transformation system made it apparent that it would be easy to graft a set of Resource Usage rules onto the existing Transformation system. Only two changes had to made to the internals of the Transformation system to use it as a frontend for the Resource Usage tool. The file containing the rules which chained from one goal to another was divided into two files, one containing the goals used by both Transformation and Resource Usage and one containing the goals used by Transformation alone in generating its output data files. The file containing the common goals would be combined with the goal chain used by Resource Usage to control the flow of the Resource Usage calculation. An additional data structure was added, with the necessary rule changes, to do a separate accounting of parallel exposures, i.e., exposures performed with a different instrument at

the same time as the primary exposure. Previously there had been no need to separately track parallel exposures.

These Transformation files would be compiled with the separate set of Resource Usage files to create the Resource Usage executable. These Resource Usage files contained the:

- Data structures.
- Goal chaining rules.
- External function declarations.
- Rules creating all the constants and lookup tables used.
- Rules which change data types to meet the assumptions of later rules.
- Rules summing exposure level resources into alignment level resources (mainly the data volume) and determining which exposures have narrow scheduling windows.
- Rules finding alignment level resource quantities, mainly the small angle maneuver times between alignments.
- Rules summing alignment level resources into obset level resources.
- Rules finding obset level quantities, mainly the occultation time and total spacecraft time.
- Rules summing obset level resources into proposal level resources.
- Rules finding maximum, minimum, and average values of proposal level resources for the different possible combinations of obsets.
- Rules reformatting the resource quantities into output formats.
- Rules writing the resources to the database and any applicable warnings to files.

Unlike the Duplication system, but yet like the Transformation system, Resource Usage made extensive use of external function calls to read from and write to the database, change datatypes, and perform mathematical calculations.

When the Transformation system was designed, using it to generate resource usage information was not a consideration of the design. This reuse of the Transformation system is mainly the result of the non-procedural nature of an OPS5 program, of the modular structure of the OPS5 files, and of the similarilarity of the data structures needed.

4.2 Isolated Modification of the Transformation Rulebase

With more familiarity with the Transformation rules, it became apparent that the different purpose of Transformation was embedded in some of the rules used both by Transformation and Resource Usage. The Transformation system was designed to operate on only exposures from one year in a proposal which may contain exposures for multiple years. The Resource Usage system was designed to estimate the resources for all years' exposures in a

proposal. Thus for Resource Usage to use the Transformation rulebase, it had to prevent Transformation from combining exposures from different years into an alignment or obset.

Preventing cross-year exposure combinations was accomplished by adding to one of the Resource Usage files two rules which in their first clauses referred to a goal which is in the part of Transformation rules used by both systems.

```
(p remove-different-cycle-exposure-links
    (goal
         ^has-name                    merge-exposures
         ^has-status                  active
         ^task-list                   find-potential-exposure-merges)
   {<mergeable-exposure-link>
    (mergeable-exposures
         ^first-exposure-number       <second-exposure>
         ^second-exposure-number      <first-exposure> ) }
    (exposure-properties
         ^has-exposure-number         <second-exposure>
         ^has-scheduling-cycle-name   <cycle-name> )
    (exposure-properties
         ^has-exposure-number         <first-exposure>
         ^has-scheduling-cycle-name   <> <cycle-name> )
-->
    (remove <mergeable-exposure-link>))

(p remove-different-cycle-alignment-links
    (goal
         ^has-name                    merge-alignments
         ^has-status                  active
         ^task-list                   find-potential-alignment-merges)
   {<link-to-remove>
    (mergeable-alignments
         ^has-first-alignment-order   <first-alignment-order>
         ^has-second-alignment-order  <second-alignment-order>
    (assignment-record
         ^has-Pepsi-exposure-number   <first-exposure>
         ^has-alignment-order         <first-alignment-order>)
    (assignment-record
         ^has-Pepsi-exposure-number   <second-exposure>
         ^has-alignment-order         { <second-alignment-order> <>
                                            <first-alignment-order> } )
    (exposure-properties
         ^has-exposure-number         <second-exposure>
         ^has-scheduling-cycle-name   <cycle-name> )
    (exposure-properties
         ^has-exposure-number         <first-exposure>
         ^has-scheduling-cycle-name   <> <cycle-name> )
-->
```

```
(remove <link-to-remove>) )
```

Since these rules are in a file which is not compiled when creating the Transformation executable, these rules have no effect on the Transformation system's operation. Because these rules are active during a Transformation portion of the goal chain, the rules are modifying the operation of the Transformation frontend to the Resource Usage system.

The non-procedural nature of OPS5 allows one user of a set of rules to alter the set's operation without affecting other users of that set of rules.

4.3 Resource Usage Status

The current Resource Usage tool uses 149 rules unique to it and 327 rules from the Transformation system. Resource Usage and Transformation are both limited at present in the number of exposures they can process. For proposals with more than about 800 exposures, the virtual memory required exceeds the 90,000 page virtual memory limit of the account used. However, these 800 exposures contain more than 6000 separate data elements for the OPS5 rules to match on - thus the large memory usage. For the current pool of GTO proposals, about 10% of the proposals have more than 800 exposures.

The speed of the Resource Usage tool is more important than for the Duplication tool. A proposal may be accepted with the condition that it be reduced to meet certain resource limits. This paring down and verifying that the resource limits are met will probably be an iterative process. Thus the Resource Usage tool may be run several times on a given proposal. It presently requires about two minutes of CPU time to run the Resource Usage tool on the average proposal and about thirty minutes of CPU time for the largest proposals which do not exceed the memory limits. These execution times are acceptable in verifying that proposals are meeting the set resource limits.

The large memory usage appears to be the result of very large numbers of partial instantiations of rules, i.e., the first few clauses in rules are satisfied by many data. The memory usage can be reduced by reordering both the clauses and the order of the elements within the clauses. Reordering some of the clauses has already reduced the memory usage by a factor of two, but this is not sufficient to process the largest proposals. The permanent solution to the memory usage problem will probably involve identifying disjoint subsets of the exposures, and operating on one subset at a time.

5 Strengths and Weaknesses of OPS5

The greatest strength of OPS5 is in its simple syntax, which allows a domain expert to learn quickly how to write rules in OPS5. The concept of data driven computation and the use of the MEA strategy with goal chaining provides the expert with a simple method of controlling the execution of the OPS5 program. With initial help and guidance from a person familiar with OPS5, domain experts can create their own expert systems.

As the expert system is created by the domain expert, rather than the Knowledge Engineer, the domain knowledge goes directly from the expert to the rules without going through the

"filter" of the Knowledge Engineer. This should decrease the chance of a misunderstanding being coded into the expert system. When the domain expert creates *AND DEBUGS* the expert system, there is a much greater chance that the system will do what the expert thinks it should.

Reducing the reliance on the Knowledge Engineer also lessens the bottleneck caused by the small number of Knowledge Engineer relative to the large number of "Experts".

Another advantage of OPS5 is the ability to reuse and modify existing OPS5 rules without affecting the other users of those rules. Due to the non-procedural nature of an OPS5 program, functionality can be added or changed by simply incorporating additional rules at compile time with any editing of the original code. This requires only that the rules are kept in small modular files. A core expert system can be written and then modified to suit the needs of other users. With a procedural language, making isolated modifications to common code is much more difficult. To quote the Computer Scientist who first introduced the author to OPS5, "Who says you can't write reusable code?"

The biggest disadvantage of OPS5 is the environment. One cannot examine the text of the rules while single stepping through the rules with the command interpreter. Thus for debugging, either a paper copy of the rulebase is needed or a multi-window hardware system is needed. An environment where one can view the rulebase, change rules, recompile, relink, and run the command interpreter would be a great aid to developing OPS5 expert systems.

The other major disadvantage of OPS5 is the large memory usage. There are ways to reduce the memory requirements by changing the code, and the availability of computer memory is always increasing. However, large memory usage is by its nature data dependent and does not constitute a fundamental limitation of the language.

References

[1] *VAX OPS5 User's Guide*, 1985, Digital Equipment Company

[2] *VAX OPS5 Reference Manual*, 1985, Digital Equipment Company

[3] Rosenthal, D., Monger, P., Miller, G., Johnston, M. 1986, *Proceedings of the 1986 Conference on Artificial Intelligence Applications*, NASA Goddard Space Flight Center

ROBOTICS

SPACE TRUSS ASSEMBLY USING TELEOPERATED MANIPULATORS

By

Walter W. Hankins III, Randolph W. Mixon, and Howard C. Jones
NASA Langley Research Center
Thomas W. Burgess
Oak Ridge National Laboratory

INTRODUCTION

The level of man's activity in space and his utilization of it is increasing at a very high rate with an accompanying accelerating requirement for more and more astronaut EVA (Extra Vehicular Activity) to deploy, repair, service, and resupply orbiting facilities. It is likely that "people" will construct the Space Station from components carried into space on the shuttle and/or expendable vehicles, thus demanding even more of EVA astronauts. Human EVA is dangerous and inefficient; a better way of getting this work done is needed. A possible solution promoted recently by NASA research is to use automated and teleoperated machines, but these have many unresolved problems. Automated devices operate extremely well on earth-based assembly lines where they very precisely perform well defined, preprogrammed, repetitive tasks; but they do not perform so well when the environment is less structured and the required activity is impossible to timeline in detail, a priori.

Teleoperation, of course, having direct human control, is not so dependent on structured environments and precise timeline knowledge of the task, but will require a high level of manipulator dexterity and controlability for realistic space tasks. The Shuttle's Remote Manipulator System (RMS) has demonstrated a number of times its ability to perform certain basic teleoperator tasks. The tasks on the horizon, however, will certainly exceed the capabilities of the RMS by a wide margin. The construction, assembly, and checkout of the Space Station is an eminent example of such requirements.

One of the difficulties that NASA has had in deciding where and how to apply manipulators has resulted from not having a confident knowledge of their dexterous capabilities to perform complex, realistic space tasks or of how long the tasks will take to accomplish. The objective of the work reported in this paper was to address this issue by employing a teleoperated manipulator controlled by a highly-skilled, experienced operator to repeat a complex task already accomplished in space by EVA astronauts. This would both show that it could be done and provide as well a data base of task completion times. The task chosen was the Access I truss assembly which was done by EVA astronauts on STS-61b in November 1985. That flight experiment proved that astronauts can perform the basic operations required to assemble trusses in space.

The trusses are of the general type expected to form the framework of Space Station. The truss elements used were, however, about one fourth the size of the anticipated 5 meter lengths of the full scale ones on Space Station. Having chosen a task to perform, the next concern was what manipulator system to use. Probably the most skilled, dexterous, and experienced examples of teleoperated remote handling are to be found in operations involving the handling of radioactive materials. Because of this, Oak Ridge National Lab (ORNL) which has very extensive experience in the processing of nuclear materials was solicited to supply both skilled operators and its dual arm, master/slave manipulator [Central Research Lab (CRL), Model M-2] for use in these tests.

The ACCESS I Flight Experiment

The ACCESS I (Assembly Concept for Construction of Errectable Space Structure) was a structural assembly flight experiment intended to study and verify the ability of astronauts to assemble in space a repetitive truss structure representative of the type likely to become a part of Space Station. It was launched in November 1985 as a part of the Shuttle Mission STS-61b. During the ACCESS EVA (fig. 1) astronauts performed a rehearsed assembly line technique using a construction fixture as an aid. The on-orbit data which resulted has provided a basis for comparison and correlation with neutral buoyancy simulations practiced in preparation for the flight experiment.

The truss was assembled from basic hardware (figs. 2 & 3) which consisted of interchangeable, aluminum nodes and columns which can be snapped together to form connected bays of structure with a triangular cross section as shown in fig. 4. The horizontal and vertical members were 4.5 ft long and the diagonals 6.36 ft long with a two position locking sleeve on each end of each member. The nodes (fig. 2) each had six nubs to which the columns could be attached. The columns were mated to a node by sliding back the sleeve on the column's end and with a side approach intermeshing the fingers on the node's nub with those on the column's end. Finally the sleeve was slid back over the joint to make it secure.

Fig. 4 shows the equipment and general setup for the flight task with the astronauts in their designated places (no.'s 1 & 2). The nodes and columns were supplied from the canisters (no.'s 3, 4, & 5) which were located so that the astronauts did not have to leave their stations to build the truss. They used the assembly fixture (no. 6) as a frame on which to place and hold parts as the truss sections were being put together. Nodes were slid up the guide rails (no. 7) from the bottom to latching positions on the fixture. The columns were attached to these to form a finished bay which was subsequentially released and slid up along the guide rails to a new latched location to make room for the assembly of an additional bay on the lower half of the fixture where the raised bay had been.

Two bays at a time could reside on the assembly aid. The fixture was rotated by the astronauts at specific intervals during the construction to provide themselves access to parts of the truss they were supposed to be working on. Each astronaut had very specific duties in the assembly sequence which were repeated in cycles until ten bays of structure had been completed.

Equipment and Facilities

The Access truss remote handling experiments were performed at Oak Ridge National Laboratory's (ORNL's) Remote Operation and Maintenance Demonstration (ROMD) facility. The ROMD facility was developed by the U.S. Department of Energy's Consolidated Fuel Reprocessing Program to develop and demonstrate remote maintenance techniques for advanced nuclear fuel reprocessing equipment. Central Research Laboratory's model M-2 servomanipulator which was used for the Access teleoperator task is a dual-arm, bilateral force reflecting, master/slave servomanipulator developed jointly by CRL and ORNL and represents the state-of-the-art in commercially available teleoperator manipulators. The M-2, in operation since FY 1983, incorporates a distributed, microprocessor-based digital control system and was the first successful implementation of an entirely digitally controlled servomanipulator. Two major assemblies comprise the M-2: (1) the slave package shown in fig. 5 and (2) the master control station of fig. 6. The slave performs "man-like" handling tasks in the remote environment. The package consists of a pair of force-reflecting servomanipulator arms, three television viewing cameras, lighting, and a 230-kg (500-lb) capacity auxiliary hoist. Each slave arm has a 23-kg (50-lb) continuous capacity, a 46-kg (100-lb) time-limited (peak) capacity, and six degrees-of-freedom (joints) which are driven by brushless dc servomotors. The servomotors are mounted at the base of the arm and transmit power to the three degrees-of-freedom closest to the base through gears and linkage, and to the remaining three degrees-of-freedom plus the end-effector jaws by cable and pulley arrangements passing through arm tubes. Each servomotor has a servoamplifier and joint processor mounted in racks on the slave. The slave arm joint processor communicates with its respective master arm joint processor through a high-speed digital serial link. Basic control is through a closed-loop, position-position error technique.

Master-to-slave arm control is in real time with slave arm tip velocity capabilities up to 152 cm (60 in) per second. The minimum slave arm loading which can be detected or "felt" at the master control arm is on the order of 1 pound or 1 percent of peak capacity. All arm joints are force reflecting.

Operator viewing of the remote work site is provided by CCTV cameras mounted on the slave package. These include two boom-mounted cameras with four positioning degrees-of-freedom (pan, tilt, boom

extend-retract, and boom pivot) and motorized lens controls (zoom, focus, and iris); and one fixed camera mounted between the slave arms. The cameras provide standard resolution color video to 19-inch monitors at the master control station. The two boom-mounted cameras one on each side, provide orthogonal views for depth perception and viewing flexibility. The lower camera produces a wide angle view of the work site from a fixed position to give additional viewing information and information concerning master-to-slave arm spacial relationships.

Control of the slave is performed by a single operator from the master control station which consists of a pair of master arms, three 19-inch color television monitors, and an operator console (see fig. 6).

The six degree-of-freedom master control arms are kinematic replicas of the slave arms with each having a 25 lb. peak capacity. The handle on the master is a pistol grip and trigger type that provides slave tong control. Switches on the handles allow the operator to perform such functions as slave tong lock, slave arm lock, master-to-slave "all joint" indexing, and electrical tool power control without releasing the handle.

The operator interfaces with the control system for other functions primarily through a CRT and touch-screen mounted in the operator console. Operating mode selection, force-reflection ratio selection, camera/lighting control and system status diagnostics are available through this interface.

Camera and auxiliary hoist controls are also on the operator console. A joystick used for overhead camera positioning has spring loaded potentiometers to provide camera lens zoom, focus, and iris control.

Camera views selected to the three control station monitors are primarily the onboard slave camera views but can also be selected from several other facility and transporter-mounted cameras as desired by the operator. In addition, for this study, three other television monitors were arranged at the M-2 control station and could similarly be selected. They provided wide angle views (typically a field of approximately 10 ft by 10 ft) of the worksite which assisted operators in seeing and orienting the entire truss strut length.

The handling and assembly of the truss struts and nodes were performed without modification to the ACCESS components or the remote handling equipment. The manipulator tongs were fitted with standard finger pads. A flat-faced finger and a V-groove type finger were used on each tong. This finger combination produced a good grip on the tubular struts and countered any pivoting action at the finger contact points.

Remote handling operations were performed at a four-to-one slave-to-master force reflection ratio as preferred by the manipulator operators, although a range of ratios from 0 to 8:1 were available at their option.

Teleoperated Assembly of the ACCESS I Hardware

Experiment Procedure

Hardware from the ACCESS I flight experiment was taken to the ORNL and set up as shown in fig. 7 where it could be operated upon by the M-2. The assembly fixture (no. 1), actual flight hardware, was mounted on a wooden support fixture (no. 2) built originally for checkout of the ACCESS flight hardware. The nodes and struts were ones used by astronauts in water immersion training in preparation for STS-61b, but otherwise identical to the flight hardware.

During data runs the M-2 was alternately placed in each of the two assembly positions occupied by the astronauts [see fig. 1 and (1), (2) of fig. 4] in the flight experiments and from these positions, it along with a human subject in the other position (fig. 8) repeatedly constructed two bays (the first two) of the ACCESS truss. The manipulator's base was not allowed to translate while runs were taking place. After each construction the two bays were manually disassembled in preparation for the next run.

Two manipulator operators with extensive experience in remote handling controlled the M-2, each operator doing eight repetitions of the construction task in each operating position. Operators and operating position were rotated to distribute over all the runs the effects of learning and to minimize operator boredom. Each test subject performed two consecutive data runs in each position after which the manipulator was moved to the other operating position where the cycle was repeated.

A test procedure was prepared detailing the step by step subtasks of each builder to assemble the two bays. During the course of a run the manipulator operator performed alone all the required teleoperations including master/slave control of both manipulator arms as well as remote operation and adjustment of TV cameras and selection of desired remote site TV images on control station monitors.

The operator at the lower station had about 70 percent of the work to do including retrieving and installing all nodes, retrieving and installing all verticals, and retrieving and installing 2/3 of the horizontals. The node canister (no. 5) and the lower strut canister (no. 4) from which he got these parts are shown in fig. 4. The nodes were installed by sliding them from the bottom up the tapered guide rails on the assembly fixture. At the beginning of each construction

the three top nodes of the first bay were already in place. In addition to the above duties the lower station operator received diagonal struts handed to him by the operator at station #2 and attached them to appropriate nodes. Finally he was responsible for rotating the assembly fixture as needed for both parties to have required access to truss sections being worked on. In the flight assembly of ACCESS I either party could rotate or assist in rotating the fixture as needed; however, it was felt that if the same option were available in these studies that the person-half of the assembly team would likely end up doing all the rotating regardless of the station he was manning. Thus, station #1 was assigned to do it all the time while station #2 was never to do it.

The builder at station #2 had less work to do (about 30 percent of the total) including retrieving the diagonals from the upper strut canister (no. 3) and handing them to the operator at station #1 while connecting one of their ends to the upper nodes. He also installed the upper set of three horizontals on the first truss bay and completed upper connections of verticals after they had been installed on the lower nodes by the station #1 operator. Finally upon completion of the bay #1 assembly, station #2 released the truss latch which held one node of the lower bay firmly in place. He then raised that entire bay one level and secured it with the latch holding one of its lower nodes. The construction of the second bay then took place where the first bay had been.

Operators were given breaks frequently, these generally occurring between construction of each pair of bays.

The ACCESS struts had to be aligned in a particular way for their ends to mate with the nodes, but their symmetrical appearance, especially through closed circuit TV, made doing this very difficult. The problem was addressed by marking the struts to ensure reasonable attachment times. Marking was accomplished by putting a thin line on one side of the column along its major axis (see fig. 3) such that if the operator could see the mark centered in the manipulator jaws as he moved the column toward the node to which it was to be attached, then the alignment was approximately correct. To assist in quickly finding the line an additional mark in the form of a diamond was printed on the opposite side of the columm.

Data acquisition consisted of video taping to provide a visual record, real time observations of task completion times as well as certain other task element completion times, and observations of task performance errors. Time synchronized video recordings were made of (1) the M-2 operator's primary TV view at the M-2 console and (2) a general overall view of the assembly at the test site. Two observers one at the assembly site and the other in the M-2 control room kept these records and operated the video tape recorders. The "primary view" was recorded at the discretion of the control room observer as the one he believed at the time to be the one being used by the manipulator operator. Thus, it

was subject to change on a continuous basis. Performance errors are deviations from proper performance of the task consisting of such events as dropping parts or taking actions which require the task to be stopped. When one of these was recorded by an observer, he, at the same time, noted specifically what had happened.

Runs were begun with both the person assembler and the manipulator assembler poised to begin their first operation. The person-assembler gave a countdown to begin each run to all other participants over a radio headset. At this point, the components required to build the structure were all in their respective canisters and everything ready to go. Runs ended for data taking purposes when the tongs of the M-2 had released the final assembly component in the last assembly step.

RESULTS AND DISCUSSION

The notion that a complex task such as building the ACCESS I truss could be done with manipulators was uncertain and unproven prior to the studies of this report, although many believed such accomplishments were possible. These experiments have proven by demonstration that tele-operated manipulators have the required dexterity to perform the ACCESS task and by implication other similiar ones. In addition, a data base of times required to complete the task have been recorded. After eight runs the subjects at ORNL were able to assemble the ACCESS truss in a continuous, almost routine fashion, generally without incident. The bar graph of fig. 9 gives a comparison of assembly times for a variety of conditions including one-G, shirt sleeves; ground-based water immersion simulation with pressure suits; Shuttle flight; and teleoperated assembly at ORNL. All data are normalized to the completion of two bays. The teleoperator assembly time shown is an averaged figure computed from the last three runs of both M-2 operators (a total of twelve runs). This figure was used because it was expected that learning effects would be greater in the earlier runs, thus the later runs would be a more accurate indicator of stable teleoperator performance. The figure for the water immersion facility is an average of times from Johnson Space Center's Weightless Environment Training Facility (WETF) and Marshall Space Center's Neutral Buoyancy Simulator (NBS) and include some results from development tests with untrained subjects. As can be seen the teleoperator assembly took about three times as long as did the pressure-suited astronauts in space to achieve the operation. The teleoperator time is very good, however, when one considers that neither the hardware being assembled nor the manipulator itself had been designed to accommodate this task. A rule of thumb at ORNL is that tasks which require no more than eight times as long to do with the manipulator as for people to do directly are well suited for remote handling. An average time computed from the two very best runs made at each of the two stations, was only about two and one half times as long as for the astronauts.

The curve of fig. 10 summarizes the task completion times on a generally chronological basis for the subjects individually as well as their average. Each point represents a total time to build both bays. The curve for individual operators are averages of the runs they made at both stations: i.e., the point for operator #1, trial #1 is an average of his first run at station #1 and his first run at station #2. As can be seen there is a general decrease in times to complete the task with increasing replications indicating learning as would be expected. The effect seems to be much greater with subject #2 than with #1, although it is certainly there for both and readily shows up in the curve for their combined performance.

The average time to complete the task for both operators at each of the stations individually are shown in fig. 11 which presents these in a chronological progression. As before, learning effects are evident and seem to be somewhat more pronounced at station #2. The time required at station #2 was much less than at station #1. This is reasonable considering that there is much less work to do at station #2. In addition, acquiring and installing the nodes (which is second only in difficulty to installing the horizontal struts) is unique to station #1. Moreover, 67 percent of the horizontals themselves are installed at station #1.

The other performance measure applied to the teleoperator runs was the number of truss components dropped during the runs. Fig. 12 shows the total number of components dropped and is broken down by test subject, by node, and by column. Fig. 13 provides the distribution of these on a run by run basis. Note that 14 of the 18 drops occurred on three of the runs leaving an average of .8 drops/run for the other five runs with three of these runs actually having no drops at all. Data in this figure combine nodes and columns at both stations as well as subjects, i.e., the five drops for run #5 were computed as the total of two nodes at station #2 and one strut at station #1 for subject #1 on his fifth run at both stations and two struts at station #2 for subject #2 on his fifth run at that station. No nodes were dropped by subject #2 on his fifth run at station #1. On each run 6 nodes and 12 columns were handled at station #1 and zero nodes and nine columns at station #2. From this, it may be noted that each data point in fig. 13 includes 54 handled truss elements or opportunities to drop an element [18 @ station #1 + 9 @ station #2) * 2 subjects = 54]. Thus the maximum number dropped (5) on a run as shown in fig. 13 was only about 9.3 percent of the maximum possible. Over all of the runs only 4.2 percent of the elements were dropped.

All of the nodes that were dropped were dropped at station #1, if for no other reason than that they were not handled at all at station #2. On the other hand, all of the struts that were dropped, but one, were dropped at station #2. Furthermore, all but one of the incidents of dropping struts at station #2 were drops back into the strut canister resulting from strut ends slipping out of the end

effector jaws. A fair amount of this problem came from the fact that the ACCESS hardware was not designed for manipulator handling; this is especially true of removing components from canisters. The nodes were tightly packed into their canister making it very difficult to grasp them with the large, nondexterous, parallel-jaw end effector. Fig. 14 shows the open node canister. There was a similiar problem with the struts, but not as severe, because the clearance was greater among them. Unlike the EVA astronauts, the teleoperator at Oak Ridge had to deal with gravity which in the case of the vertically located strut canister at station #2 turned a little slip in grasp into a drop back into the canister. A vertical extraction was also required at the node canister. The other strut canister located at station #1 (see fig. 4), (no. 4) was oriented for extraction in the horizontal plane and apparently because of the orientation had no drop-back problem. Both the node canister and the vertical strut canister were located near the reach limits of the manipulator magnifying the grasp problem. The remaining drop of a strut at station #2 did not take place as a result of an end effector grasp problem but rather occurred while the teleoperator was passing a diagonal to its human construction partner and happened because of the mistaken belief that the human had grasped it. The only dropping of a strut at station #1 occurred on the same run that two nodes were also dropped and probably occurred because of fatigue (the M-2 operator said fatigue was the problem and the observers agree). The components dropped on this run were the most dropped by an M-2 operator on any one run. Of the remaining six dropped nodes two were drops back into the canister, one resulted from the inadvertant disengaging by the operator of the grip lock on the hand controller, and the other three were probably victims of handling complications. The nodes were difficult to deal with. After they came out of the canister they had to be reoriented by grasping, turning, passing off, and regrasping using both end effectors to poise them for placement on the guide rails.

Work load was very high during these experiments. The requirement for dual arm activity was much greater than the operators were accustomed to. The task was intense, perhaps overly so, because of the competition with the clock. For at least these reasons, operators reported fatigue to have been significant throughout the course of the tests.

Discussions with the subjects and observations by those conducting the tests have identified several modifications of the hardware which if implemented should positively influence the teleoperator performance of this task:

1. Flat grasping points on the columns and the nodes. These would serve to produce grasping compatibility between work pieces and end effector and as well provide a means for incorporating indexing to expedite alignment.

2. Sleeves on the column ends without mechanisms which hold them in cocked and locked back positions.

3. Good audio from the remote site. Sound would provide an additional channel of information to the operator without imposing any requirements for directing attention. Completion of operations would be indicated by sounds such as the snap of the column sleeve sliding over the node end at the connection or of the operation of the latch securing the position of the upper bay on the construction fixture. Undesired hardware impacts would be heard, etc.

4. Better markings on struts to indicate orientation.

5. Canisters located and oriented to accommodate the manipulator. The canister location should be far enough away to permit extraction of the columns without the manipulator pushing them into its own base and jamming. The distance away that the canister is located should not, however, be so far as to cause greatly diminished dexterity because of proximity to reach limits.

CONCLUDING REMARKS

Teleoperator experiments were conducted which have demonstrated that a realistic, complex task, typical of those accomplished on-orbit by EVA astronauts, can be done in a smooth, timely manner with manipulators remotely controlled by humans. The real concerns were: 1) Do manipulators have sufficient dexterity for these tasks? 2) Can sufficient information from the remote site be provided to permit adequate teleoperator control? 3) Can reasonable times relative to EVA times be achieved? and 4) Can the task be completed without frequent and/or damaging impacts among the task components and the manipulators? Positive answers were found to all of these concerns. Task times, operator fatigue, and smoothness of operation could be improved by designing the task components and the manipulators for greater compatibility.

Because of certain very specific operations such as turning and sliding the sleeves on the column ends, and holding the latch open while raising the first bay; coordinated dual-arm manipulation was definitely a requirement in this task. However, there were strong indications as well that the general task may not have been doable without two arms even if modifications had been made to accommodate with one arm the idiosyncrasies of these particular subtasks. Especially in one G, final alignment of the horizontal columns to mate with the node ends benefits greatly from being able to support the column simultaneously at both its ends. Two arms permit rotation about specific points by one arm holding

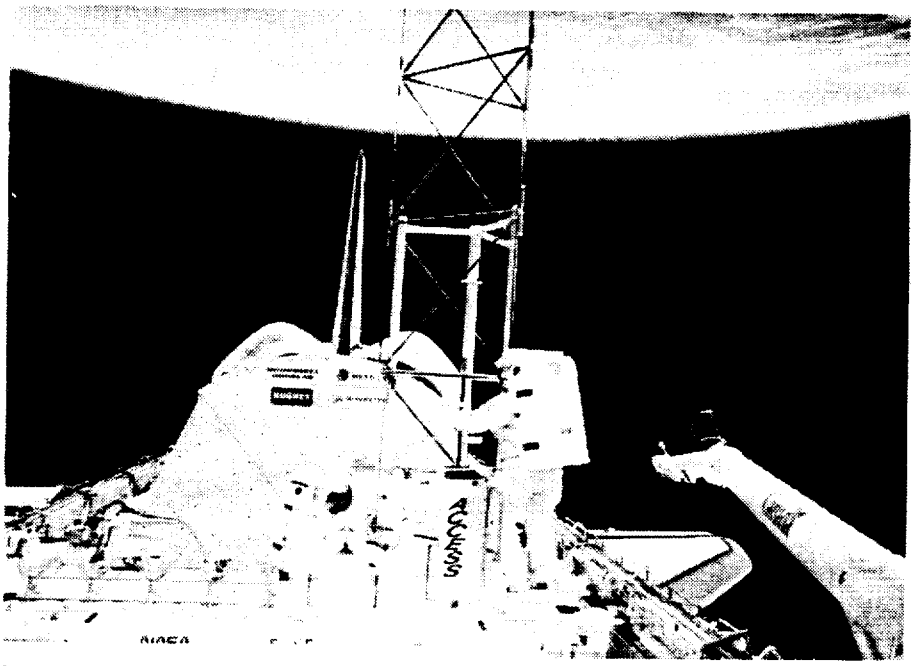


Figure 1 Astronauts Ross and Spring Assemble ACCESS on STS 61b

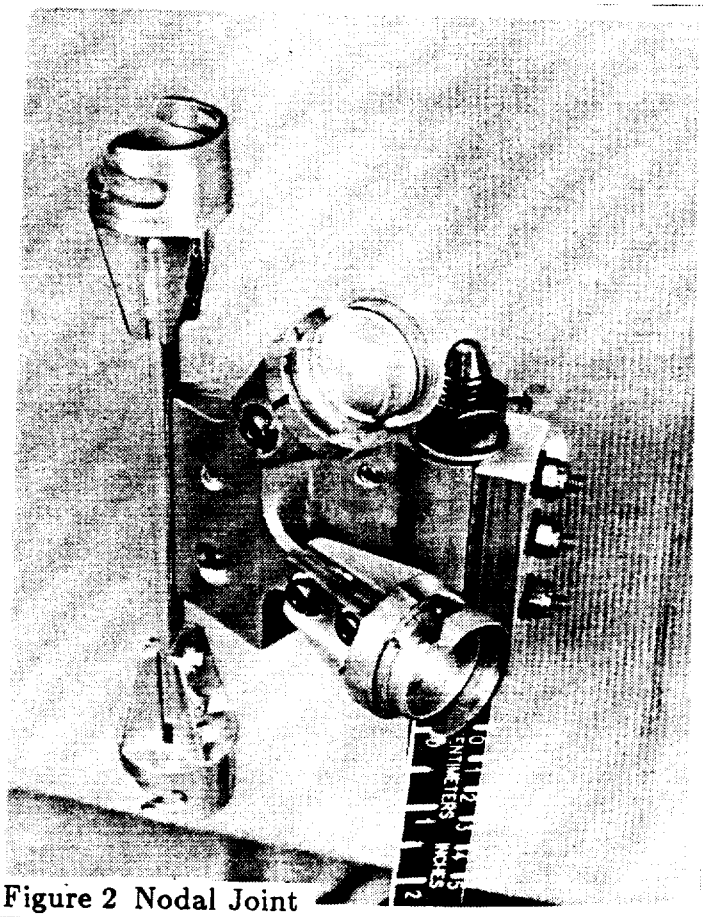


Figure 2 Nodal Joint

C-5

the part at that point while the other generates a moment about it. Node reorientation is facilitated by using one arm to hold while the other regrasps, etc.

The data recorded supplements a data base of performance metrics for the same task done in the water immersion training facilities as well as space flight and provides management with a objective basis for deciding how and where to apply manipulators in space.

REFERENCES

1. Heard, W. L., Jr.; and Watson, J. J., LaRC; Ross, J. L., Spring, S. C.; and Cleave, M. L., JSC: Results of the Access Space Construction Shuttle Flight Experiment. NASA Paper Presented at AIAA Space Systems Technology Conference, San Diego, California, June 9-12, 1986.

2. Burgess, T. W.: The Remote Operation and Maintenance Demonstration Facility at the Oak Ridge National Laboratory, presented at the ANS Topical Meeting on Waste Management and Decontamination and Decommissioning, Niagara Falls, New York, September 14-18, 1986.

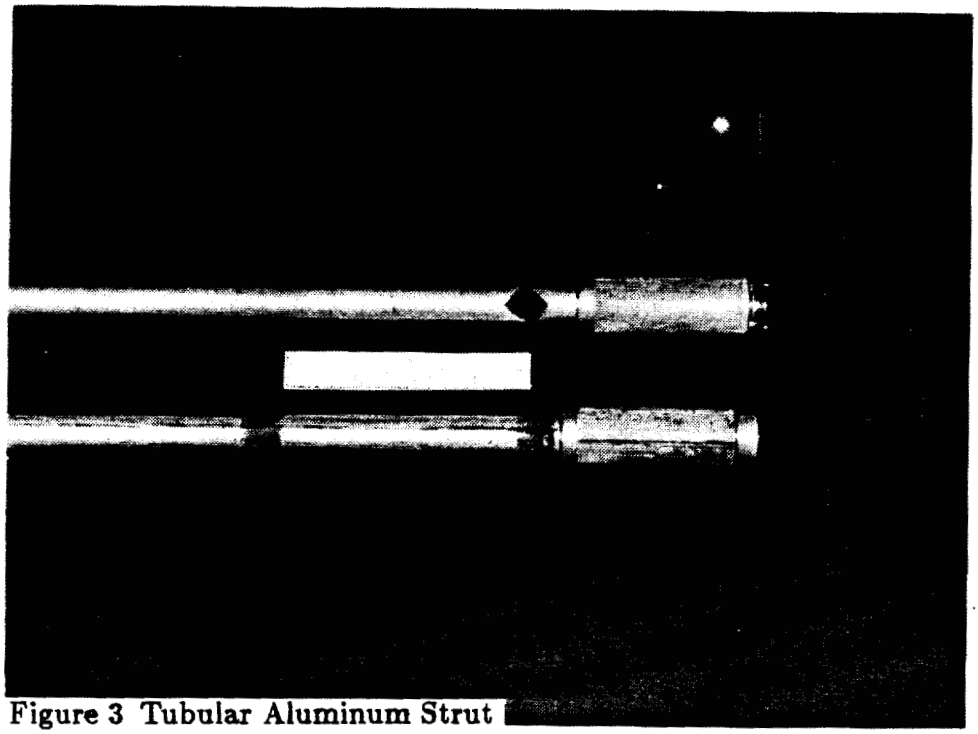


Figure 3 Tubular Aluminum Strut

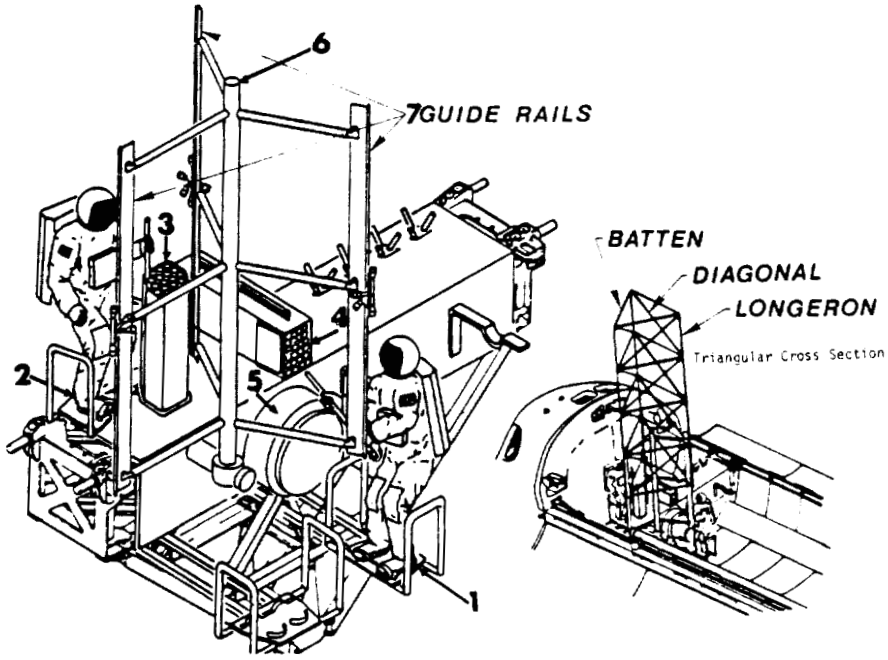


Figure 4 Schematic Showing EVA Construction of ACCESS Truss on Assembly Fixture

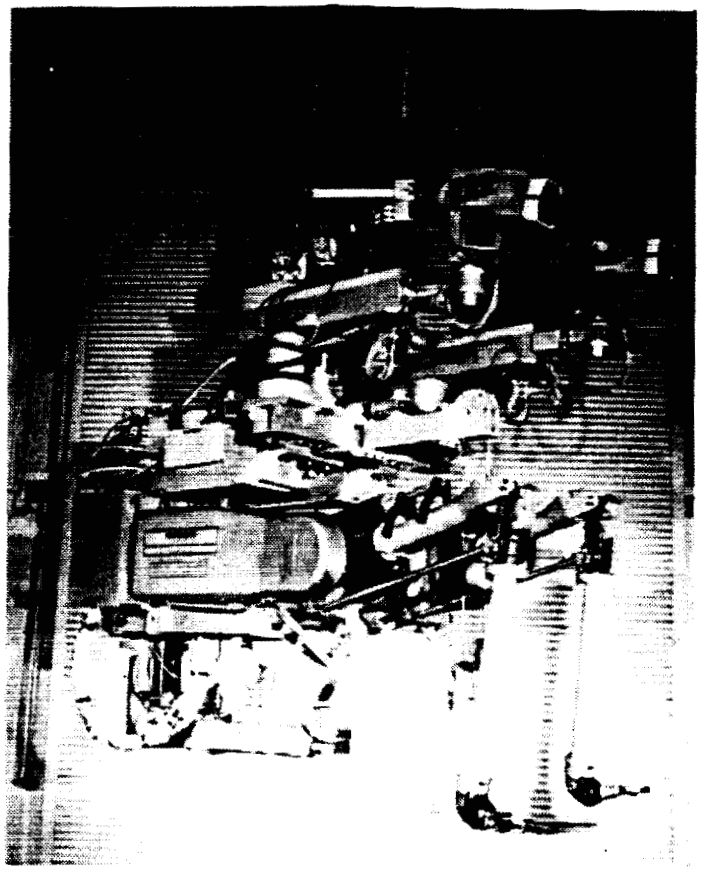


Figure 5 CRL Model M-2 Servomanipulator Slave Package

Figure 6 Master Control Station of Model M-2 Servomanipulator

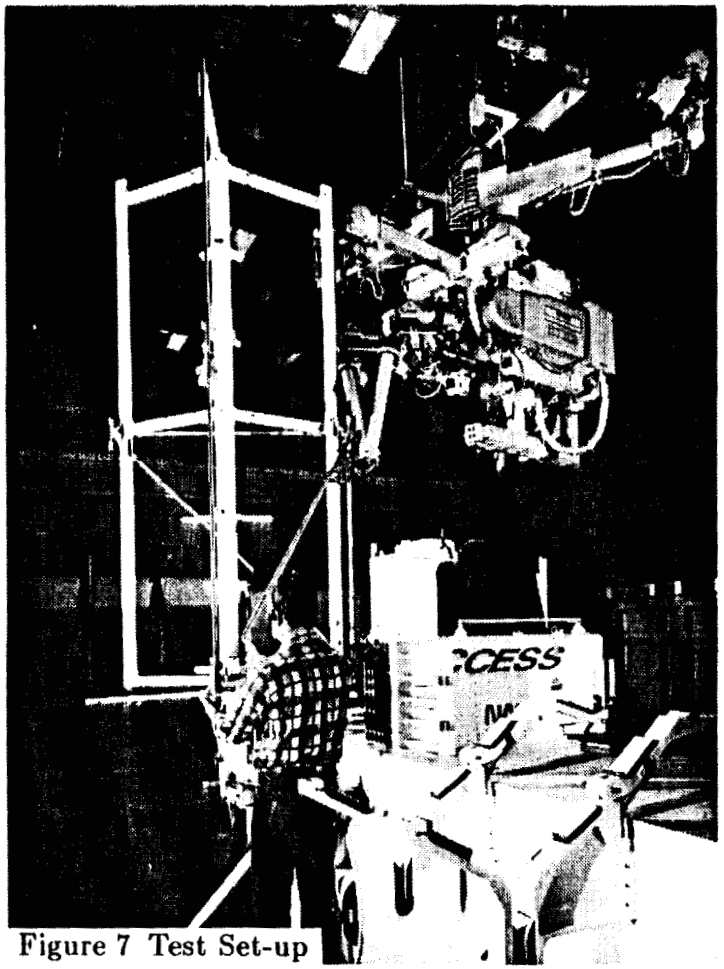


Figure 7 Test Set-up

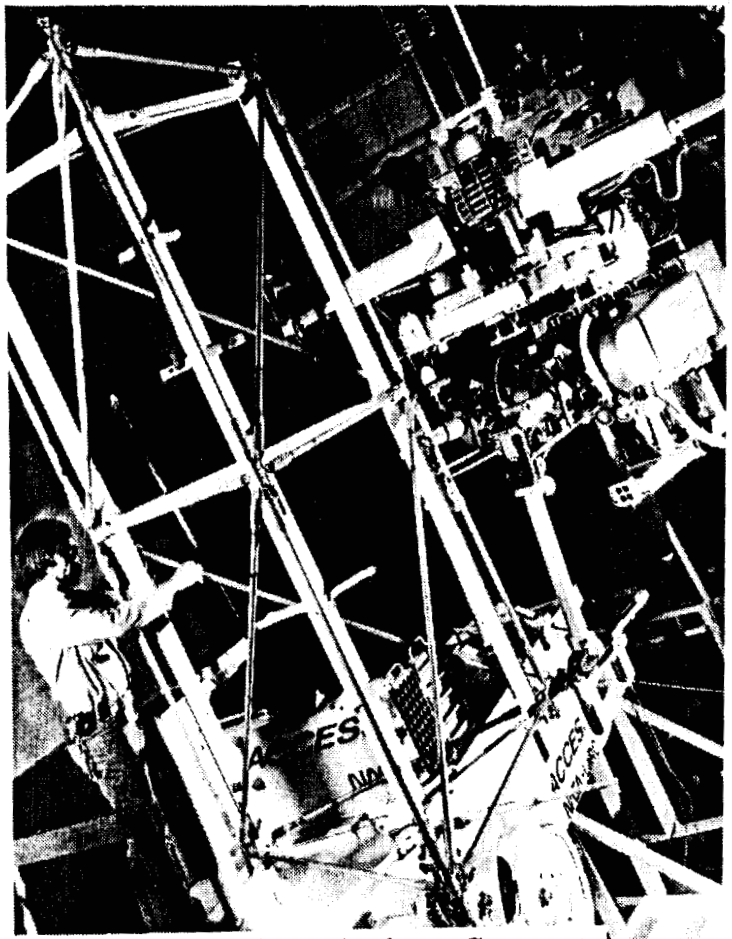


Figure 8 Man and Manipulator Cooperate to Construct Two Bays

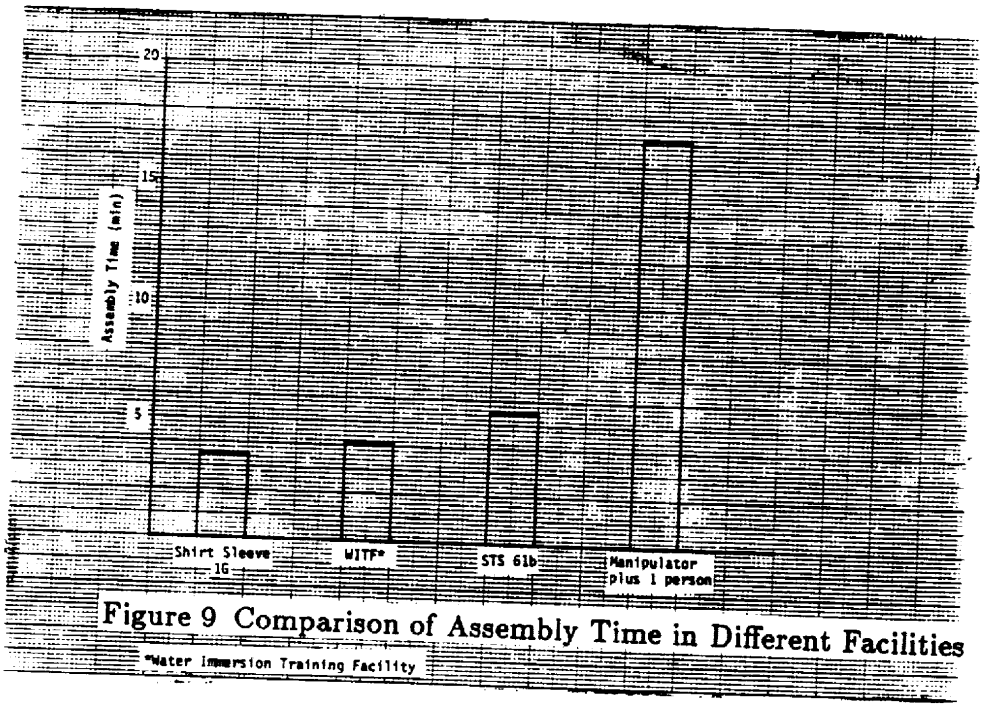


Figure 9 Comparison of Assembly Time in Different Facilities

*Water Immersion Training Facility

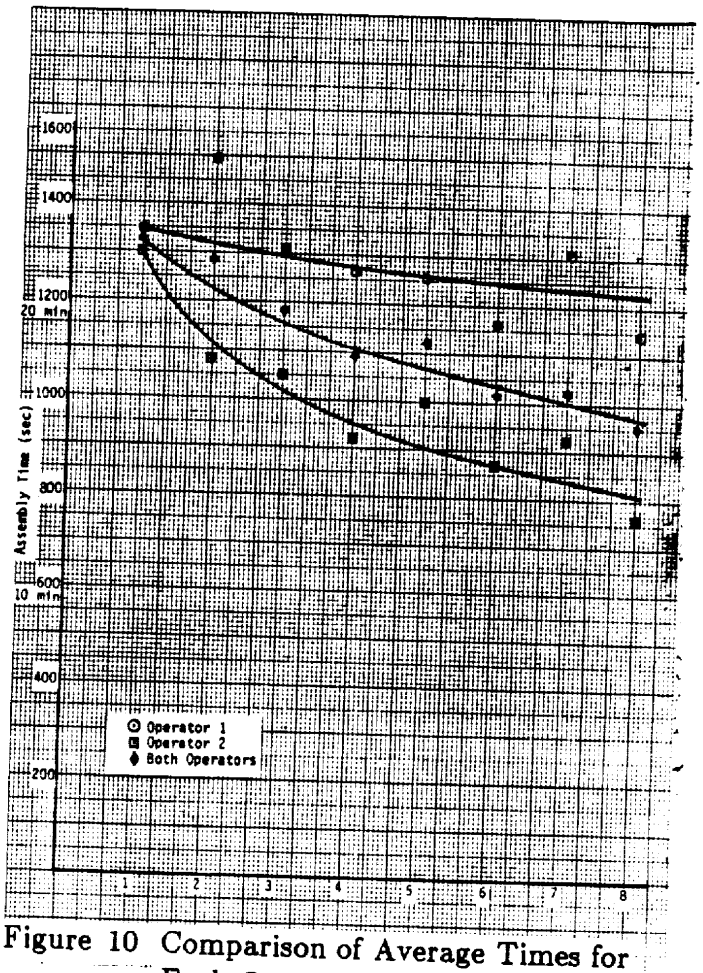


Figure 10 Comparison of Average Times for Each Operator

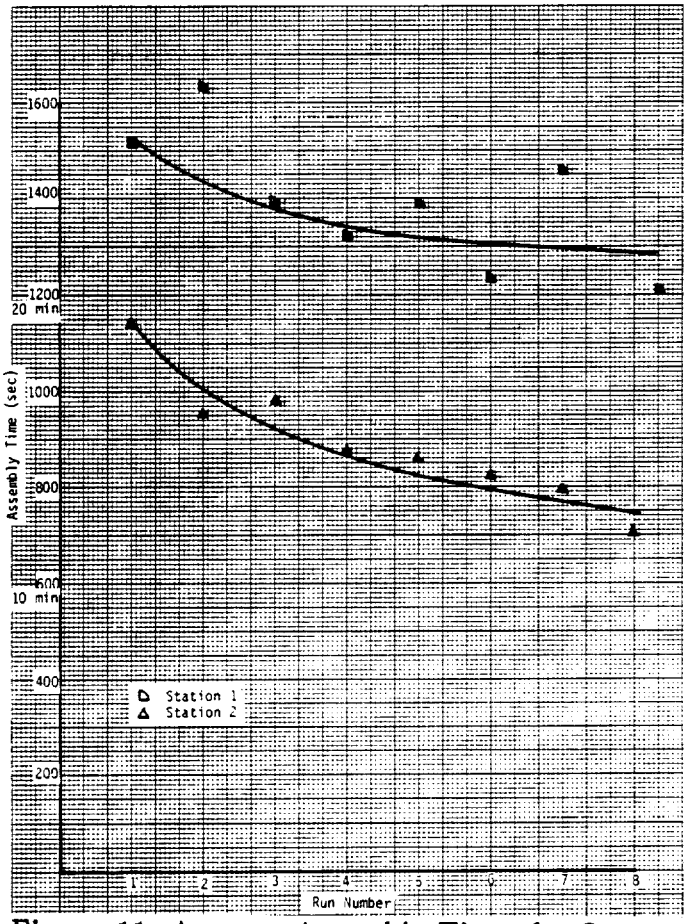


Figure 11 Average Assembly Times by Operator

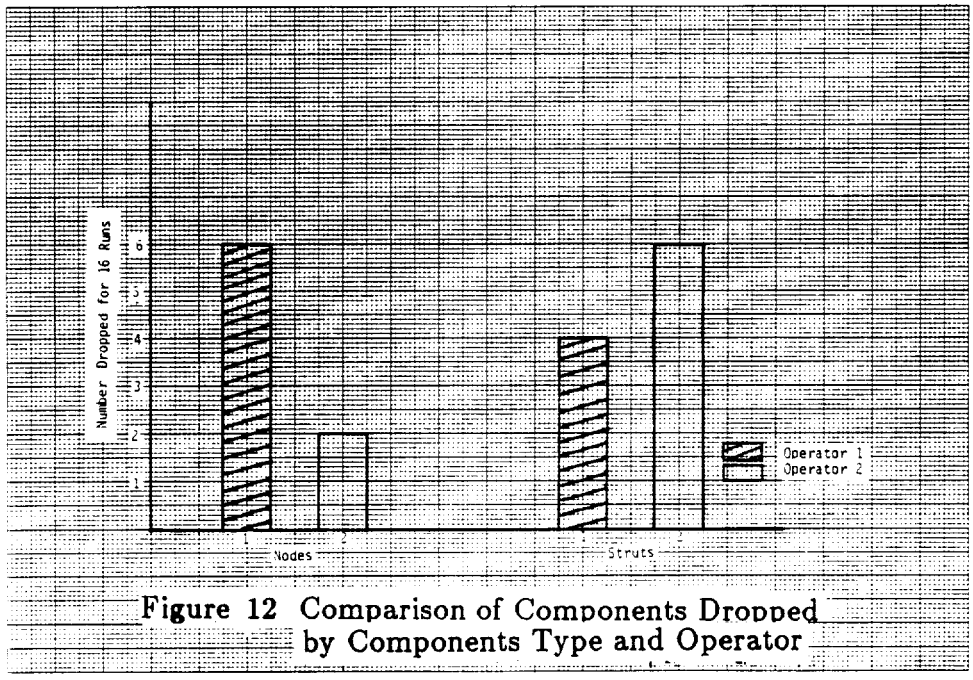


Figure 12 Comparison of Components Dropped by Components Type and Operator

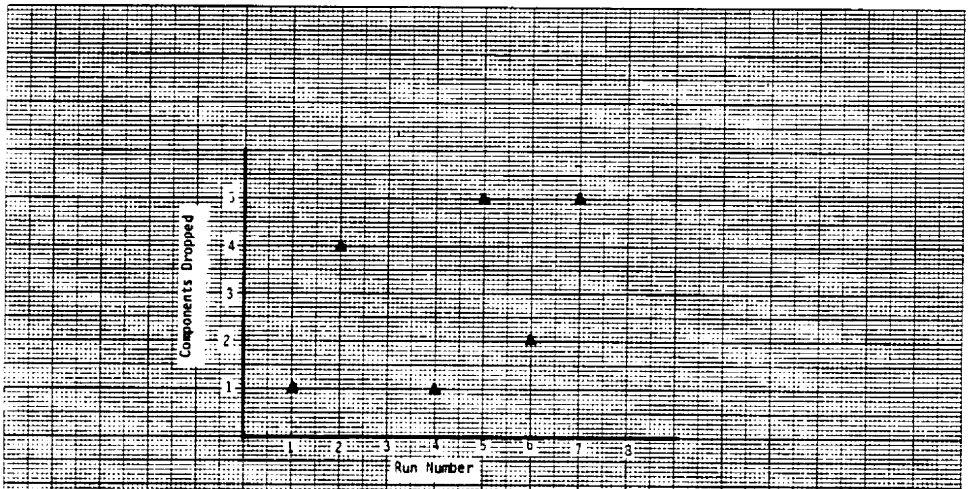


Figure 13 Total Dropped by Run

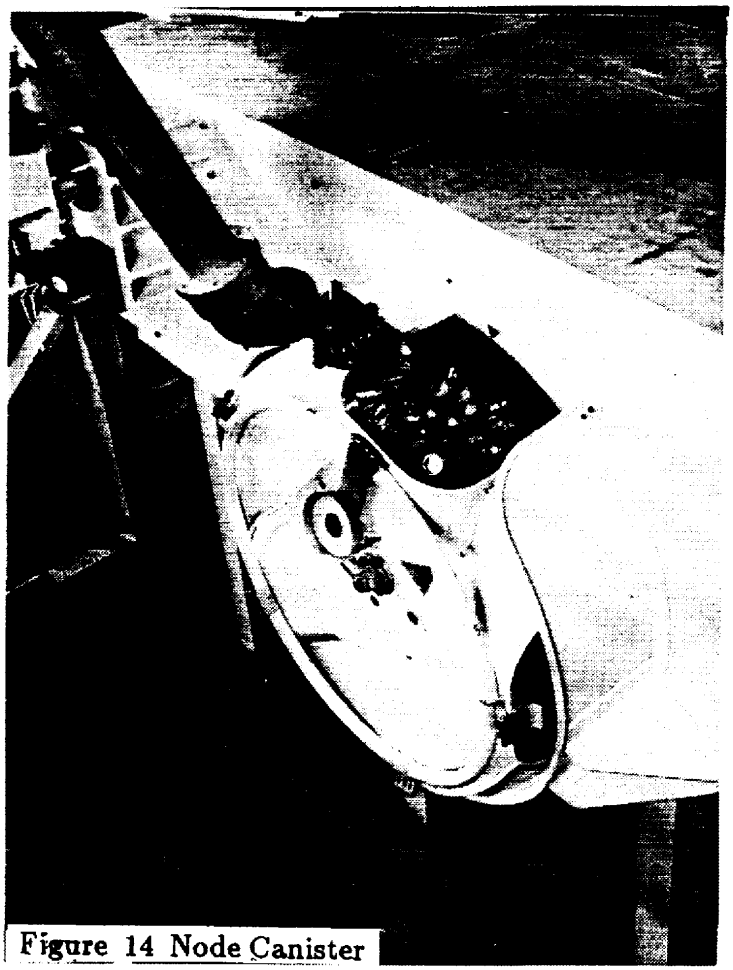


Figure 14 Node Canister

A UNIVERSITY TEACHING SIMULATION FACILITY

Lawrence Stark, Won-Soo Kim, Frank Tendick, Mitchell Tyler, Blake Hannaford, Wissam Barakat[+], Olaf Bergengruen[*], Louis Braddi[+], Joseph Eisenberg[+], Stephen Ellis[*], Steven Ethier[+], Denise Flora[+], Sanjay Gidwani[+], Ronald Heglie[+], Ted Jordan[*], Nam Heui Kim[*], Bryan Martel[+], Mark Misplon[+], Eric Moore[+], Steven Moore[+], An Nguyen[*], Cecilia Nguyen[+], Scott Orlosky[+], Girish Patel[+], Michael Rizzi[+], Eric Shaffer[+], Mitch Sutter[+], Harris Wong[+]

Telerobotics Unit, University of California

Berkeley, California 94720

ABSTRACT

An experimental telerobotics (TR) simulation is described suitable for studying human operator (H.O.) performance. Simple manipulator pick-and-place and tracking tasks allowed quantitative comparison of a number of calligraphic display viewing conditions.

A number of control modes could be compared in this TR simulation, including displacement, rate and acceleratory control using position and force joysticks. A homeomorphic controller turned out to be no better than joysticks; the adaptive properties of the H.O. can apparently permit quite good control over a variety of controller configurations and control modes. Training by optimal control example seemed helpful in preliminary experiments.

*Research staff

+Students in graduate bioengineering class ME 210, Biological Control Systems, Fall 1985.

An introduced communication delay was found to produce decrease in performance. In considerable part, this difficulty could be compensated for by preview control information. That neurological control of normal human movement contains a sampled data period of 0.2 seconds may relate to this robustness of H.O. control to delay.

The Ames-Berkeley enhanced perspective display was utilized in conjunction with an experimental helmet mounted display system (HMD) that provided stereoscopic enhanced views. Two degree-of-freedom rotations of the head were measured with a Helmholtz coil instrument and these angles used to compute a directional conical window into a 3-D simulation. The vector elements within the window were then transformed by projective geometry calculations to an intermediate stereoscopic display, received by two video cameras and imaged onto the HMD mini-display units (one-inch CRT video receivers) mounted on the helmet.

Acknowledgements We are pleased to acknowledge support from the NASA-Ames Research Center (Cooperative Agreement NCC 2-86) and the Jet Propulsion Laboratory, California Institute of Technology (Contract #956873).
We would also like to thank visiting lecturers from NASA-Ames Mark Cohen, Stephen Ellis, Scott Fisher, Arthur Grunewald, John Perrone and Mordeccai Velger; Constance Ramos and Christopher Clark of University of California Berkeley.

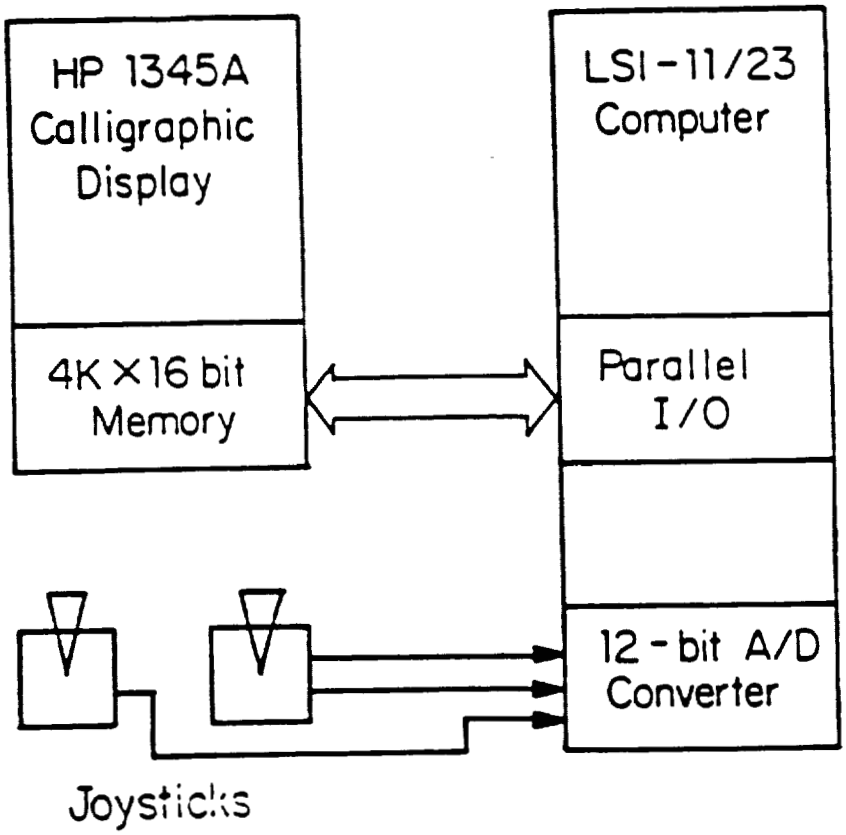
INTRODUCTION

A telerobotic, TR, system is defined as a distant robot with vision and manipulator and/or mobility subsystems controlled by a human operator, HO. The HO is informed mainly by a visual display, but also by other sensors and other sensory displays, i.e. auditory, force or tactile. His control can be direct via joysticks, or supervisory via command and control primitives effected by paritally autonomous robotic functions. Delays and bandwidth limitations in communication are key problems, complicating display and control (Stark, Kim, Tendick, et al, 1986).

The research presented here was initially carried out by the students taking a graduate control course, ME 210 "Biological Control Systems: Telerobotics."

EXPERIMENTAL SET-UP FOR THREE-AXIS PICK-AND-PLACE TASKS

A teleoperation simulator constructed with a display, joysticks, and a computer enabled three-axis pick-and-place tasks to be performed and various display and control conditions evaluated (Figure 1). A vector display system (Hewlett-Packard 1345A) was used for fast vector drawing and updating with high resolution. In our experiments, displacement joysticks were mainly used, although in one experiment a force joystick was used to compare with a displacement joystick. An LSI-11/23 computer with the RT-11 operating system computer was connected to the joystick outputs through 12-bit A/D converters, and to the vector display system through a 16-bit paraller I/O port.



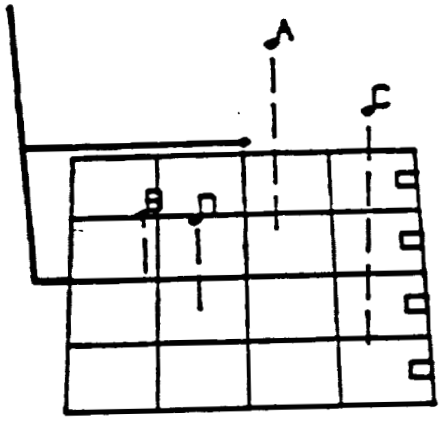
Experimental Arrangement

Figure 1

A typical presentation on the display screen for three-axis pick-and-place tasks included a cylindrical manipulator, objects to pick up, and boxes in which to place them, all displayed in perspective (Figure 2). Since perspective projection alone is not sufficient to present three-dimensional information on the two-dimensional screen, a grid representing a horizontal base plane and reference lines indicating vertical separations from the base plane are also presented (Ellis, Kim, McGreevy, Tyler and Stark, 1985; Kim, Ellis, Tyler and Stark, 1985). The human operator controlled the manipulator on the display using two joysticks to pick up each object with the manipulator gripper and place it in the corresponding box. One hand, using two axes of one joystick, controls the gripper position for the two axes parallel to the horizontal base plane (grid). The other hand, using one axis of the other joystick, controls the gripper position for the third axis (vertical height) perpendicular to the base plane. Picking up an object is accomplished by touching an object with the manipulator gripper. Likewise, placing an object is accomplished by touching the correct box with the manipulator gripper.

Puma Arm Simulator

In addition to the cylindrical manipulator simulation, the kinematics and dynamics a six degree-of-freedom Puma robot arm were simulated. Each of these degrees of freedom were controlled simultaneously using two joysticks. Although no experiments have yet been performed with the puma simulation, it



Ames-Berkeley Visual Enhancement Display

Figure 2

is hoped that it will be a step toward experiments with more complex manipulators. A low-bandwidth telephone connection to control two puma arms at Jet Propulsion Labs in Pasadena is planned. The simulation will allow prediction of the robots motion to provide a preview display to help overcome the communication delays inherent in such a low bandwidth connection, or as in transmissions to manipulators in space.

CONTROL MODE EXPERIMENTS

Position and rate controls are the two common manual control modes for controlling telemanipulators with joysticks (or hand controllers) (Johnsen & Corliss, 1971; Heer, 1973). In the position control the joystick command indicates the desired end effector position of the manipulator, whereas in the rate control the joystick command indicates the desired end effector velocity.

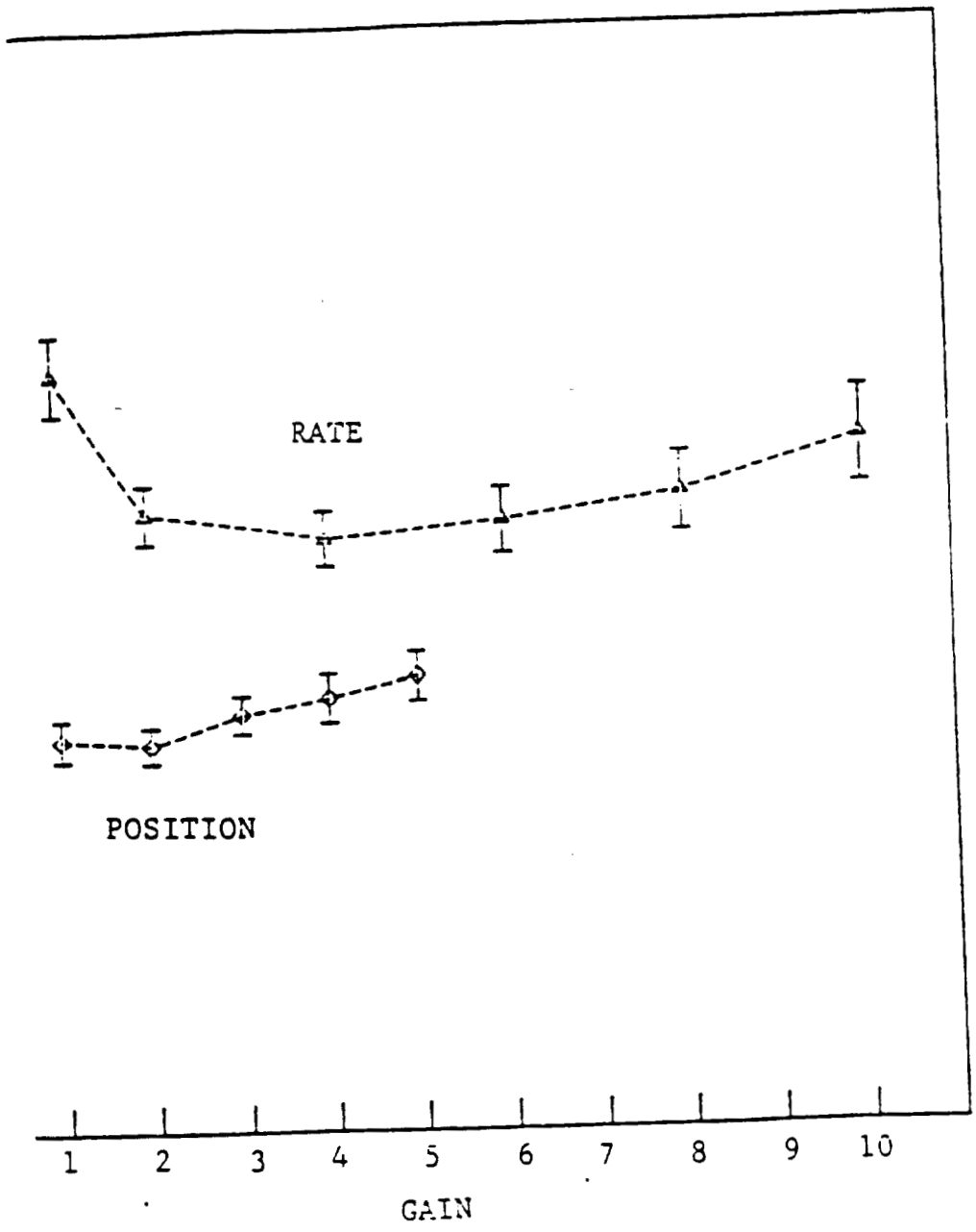
In our three-axis pick-and-place tasks, the human operator controls the manipulator hand position in the robot base Cartesian coordinate by using three axes of the two displacement joysticks. In pure (or ideal) position control, the system transfer function from the joystick displacement input to the actual manipulator hand position output is a constant gain G_p for each axis. In pure rate control, the system transfer function is a single integrator G_v/s for each axis. In the rate control, a 5% dead-band nonlinearity is introduced before the pure integrator in order to inhibit the drift problem associated with the pure integrator.

Comparison of pure position and rate controls

Three-axis pick-and-place tasks were performed with both pure position and rate control modes for various gains (Figure 3). The mean completion time plot clearly shows that pick-and-place performance with pure position control (mean completion time 2.8 seconds at $G_p = 2$) was about 1.5 times faster than that of the pure rate control (mean completion time 4.3 seconds at $G_v = 4$).

Trajectories of Joystick and Manipulator Movements

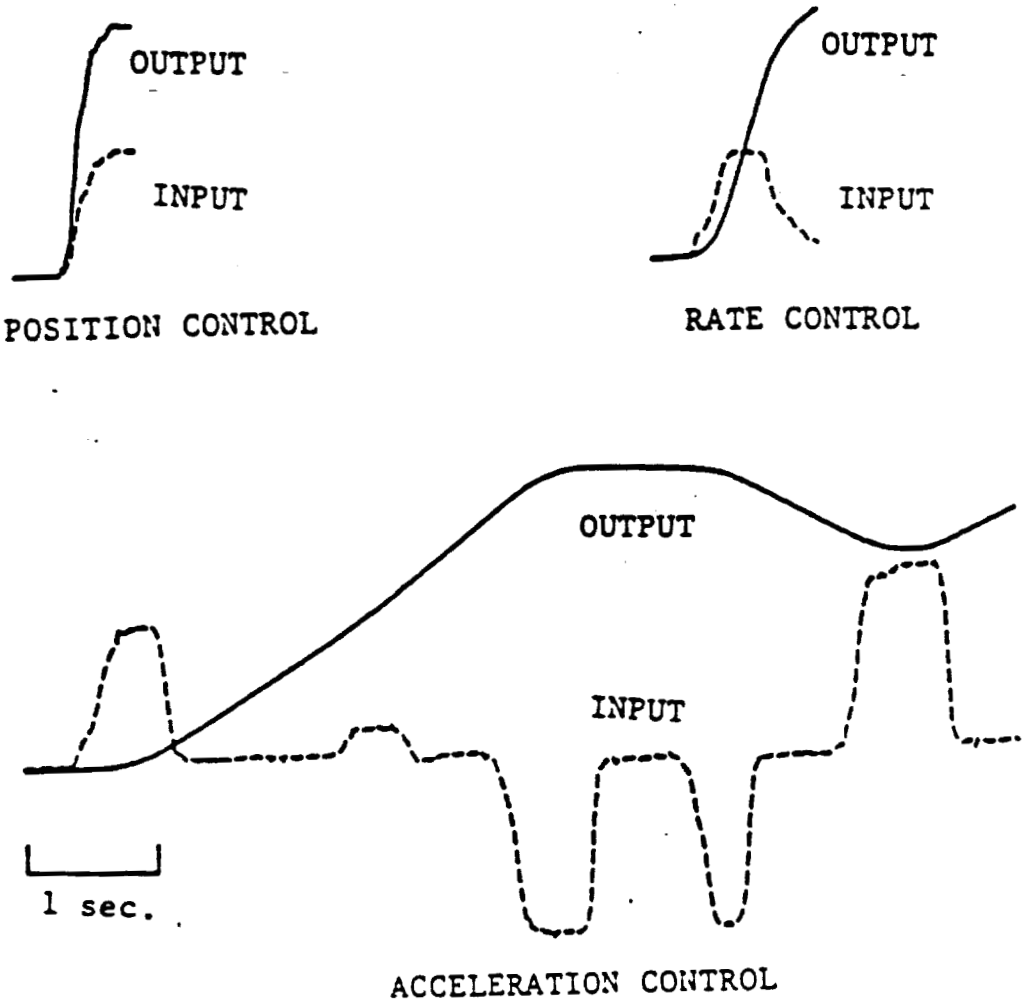
In order to examine why the position control performed better than the rate control, several trajectories of the joystick displacement input and the manipulator hand position output during the pick-and-place operation were observed. Typical trajectories from the start of trying to pick up an object to its accomplishment were plotted to illustrate position, rate, and acceleration controls (Figure 4). Components only for the x-axis (side-to-side) are plotted, since components for the other two axes are similar. Observation of several trajectories indicates that a precise re-positioning of the manipulator hand is achieved by a combination of quick step re-positioning operations and slow smooth movement operations. In position control one quick step re-positioning of the manipulator hand from one position to another requires one joystick pull or push operation, whereas in the rate control it requires a pair of operations; pull-and-push or push-and-pull operations (Figure 4). This is a major reason why the position control yielded better performance than the rate control for our



rmance Comparison of Position and Rate Control. Note low

:ivity to gain change.

Figure 3



Position, Rate and Acceleration Control. Typical trajectories of HO control and resultant manipulator output illustrate these control modes.

Figure 4

pick-and-place tasks. It should be noted, however, that the pick-and-place task is a positioning task. If the task is following a target with a constant velocity, then velocity (rate) control would perform better.

Acceleration Control

Three-axis pick-and-place task were also tried with acceleration control. It turned out, however, acceleration control was not adequate to perform stable, safe pick-and-place operations. In acceleration control, the manipulator tends to move almost all the time even though the joystick is at the center position. Note that in pure rate control, the manipulator does not move when the joystick is at the center position regardless of previous history of the joystick displacement.

Human Adaptation to Gain Change

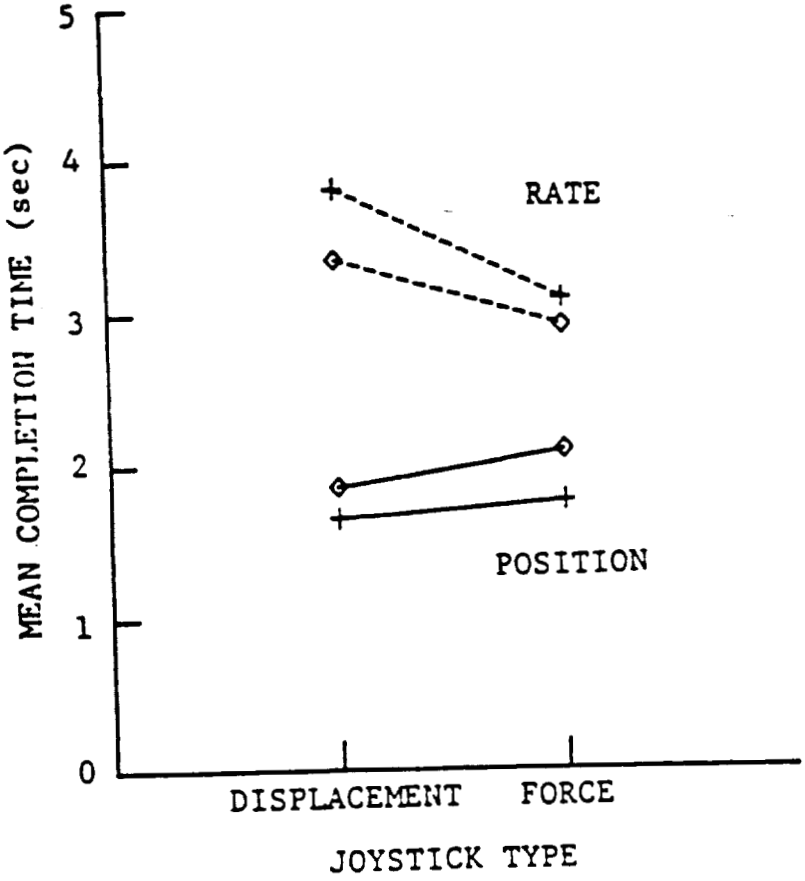
Mean completion time did not change much for the various gains tested (Figure 3), which means that the human operator adapted well to the gain change (McRuer, et al, 1965; Young, 1969; Stark 1968). Both lower and higher gains relative to the optimal gains caused slight increase in the mean completion time. A reason of slightly longer mean completion times with lower gains is because lower gains demand wider joystick displacements and it takes longer for the finger or hand to displace the joystick wider. A reason for slightly longer mean completion times with higher gains is the higher gains

demand more minute joystick displacements, degrading effective resolution of the joystick control. An additional major reason for longer mean completion times with lower gains for the rate control is due to the velocity limit.

Force joystick

The two common joystick types are the displacement and force joysticks. The output of the displacement joystick is proportional to the joystick displacement, whereas the output of the force joystick (isometric or stiff joystick) is proportional to the force applied by the human operator. The advantage of the force joystick is that it requires only minute joystick displacements (a few micrometers) in contrast with the displacement joystick (a few centimeters).

Pick-and-place tasks were performed for pure position and rate controls with displacement and force joysticks. The experimental results for two subjects (Figure 5) shows that in the rate control, task performance with force joystick was significantly faster than that with displacement joystick. This is mainly because the force joystick senses the applied force directly, requiring only very minute joystick displacements. In the position control, however, the force joystick performed no better than the displacement joystick. In fact, all three subjects preferred to use the displacement joystick in this mode, since the force joystick required more force to be applied than the displacement joystick , especially when the manipulator hand is to be positioned far away from the initial center position. Position control also performed better than the rate control regardless of joystick types, and furthermore the position control with the displacement joystick performed best for our pick-and-place tasks (Figure 5).



Displacement and Force Joystick Control. Note adaptive ability of HO to utilize these different joysticks. Position control is superior to rate control in the particular task studied. Two subjects: diamond (WK), cross (MT).

Figure 5

Resolution

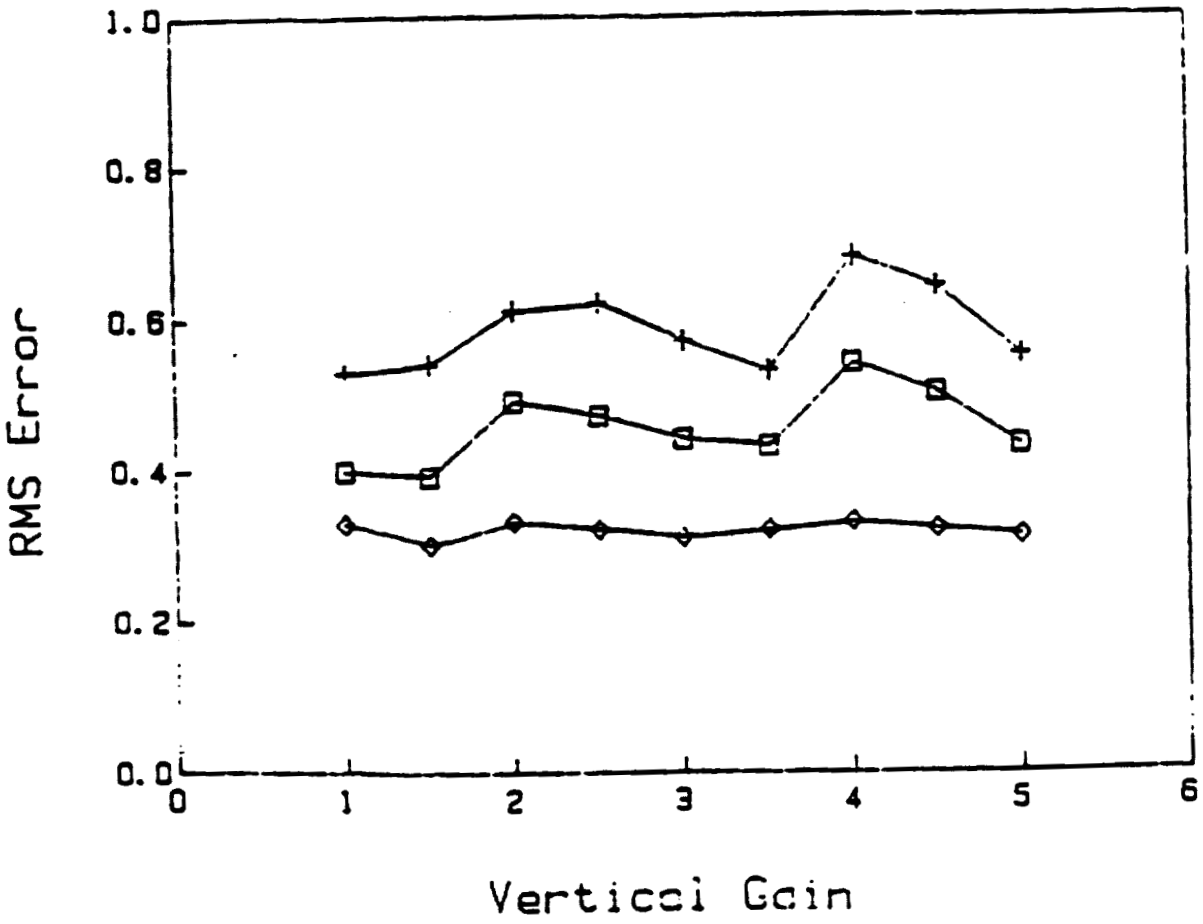
The experimental results demonstrate the superiority of position control when the telemanipulator has a sufficiently small work space (Figures 3, 4, & 5). Note that our three-axis pick-and-place tasks used in this experiment implicitly assumes that the manipulator work space is small or at least not very large, since our task allows the human operator to perform successful pick-and-place operations with a display showing the entire work space on the screen. Examples of small work space telemanipulators can be found in nuclear reactor teleoperators, surgical micro-telerobots, or small dexterous telerobotic hands. Position control can also be utilized during proximity operations in conjunction with the force-reflecting joysticks for enhanced telepresence (Bejczy, 1980). When the telemanipulator's work space is very large as compared to human operator's control space, position control of the entire work space suffers from poor resolution since human operator's control space must be greatly up-scaled to accommodate the telemanipulator's large work space (Flatau, 1973). One way of solving this poor resolution problem in position control is using indexing Johnsen & Corliss, 1971; Argonne National Lab, 1967). In the indexed position control mode, the control stick gain is selected so that the full displacement range of the control stick can cover only a small portion of the manipulator work space, and large movements of the manipulator hand can be made by successive uses of an indexing trigger mounted on the control stick. Note, however, that rate control can inherently provide any higher degree of resolution by mere change of control stick gain without use of indexing.

HOMEOMORPHIC CONTROLLER

Most of our pick-and-place and tracking experiments were performed with joysticks as the input device through which the human operator controlled the simulated manipulator. The operator's movements when using joysticks are non-homeomorphic, so that the movements he must make to produce a desired manipulator response do not match the movement of the manipulator and effector. Thus, he must mentally convert the desired end effector position to Cartesian coordinates and use the joysticks to input these coordinates.

To attempt to study whether a truly homeomorphic input device could improve performance in tracking tasks, an apparatus of identical form to our simulated cylindrical manipulator was built. A vertical rod was supported by bearings on the base to allow rotation, theta. A counterweighted horizontal arm was attached to the rod with sliding bearings to permit rotation and translation in the r and z axes respectively. The human operator could control position through a handle on the end of the arm corresponding to the end effector of the simulated manipulator. Potentiometers measured movement in each axis to determine input r, theta, and z. The LSI -11/23 computer read these values through A/D channels and displayed the manipulator in the identical position.

Three-dimensional tracking experiments were performed with the homeomorphic controller and with joysticks for gains varying from 1 to 5 to compare performance (Figure 6). The results do not show a significant difference between the homeomorphic controller and joysticks over the range of



Homeomorphic Controller. Note similar low sensitivities to gain for all three axes. (x-axis, diamonds; y-axis, crosses; and z-axis, squares). Figure 6

gain values. Although the larger movements required for the homeomorphic controller, with greater inertia and friction than the joystick, may have limited performance, we believe that human adaptability minimizes its advantages.

TRAINING BY OPTIMAL CONTROL EXAMPLE

A simplified simulation of the manned maneuvering unit, MMV, enabled study of training of human control performance (Jordan, 1985). Only three translatory degrees-of-freedom, x, y and z, were used. Thrusters generating pulses of acceleratory control were controlled via a keyboard and the task was to accelerate simultaneously in x, y and z to a maximum velocity, transit to the desired new location, and decelerate again simultaneously. Two displays were used -- a perspective display of a minified model of the MMV, or two two-dimensional projectors of that model with a small inset of the perspective display.

Subjects generally performed poorly during the few hundred seconds allowed for the tasks (upper panels, Figure 7). It was decided to allow the subjects to view this control problem carried out by a simple optimal control algorithm (see middle panel, Figure 7). This experience was of considerable help and several subjects then performed quite well (bottom panel, Figure 7).

This experiment, learning-by-example, illustrates a strategy that perhaps may be effective in more complex and realistic tasks as well.

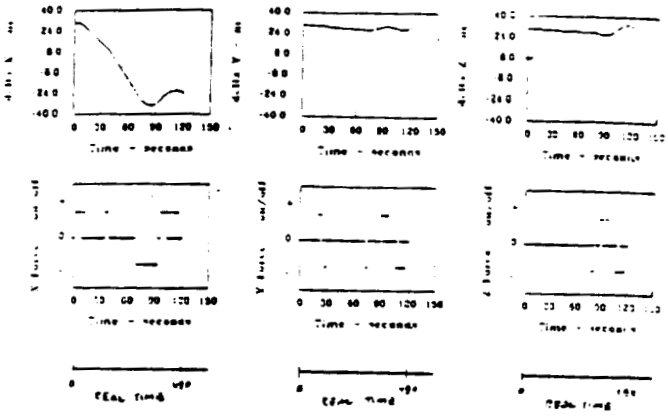


Figure 7 Top

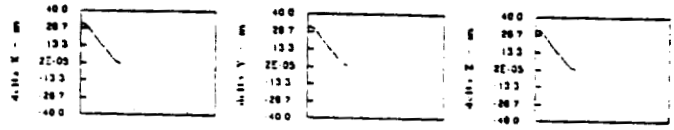
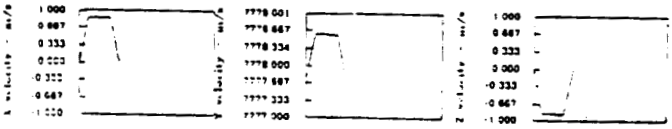


Figure 7 Middle



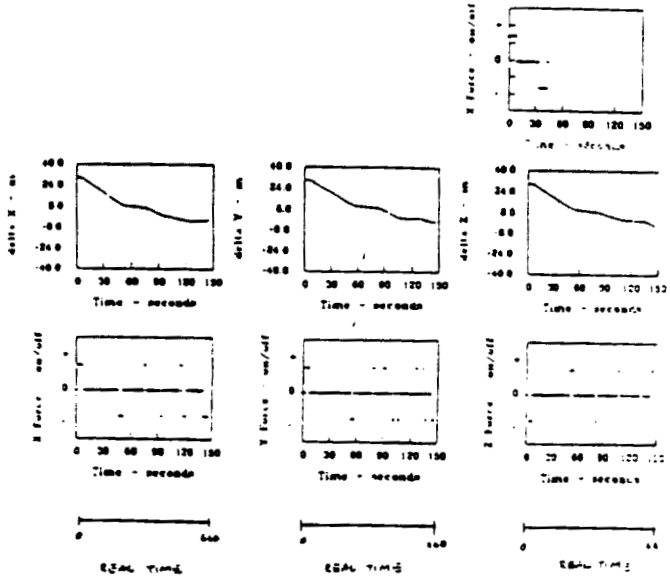


Figure 7 Bottom

Figure 7

Training by Example. MMV control in three axes showing displacement, forces, and velocity for automatic control. Note Improvement after training.

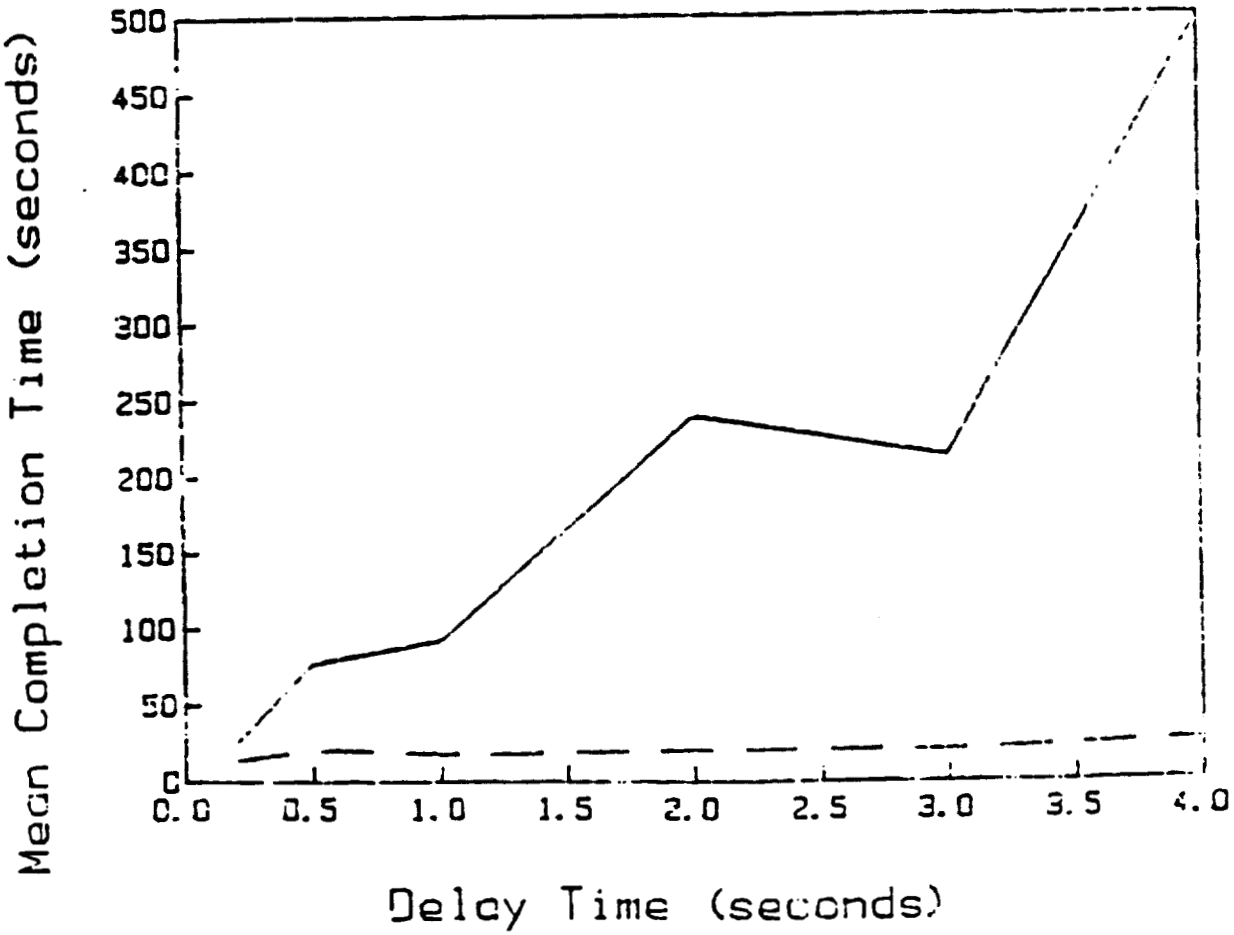
COMMUNICATIONS DELAY AND PREVIEW

Communication delay is a significant constraint in human performance in controlling a remote manipulator. It has been shown (Sheridan et al, 1964, 1966; Tomizuka and Whitney, 1976) that preview information can be used to improve performance. Stark et al (1986) demonstrated that preview can significantly reduce error in tracking experiments with imposed delay.

Experiments were performed to investigate whether a preview display could improve performance in pick-and-place tasks with delay. A single bright diamond-shaped cursor was added to the display to represent current joystick position. This was a perfect prediction of what the end effector position would be after the delay interval. Thus, the task was the same as if there were no delay, except that the HO had to wait one delay period for confirmation that a target had been touched or correctly placed (in the non-previewed display, the target letter was doubled when picked up, and became single again when placed in the correct box).

Performance affected by

Preview improved performance at delays up to 4 seconds so that it was almost as good as for a small delay of 0 2 seconds (Figure 8). While task completion time in the delayed condition increased greatly with delay, there was only a small increase in the preview case. This is because the H.O. must compensate for delays by using a "move-and-wait" strategy, making a joystick



Performance Affected by Delays and by Preview Control Mode.

Note severe adverse influence delay and beneficial effect of preview control in this pick-and-place task..

Figure 8

movement and waiting to see the resultant end effector movement. In the preview case, this strategy is only necessary when very close to the target or box to wait for confirmation that the goal has indeed been touched.

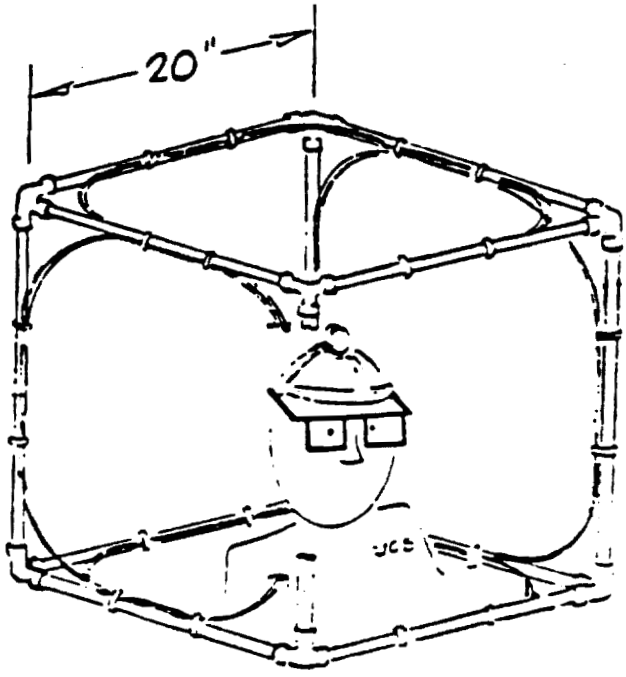
HELMET MOUNTED DISPLAYED DESIGN

Motivation

The motivation of the HMD system is to provide the human operator with a telepresence feeling that he is actually in the remote site and controls the telemanipulator directly. The HMD system detects the human operator's head motion, and controls the remote stereo camera accordingly. In our current system, the remote telemanipulation task environment is simulated and the pictures for the display are generated by the computer.

Head Orientation Sensors

A two-axis magnetic Helmholtz coil arrangement was used as a head orientation sensing device, to detect horizontal and vertical head rotations (Figure 9). By assuming that the pan and tilt angles of a remote stereo camera are controlled in accordance with the horizontal and vertical head rotations, respectively, the computer generated the corresponding stereo picture for the HMD. The head orientation sensing device is composed of a search (sensing) coil mounted on or beneath the helmet and two pairs of field coils fixed with respect to human operator's control station. The right-left pair of the field coil generates the horizontal magnetic flux of a 50 KHz



Head Orientation Sensor Device.

Figure 9

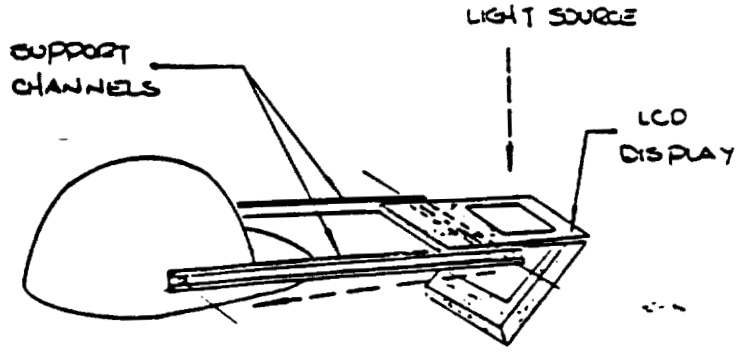
square wave. The up-down pair of the field coil generates the vertical magnetic flux of a 75 KHz square wave. The search coil detects the induced magnetic flux, which is amplified and separated into 50 and 75 KHz components. The magnitude of each frequency component depends upon the orientation of the search coil with respect to the corresponding field coil (Duffy, 1985).

LCD Display

An early configuration of the HMD had a flat-panel LCD (liquid crystal display) screen (a commercially available portable LCD television) mounted on the helmet for the display (Figure 10). However, the picture quality of the LCD screen was poor due not only to low resolution but also to poor contrast.

CRT Display

A new design of the HMD that we currently have, mounts a pair of Sony viewfinders (Model VF-208) on the helmet (Figure 5). Each viewfinder has a 1-inch CRT (cathode ray tube) screen and converging lens through which the human operator views the CRT screen. The computer-generated stereo picture pair (stereogram) is displayed on the CRT screens; one for the left eye and the other for the right. The converging lens forms the virtual image of the stereogram behind the actual display screen. When the CRT screen is 4.2 cm



Early HMD Design with LCD Screen.

Figure 10

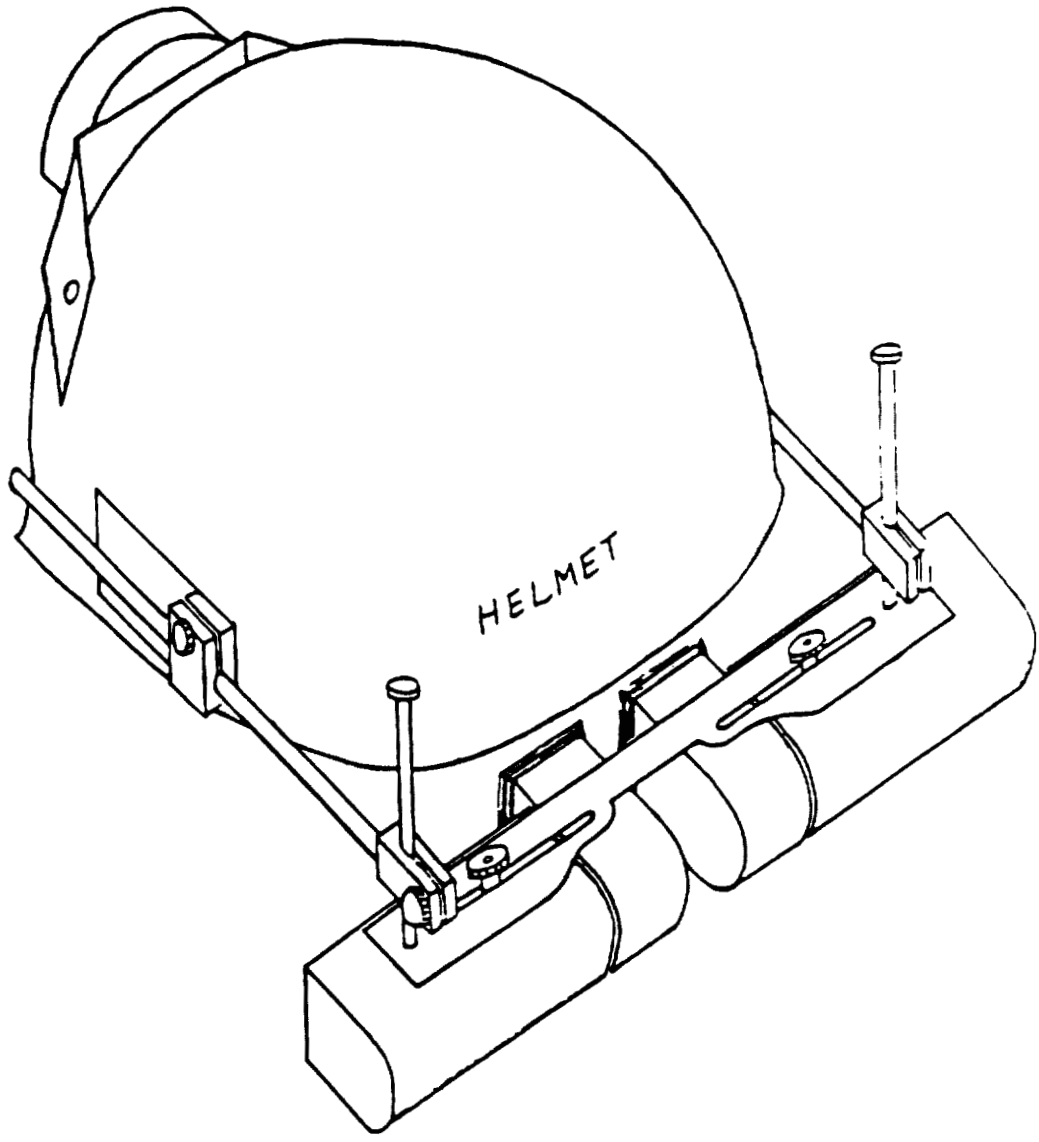
apart fros whose focal length is 5 cm, the virtual image of the CRT screen ist 25 cm apart from the lens with an image magnification of 6. Thus, CRT screen appears to be a 6-inch screen to the viewer. At appropriatrical and optical conditions, the right and left images overlay, aeople can fuse the two images into a single three-dimeimage. The stereoscopic display formulas used to generate the stereo the helmet mounted display are described in references (Kim, et ated 1985).

Mechanical

Fes of freedom were provided for the mechanical adjustment of the posi orientation of each viewfinder, allowing three orthogonal slidings artations (Figure 11). A 1 lb. counterweight was attached to the back ofmet for counterbalancing.

SUMMARY

Theexperiments enabled our Telerobotic Unit at the University of Californley, to explore in a number of research directions. The HMD directiow been greatly extended and is a major focus in our laboratory. other hand, the homeomorphic controller did not seem to be a productive to continue because of the adaptability of the H.O. to many configuof control. Also, our interest in supervisory and other high level cis leading us away from direct manual control. The



Current HMD Design. CRT screens provide stereo vision, with high resolution. Slave stereo camera could provide distant scene information in accordance with helmet pan and tilt; however, we have so far used simulated stereoscopic scenes.

Figure 11

ıthusiastic and felt the course stimulated their creativity and
ırtunity for them to engage in relatively unstructured
--- a good model for subsequent thesis research.

REFERENCES

Argonne National Laboratory, Manipulator Systems for Space Applications, Technical Report, Argonne, 1967.

Bejczy, A. K., "Sensors, Controls, and Man-Machine Interface for Advanced Teleoperation," Science, Vol. 208, No. 4450: 1327-1335, 1980.

Duffy, M. K., "A Head Monitor System Using The Search Coil Method," Master's Thesis, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 1985.

Ellis, S. R., M. Tyler, W. S. Kim, M. W. McGreevy and L. Stark, "Visual Enhancements for Perspective Displays: Perspective Parameters,: IEEE 1985 Proceedings of the Int. Conf. on System, Man, and Cybernetics, pp. 815-818, 1985.

Flatau, C.R., "The Manipulator as A Means of Extending Our Dexterous Capabilities to Larger and Smaller Scales," Proceedings of 21st Conference on Remote Systems Technology: 47-50, 1973.

Heer, E., Remotely Manned Systems: Exploration and Operation in Space, California Institute of Technology, 1973.

Johnsen, E. G. and W. R. Corliss, Human Factors Applications in Teleoperator Design and Operations, Wiley Interscience, 1971.

Jordan, T., "The Simulated Manned Maneuvering Unit and Pursuit Experiments," Master's Thesis, Department of Mechanical Engineering, University of California, Berkeley, 1985.

Kim, W. S., S. R. Ellis, M. Tyler and L. Stark, "Visual Enhancements for Telerobotics," IEEE 1985 Proceedings of the Int. Conf. on System, Man, and Cybernetics, pp. 807-811, 1985.

Kim, W. S., F. Tendick, and L. Stark, "Visual Enhancement in Pick-and-Place Tasks: Human Operator's Controlling A Simulated Cylindrical Manipulator," submitted to the IEEE J. of Robotics and Automation.

Kim, W. S., S. R. Ellis, M. Tyler, B. Hannaford and L. Stark, "A Quantitative Evaluation of Perspective and Stereoscopic Displays in Three-Axis Manual Tracking Tasks," submitted to the IEEE Trans. on System, Man, and Cybernetics.

McRuer, D., D. Graham, E. Krendel and W. Reisener, "Human Pilot Dynamics in Compensatory Systems: Theory, Models, and Experiments with Controlled Elements and Forcing Function Variations," U.S. Air Force AFFDL-TR-65-15, 1965.

Sheridan, T. B., "Three models of preview control," IEEE Trans. Human Factors in Electronics, HFE-7: 91-102 (1966).

Sheridan, T. B., M. H. Merel, et al, "Some predictive aspects of the human controller," in Progress in Astronomics and Aeronautics, Vol. 13, Academic Press Inc., New York, 1964.

Stark, L., W. S. Kim, F. Tendick, et al., "Telerobotics: Display, Control and Communication Problems," in press in the IEEE J. of Robotics and Automation, 1986.

Stark, L., Neurological Control Systems: Studies in Bioengineering, Plenum Press, 1968.

Tomizuka, M. and D. E. Whitney, "The Human Operator in Preview Tracking: An Experiment and Its Modeling via Optimal Control," Trans. ASME J. of Dynamic Systems, Measurements, and Control, Vol, 98: 407-413, 1976.

Young, L. R., "On Adaptive Manual Control," IEEE Trans. Man Mach. Syst., Vol. MMS-10, No. 4: 292-331, 1969.

N89-10089

OPEN CONTROL/DISPLAY SYSTEM FOR A TELEROBOTICS WORK STATION

Saul Keslowitz, PhD
Grumman Corporation
Bethpage, NY 11714

ABSTRACT

Grumman Space System's Controls and Displays (C&D) Laboratory has developed a working Advanced Space Cockpit that integrates advanced control and display devices into a state-of-the-art multimicroprocessor hardware configuration, using "window" graphics and running under an object-oriented, multitasking real-time operating system environment. This Open Control/Display System supports the idea that the operator should be able to interactively monitor, select, control, and display information about many payloads aboard the Space Station using unique sets of I/O devices with a single, software-reconfigurable workstation. This is done while maintaining system consistency, yet the system is completely open to accept new additions and advances in hardware and software.

As we see it, operators aboard the Space Station will be required to monitor and maintain all of the Station's subsystems from a single, small, shared work area. We have designed displays that provide consistency between operations. The information displayed to the operator is easily accessible, understandable, and useful. There is no need to train an operator on each specific display since each display page follows a consistent format. Typical displays include graphic aids such as docking reticles, force/moment and motor torque plots, joint angle parameter displays, and graphic handbook information such as schematics. We have demonstrated the usefulness of infrared touch-sensitive color graphic switches and of voice recognition systems, in addition to conventional dedicated input devices.

The hardware/software architecture is configured so that when new technologies become available, it will be easy to modify the system to handle the new requirements. The system was made hardware-independent by applying the concept of "software layering." This entailed writing generic software modules that can communicate with hardware-specific software "device drivers" in a consistent manner. The concept of "object-oriented programming" makes a large complicated software task manageable by breaking it down into tasks or "objects" that communicate by passing uniformly defined messages to each other. Applying these modern approaches makes the resulting software manageable, easily modifiable, and transportable. We have made our software "data driven," so that changes in program requirements only require a modification of the data file rather than a program rewrite, thus avoiding a tedious reverification and validation effort.

The Advanced Space Cockpit, linked to Grumman's Hybrid Computing Facility and Large Amplitude Space Simulator (LASS), has been used to test the Open Control/Display System via full-scale simulations of the following tasks: telerobotic truss assembly, RCS and thermal bus servicing, CMG changeout, RMS constrained motion and space constructible radiator assembly, HPA coordinated control, and OMV docking and tumbling satellite retrieval. The proposed man-machine interface standard discussed below has evolved through many iterations of the tasks, and is based on feedback from NASA and Air Force personnel who performed those tasks in the LASS.

1 - INTRODUCTION

ɑ Space Systems' Advanced Space Cockpit Laboratory is used ɹl for investigating advanced state-of-the-art technologies as they ɔ the man-machine interface. With this laboratory, it is possitemonstrate a variety of ways that humans can communicate more ely and naturally with the computer. A principal function of thisl station laboratory is to demonstrate methods of monitoring and cɑg simulated Space Station payloads, remote vehicles, and manipɪystems. The workstation has been developed, as should a typica Station workstation. It has been designed to interface with v any input/output peripheral device, while imposing a minimum in software or hardware redesign. To accommodate the rapidly cl technology in the areas of hardware, software, and artificial ince, it should be possible to easily upgrade the workstation with tɔdern approaches as proven research results become available. ɪg the limitations of dedicated workstations for each payload (possiling weight and requiring more space in the Space Station) makes It is possible to have a generic workstation with a windowingility so that the operator can control and display anything from aɪstation. This also makes sense in terms of reliability.

2 - MAN-MACHINE INTERFACE

Cuirements should be imposed on future workstations so that it is ead more natural for the operator to communicate with all aspects ohisticated computer-based system. Some guidelines in the followiɪs should be considered:

- devices
- ɪy formats
- ɔnality among workstations
- rotocol
- f color
- shape and texture
- ology trends
- e of automation.

ave addressed these issues and, based on several iterations of changve have developed our own requirements based on feedback from h factors tests, Air Force, and NASA personnel suggestions[5].

2.1 STATION FEATURES

ad of having a separate workstation to monitor and control Spaceion payloads, which adds weight and uses more space, the AdvarSpace Cockpit was designed so that almost all C&D functions can bntrolled from any console and that interaction with the user is consis between workstations. Using infrared touchable color graphic switcleach workstation can be reconfigured by simply redefining the touchches displayed on the workstation. At any given time only the pertirswitches are displayed. The software is set up so that the displaompts the user and guides him through a diagnostic scenario when ut-of-tolerance condition or failure occurs. The master alarm switchhe upper right hand corner turns red, blinks, and an audible tone ls until this switch is touched to signify acknowledgement of the pn.

al functions available to our workstations are clearing and callink the graphics in a zone on the screen. This is useful if the graphbstruct the image of the live target. Other functions on the bottothe screen are touch switch control of brightness and contrast of thV image, calling up docking reticles, parameter displays, and real-lots in designated zones.

2.2 H INPUT

rincipal input device is an infrared touch bezel. It allows a fast,, and natural way for the user to interact with a computer. The reasons for selecting the touch bezel are as follows[6]:

special skills required, such as typing

u touch what you see

No coordinated hand movement is necessary such as with a trackball or joystick

All options are available on the screen

st response

- Faster than entering commands on a keyboard or other input devices
- Can be programmed for immediate user feedback (aural or visual)

- Only valid options displayed
  - Computer guides user step by step
  - Reduce operator search time
- Touch targets can be located at different points on display
  - Can touch desired element in schematic (electrical, thermal, etc)
- Touch system displays can be interesting, illustrated and visually pleasing as well as functional
- Input errors are significantly reduced, since only valid selections are on screen
- User training time reduced as system becomes more complex
  - Not necessary to learn a computer language or key sequences to memorize

- It does not require desk space in work area
  - Large number of options can be made available by redefining touch points.

2.3 VOICE INPUT

An attractive input device is a voice recognition system. This allows hands-free advantage along with little operation training required. However, voice recognition systems presently suffer from limited recognition success along with limitations such as speaker dependence and finite number of words. As a result, voice control should not be used to control critical functions. We have successfully used it for voice control of cameras. However, statistical and expert system enhancements using syntactic and semantic considerations in increasing recognition accuracy will be studied in the future.

2.4 DISPLAY FORMATS

Each cockpit display has a consistent screen format developed by Grumman so that the operator knowns exactly where to look, no matter

at which ɯtion he is working. The displays are partitioned (Fig. 1) such thɨtouch switches on the lower portion of the display control displaɣtions, the left switches control mission functions, the upper portɨplays status information, and below that is a message area. The right corner of the display is reserved for caution and warning infɷon, and the right side is for real-time plots and parameter displɨ The center of the screen displays live video camera images withɨability to select graphic aids (such docking reticles, force displaɒ overlay the live video. Figure 2 is a picture of an RMS displayɹen live video is not required, the center and left side of the displaɣ used for windows.

2.5 WINDOW

To monithe status of various components and perform tasks on board the SpStation, we have developed a windowing system in which the operator ɔorms an orderly database search to retrieve any information he wa The windowing system also allows him to perform almost any tadesired by interacting with the touch sensitive graphic switches in thɯdow.

The use windowing has proven to be a powerful display capability in termɒf improving the man-machine interface. Multiple windows displayedɧ a CRT are basically equivalent to multiple "software CRTs". That ɨ instead of having many bulky CRTs displaying various types of informɷon, one CRT can display the same information with multiple windoɯ each corresponding to a CRT. In addition, a given window of displɨ information can be "opened" or "closed", interactively. Typically, withɳultiple windows on a CRT, one window can display real-time informɑion such as plots of a selected pump's pressure and temperature, anɔher can display thermal control system schematics, and another EVA staus, all simultaneously and interactively. Figure 3 shows a typical eɤample of multiple windows displayed simultaneously.

2.6 TOUCH SWITCH STATES

Since "touch switch" switches are different from conventional switches, we have developed consistent representations of switch states

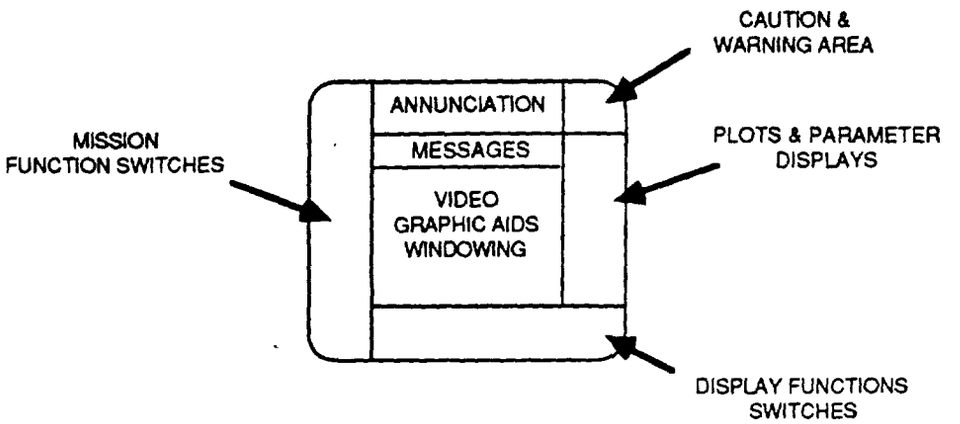
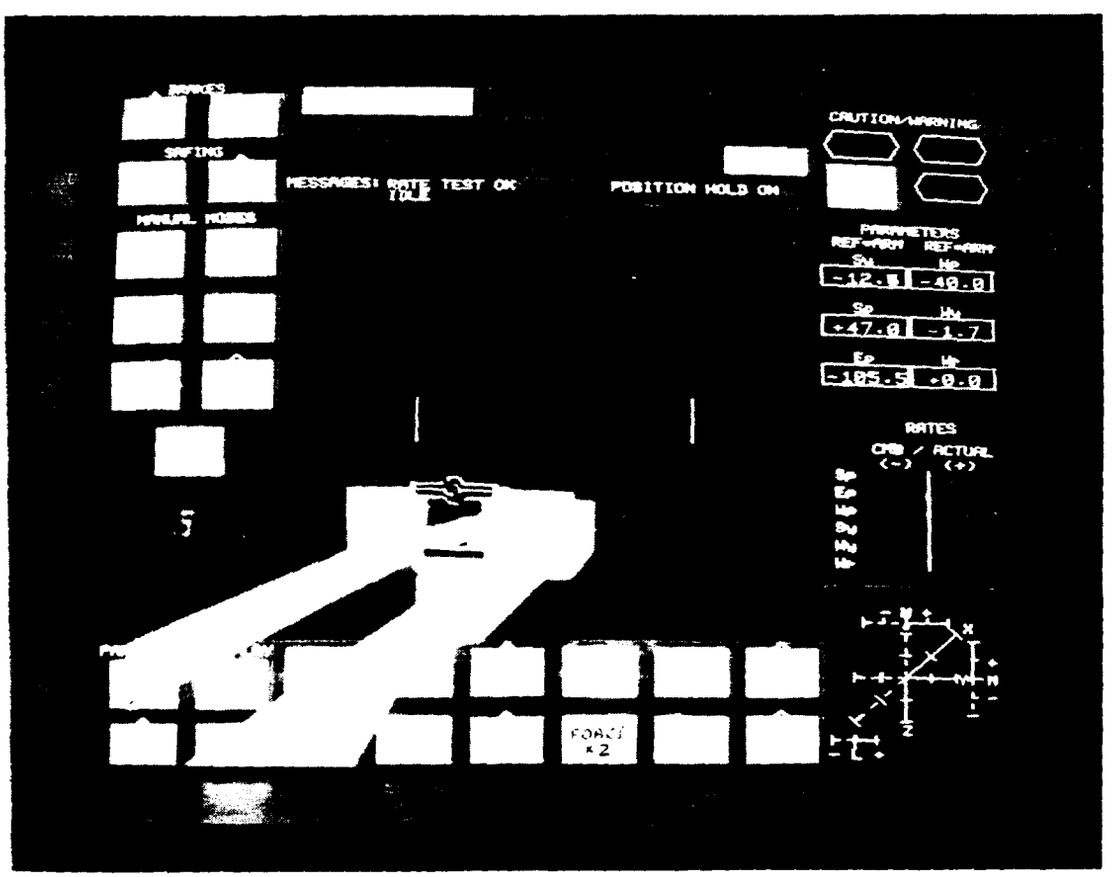


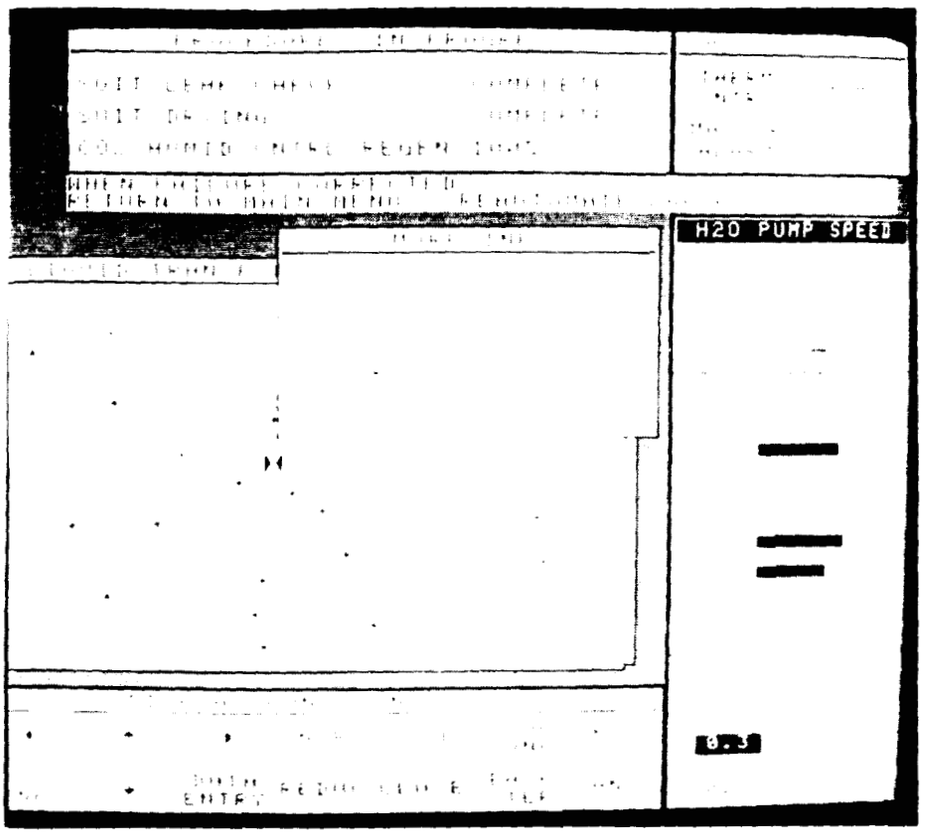
Fig. 1 Basic Display Format



NOTE: STANDARD DISPLAY FORMAT SHOWING THE GRAPHICS OVERLAYING A LIVE CCTV CAMERA IMAGE FROM THE CAMERA MOUNTED ON THE LASS DURING A SPACE CONSTRUCTIBLE RADIATOR SIMULATION. WHEN THE RADIATOR MAKES CONTACT WITH THE MANIFOLD WITH THE HELP OF THE DOCKING RETICLE, THE OPERATOR PRESSES FORCE X 2 AND THE FORCE/MOMENT DISPLAY IN THE LOWER RIGHT CORNER IS MAGNIFIED AND DISPLAYED IN THE CENTER OF THE SCREEN, OVERLAYING CAMERA IMAGE OF THE CONTACT AREA.

R87-3538-008G

Fig. 2 RMS Display



IDARD DISPLAY FORMAT SIMULTANEOUSLY SHOWS PROCEDURES IN PROGRESS STA-
ROLLABLE WINDOW, A MESSAGE IN ITS DESIGNATED AREA, AN OUT OF TOLERANCE
D SPECIFIED BY A REAL TIME H_2O PUMP SPEED PLOT, A MORE INFO WINDOW, A RED
JMP IN THE LIQUID TRANSPORT CIRCUIT SCHEMATIC AND HIGHLIGHTED EVA IN THE
ARNING AREA.

Fig. 3 Interactive Windowing Display

by)lor, texture, and shape. A normal switch available for selapproximately a 1.25 in. x 0.75 in. light blue filled rectangle witction written or abbreviated inside it. A brick face unfilled swesents a function that is unavailable at present. When a ligwitch is touched, it turns yellow and an unfilled arch is plap of it. This represents an intermediate state, signifying thaD computer is waiting for further operator action or a confirmail that the device, mode, or function selected was activated. For, when "ACCEL" mode is selected on an OMV display, the swis yellow, and the C&D computer causes a message to be tra into space to the OMV computer and then waits for an actignal. When the signal is received, the switch turns green anc green arch is drawn on top of the switch, signifying the actate. The arches are used to distinguish between the states of hes if color is not available.

A short audible tone occurs each time the screen is touched, indicating to the operator that his touch registered. The possibility exists that a wrong switch was selected. Therefore, in order to prevent false activation of some switches, "ENTER" and "CANCEL" switches are available. Usually, these switches are used in conjunction with the mission function switches. For example, in the RMS display, if "SINGLE" is touched it turns yellow as do "ENTER" and "CANCEL", signifying that a choice should be made.

3 - HARDWARE/SOFTWARE OBJECTIVES

The hardware/software architecture of the workstation was originally configured with the following objectives:

- Must be powerful and general enough to accommodate all potential real-time requirements that it may have to satisfy with respect to controlling and monitoring payloads considering the human interface point of view
- Drive a number of color graphics monitors and plasma displays, perform camera control functions, manage real-time communications and color displays (such as docking, rendezvous, berth curves, plots), and accept voice, touch bezel, switches, and hand controller inputs
- Use state-of-the-art hardware and software, and use modern concepts
- Easily expandable, so as to not become obsolete
- Should use the "open system" concept and not be tied to one vendor
- Demonstrate a highly reliable system
- Hardware and operating system software should be characteristic of the flyable system to minimize redesign.

The workstation system architecture put together in this laboratory was chosen with these goals in mind. Our laboratory is built around a powerful multimicroprocessor configuration which uses a real-time multitasking operating system. This flexible architecture, which is representative of most modern advanced workstations, allows tasks to run in

real time and with each microprocessor performing its particular function simultaneously. This greatly increases the performance from a single processor system.

4 - OPEN SYSTEM CONCEPT

With rapidly advancing technology over many generations of Space Station existence, and in the face of increasing and changing requirements, it becomes necessary to design an initial workstation that can be upgraded, expanded, or reconfigured quickly and easily, in order to prevent obsolescence. Therefore, when developing a workstation, consideration should be given to choosing an open system architecture which is easily expandable, with hardware/software products available from many manufacturers.

The C&D laboratory configuration consists of three Intel 80286-based multibus microcomputers running under the iRMX-86 real-time operating system. The 80286 microprocessor was chosen because it was the most powerful available at the time. A multibus system was selected because it is an IEEE standard and over 2000 board level products from over 200 manufacturers support it. As a result, we are not locked into one vendor, our system will not become obsolete, and is easily expandable. Some multibus cards perform functions such as D/A, A/D conversion, color graphics, and serial communication.

5 - HARDWARE DESCRIPTION

Figure 4 shows a block diagram of the Advanced Space Cockpit and its associated computer control room (C&D laboratory) driving it. The C&D laboratory consists of three Intel 80286-based multibus computers, each with 30 megabyte hard drives, a 1 megabyte floppy disk drive with at least 512K of Random Access Memory (RAM), and a terminal. Each computer runs under the iRMX-86 real-time multitasking operating system.

Messages are passed between computers over an RS-232 link, which will be upgraded to Intel's Bitbus. The computers are populated with

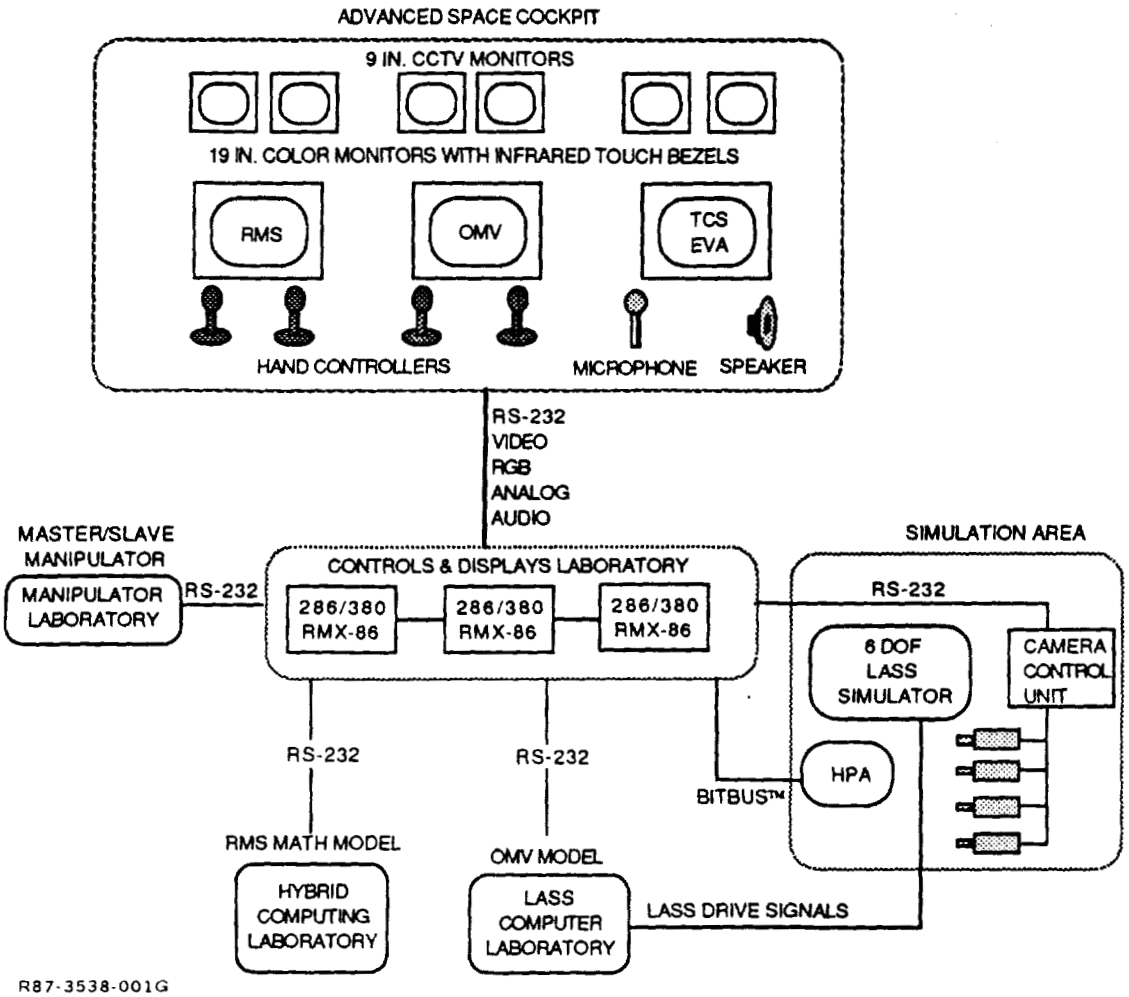


Fig. 4 Advanced Space Cockpit Block Diagram

multibus-compatible RS-232 cards, high and medium resolution Matrox color graphics cards, voice recognition/synthesis cards, discrete I/O, analog/digital converter cards, and Bitbus cards. The medium resolution graphics systems are capable of overlaying color graphics and text over a real-time CCTV camera image. This is how we overlay a docking reticle over a live camera image of a target during an OMV docking simulation. In addition, some of the cards have their own microprocessor on board, allowing concurrent processing within a microcomputer itself.

The main input device is a Carroll infrared touch bezel mounted on all our 19 in. color monitors. Graphic touch switches are used for input selection capability. Three workstations are presently active in the cockpit, however it can accommodate at least eight workstations. Figure 5 shows two of the workstations.

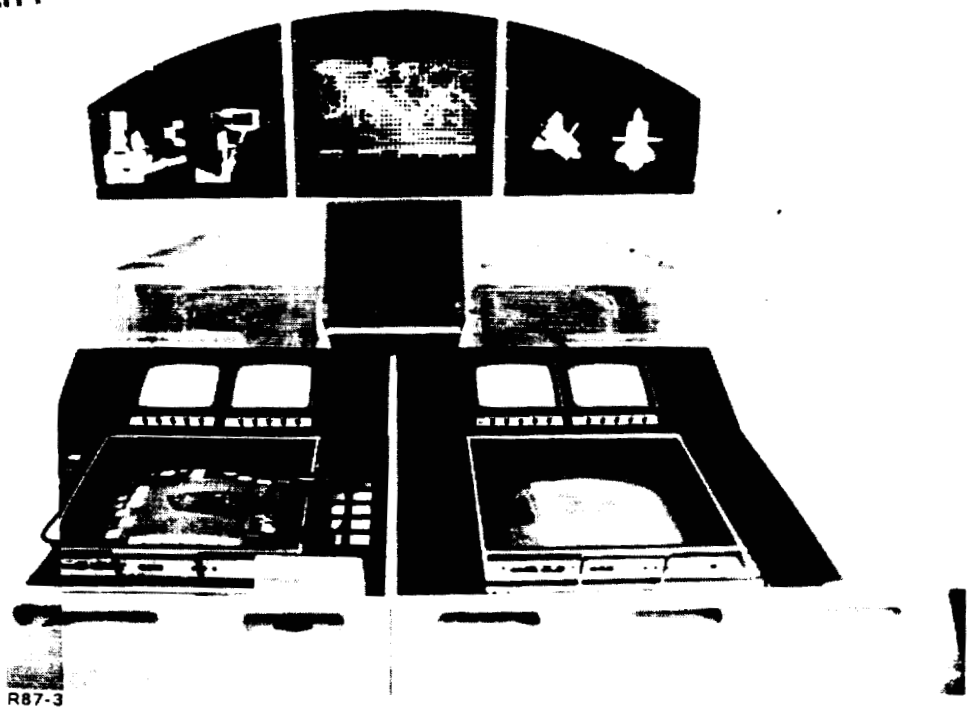


Fig. 5 Mockup of Two Bays of Prototype Workstation

neric LED dot matrix switches and plasma displays with touchare also in the cockpit. Three and six degree-of-freedom hand ers are available for OMV and RMS simulations. Headsets with nes are available for communications with the control room and oratories involved with the simulations, in addition to suppoice control of the cameras and displays using a Votan voice recogstem.

dback is achieved by using the SpeechPlus' Prose 2000 voice s system or human computer compressed speech available from th system. Tones are used for caution, warning, and failed condit

6 - SOFTWARE DESIGN PHILOSOPHY

Ooach is to develop an easily reconfigurable workstation that has a software configuration built around the iRMX-86 operating system s the following principle of good design.

"A system should be built with a minimum set of unchangeable parts; those parts should be as general as possible; and all parts of the system should be held in a uniform framework."[1]

Any part of the system that is not easy to change, is not sufficiently general, or works differently from the others will require additional effort and will impede development. Also, the software models used for this "rapid prototyping" configuration should be based on mechanisms that are familiar, easily understood, and have desirable characteristics such as simplicity, speedy development, extendibility, and reusability.

As the field of software engineering develops, various architectural concepts are emerging for developing portable reusable and maintainable software. These include the "virtual machine"[2] concept which protects the software from hardware changes and "information hiding" throughout the layering of software functions (Fig. 6). Also, "Anthropomorphic Programming is a proven technique for building systems that work using a multitask structuring and message passing scheme to simplify the analysis of complex systems".[3] Our approach keeps these modern concepts in mind.

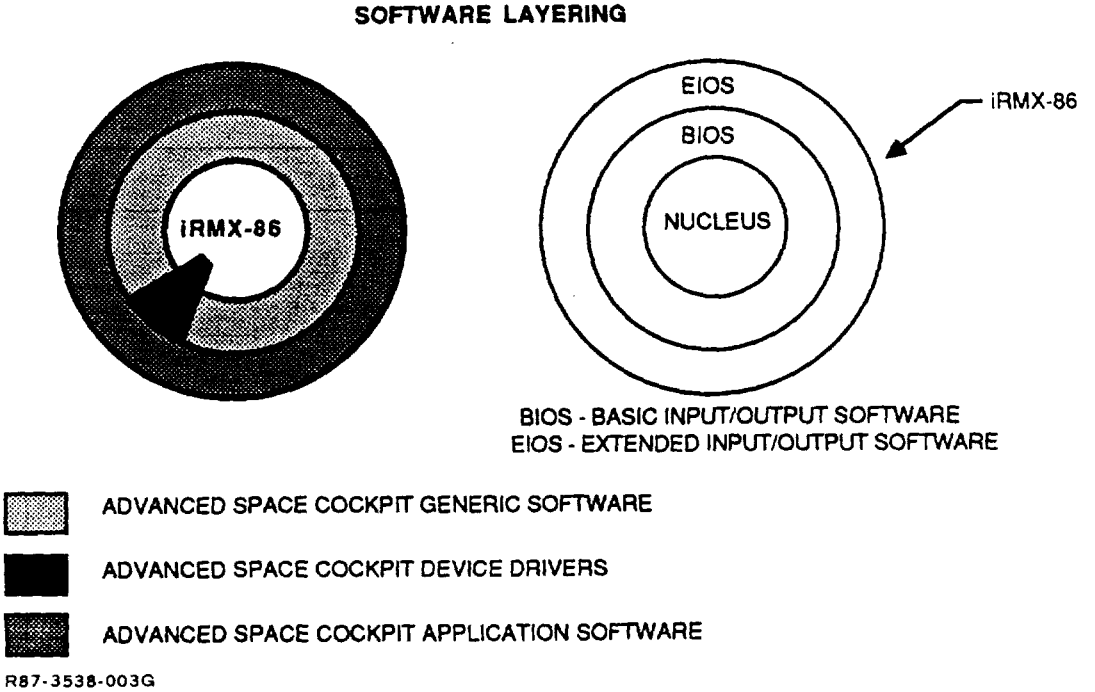
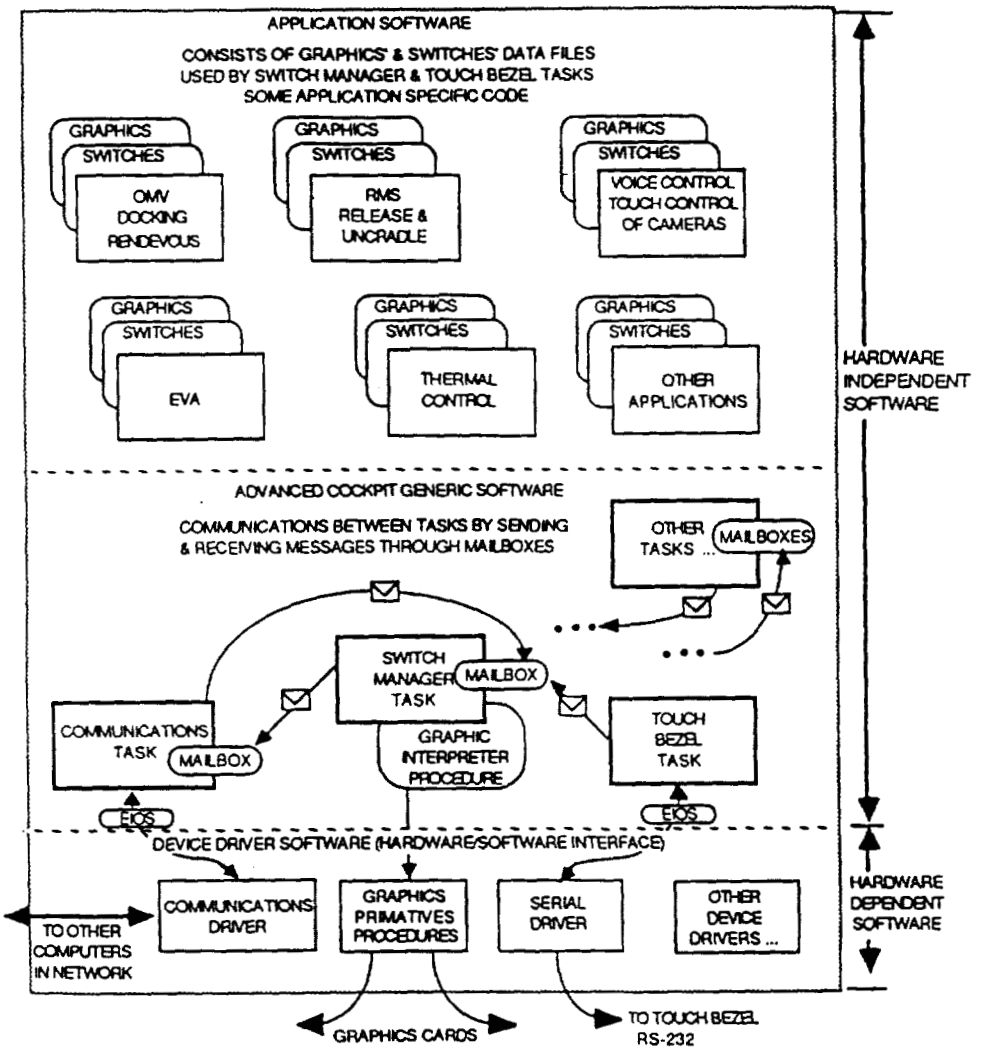


Fig. 6 Outer Layers Are Built On Inner Layers

Figure 7 shows a typical software architecture that exists on each microcomputer. Based on our design of a generic workstation of this type, we have defined the requirements of each iRMX-86 task in addition to a specific message format that each module expects to see and produce. As a result of this general architecture, additional modules can be easily added as required with little or no impact on the other modules. They can be written independently by separate programmers by simply defining the task's functions and the messages it wants to receive and send. Later, modules can be optimized internally, if nec-



NOTE: ADVANCED SPACE COCKPIT GENERIC SOFTWARE IS DATA DRIVEN, HARDWARE INDEPENDENT AND RUNS IN A MULTITASKING ENVIRONMENT. WITH THIS ARCHITECTURE THE SOFTWARE/HARDWARE IS EASILY RECONFIGURABLE AND EXPANDABLE.

R87-3538-004G

Fig. 7 Typical Software Architecture On Each Microcomputer In Network

essary, as long as the external specifications are unchanged. Note that this general specification can be applied to any workstation that uses a multitasking operating system.

7 - iRMX-86 MULTITASKING OPERATING SYSTEM

The IRMX-86 operating system[4] used in our workstation has capabilities representative of any real-time operating system. The multitasking capability allows the programmer to factor a problem into simple processes or tasks. It makes available the usual assortment of object types to the software developer. It allows the developer to create/delete tasks, mailboxes, semaphores, and segments. Intertask communication is performed by sending and receiving messages to and from a given task's mailbox. Although it appears that all the tasks are running concurrently, in reality, with only one host 80286 microprocessor in a computer, only one task can be running at a time. If a task is not running then its state is put on either the ready, asleep, or suspended queue (Fig. 8) until the present task stops running, and then the operating system starts running the next available task on the ready queue.

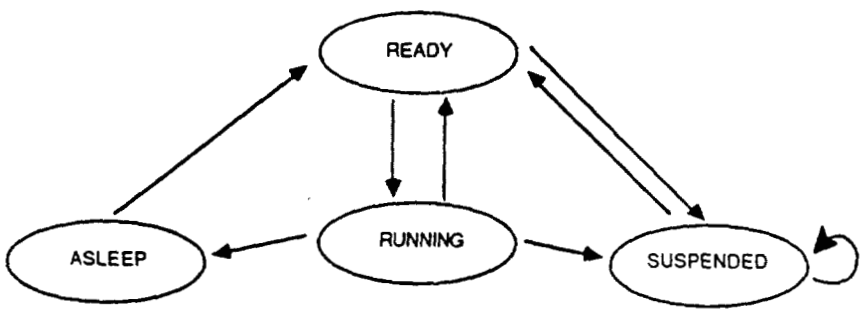


Fig. 8 Task State Transition Diagram

In Fig. 6, it is seen that iRMX-86 uses the software layering concept. The outer software layers use the inner ones. The nucleus makes the object types mentioned above available to the outer layers. The Basic Input Output System (BIOS) layer allows the outer layer to communicate with any peripheral input/output device in a uniform manner. This assumes the hardware-specific device driver was written for the device using Intel's prescribed format. Devices can therefore be

opened, closed, read from, and written to. The next higher layer, the EIOS, makes it even easier to communicate with the I/O devices. Although there are more outer layers available, our software is only built on those mentioned.

8 - DESCRIPTION OF SOFTWARE

As mentioned above, the laboratory is to be used as "rapid prototyping" tool for investigating the workstation design with respect to the man-machine interface. With this in mind, most of the application software consists of switches and graphics data files. This is consistent with our goals of minimizing the amount of new code we have to write for a new application or modification to an existing one.

Figure 7 shows a typical software configuration that allows rapid reconfiguring of the hardware and software. Each 80286-based microcomputer running under iRMX-86 can be considered a processor node linked to each other over a communication network (presently RS-232 and later upgraded to Intel's Bitbus).

When the touch-sensitive display is touched, the position of the touch point is sent over the RS-232 link to the host microcomputer. The touch bezel task receives this position, converts it to screen (pixel) coordinates, consults the displayed switches file, and determines which switch is touched. It then sends a message identifying the switch to the switch manager task's mailbox. When the switch manager gets the message it consults the switches file, finds the switch's data structure, and executes the prescribed action to be taken based on the selected state machine's outputs, as described below.

This software configuration is advantageous from the reliability and fault-tolerance point of view. If the same or similar hardware exists on two or more systems and if it fails on one, it is possible to still perform the function if a failure detection task routed the messages to the other system. Also, if a task requires a resource (hardware) that is busy or is only on another system, then a task manager can assign it to that resource on another system, if it exists.

Our configuration is hardware-independent. Each peripheral of the same type has associated with it a set of well-defined primitives and, similarly, each task has messages that it understands. Whenever a new peripheral is interfaced to our workstation, we simply write a software device driver for it. This device driver is the software to hardware interface that performs the hardware functions specified by the primitives. For example, all graphics systems in our workstation recognize primitives such as DrawLine, DrawCircle, etc. With this capability, any application display can be run on any monitor. Another result of this configuration is that it is easy to control any function from any input device (touch bezel, voice control, etc) by causing the same messages to be generated when the inputs from different devices are considered the same.

9 - FINITE STATE MACHINE CONCEPT

Common to all workstations are ways of inputting information to control functions such as selecting modes of a system and activation or deactivation of a function. In order to simplify the management of a large number of input touch switch selections, it is necessary to apply the same general scheme for all switches, rather than program the operation of each separately. In addition, it is advantageous to keep information about the switches in a data file rather than in the code. This makes it easier to make modifications to a given switch without recompiling the program. Typical fields in our touch switch data structure are described as follows:

```
Type
   TouchSwitch=Record
      PositionOnScreen:Point;
      SwitchLength, SwitchWidth:Integer;
      NameInSwitch:String;
      SwitchDisplayed:Boolean;
      PresentState:Integer;
      StateMachineToUse:Integer;
   End;
```

With this information, if a switch is touched, then the software searches the switches file, finds the switch that was touched, and then uses the StateMachineToUse field to determine what action to take. Each switch is modelled as a state machine, which is an extremely powerful and useful mathematical concept having roots in autonoma theory. This finite state automation model describes the behavior of a system as follows: the next state of a system is determined by the present state and input; the output of a system is determined by the present state and input. Figure 9 shows how this model can be graphically represented by a state diagram and by a matrix when applying it to the "ACCEL" touch switch on the OMV display. Each one of the touch

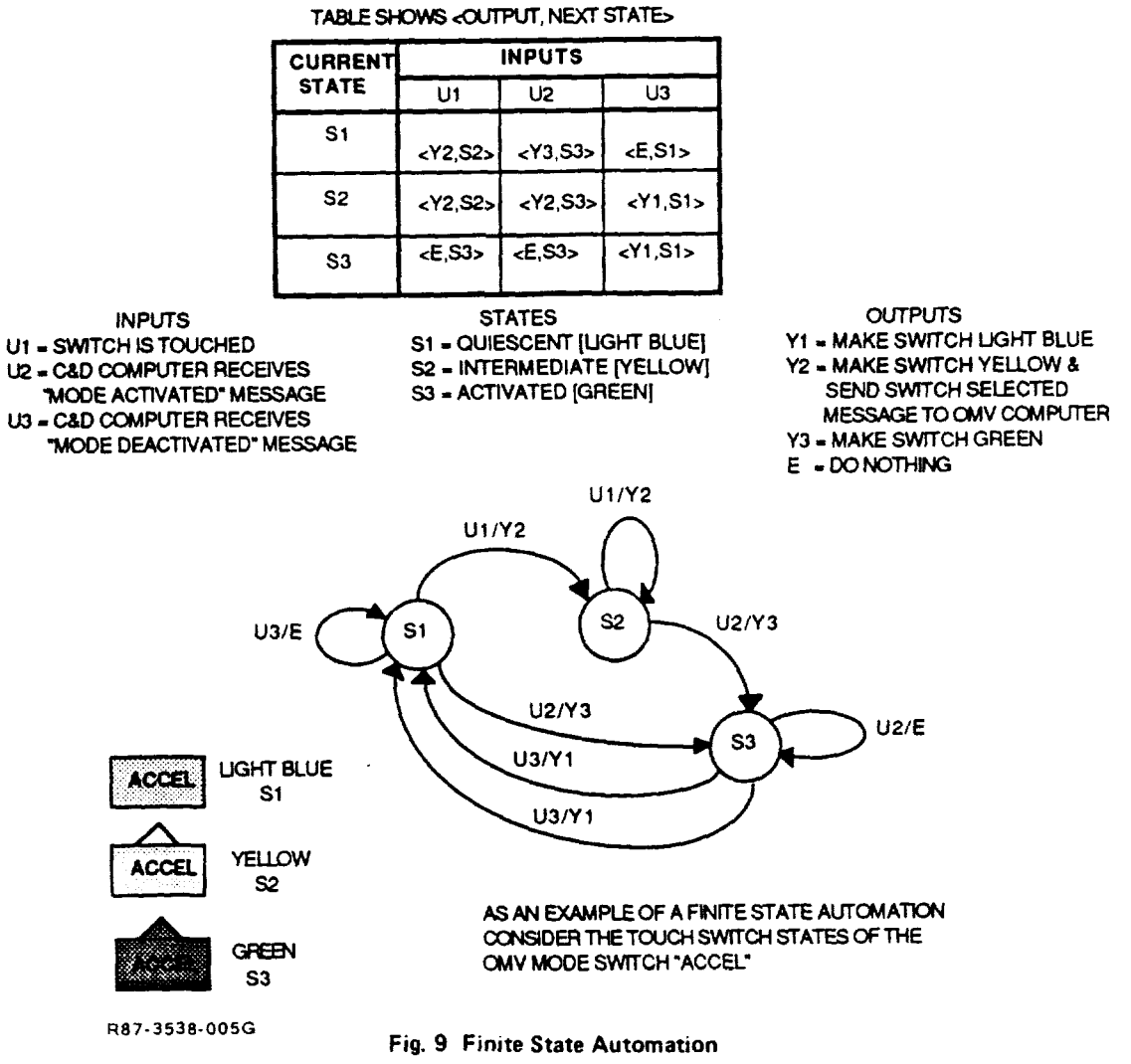


Fig. 9 Finite State Automation

switches has its own state machine associated with it. Typical state machine outputs are:

1. Load and display a switches file
2. Display graphics file
3. Set a switch to a color
4. Send a message to a computer
5. Send a message to a task's mailbox
6. Suspend/resume a task.

Outputs associated with windows are open, close, move, scroll, and scale.

10 - GRAPHICS INTERPRETER

Although not all our graphics hardware is the same, we can run all our software on any one of the graphics systems. This is made possible by defining and writing the same low-level graphics primitives (such as MoveCursorTo(x,y); DrawLineTo(x,y); DrawCircle(x,y,r)) for each system. Therefore, graphics applications software just calls these primitives and the pictures can be presented on any one of the graphics systems. In addition, by putting our display list, consisting of a sequence of primitives, in a data file, our software can interpret this list and draw the picture. This is a tremendous advantage, for graphics pictures can be changed immediately by changing the data file.

11 - EXPERT SYSTEMS

This popular subfield of artificial intelligence allows a program to be broken up into a knowledge base and an inference engine. Using schemes such as forward and backward chaining, the inference engine searches the knowledge base in an attempt to satisfy the present goal. This knowledge base contains if/then rules about the subject at hand, which is called the domain of discourse.

We plan to apply expert system schemes in order to assist the operator at the workstation by automating some of the intelligent decisions that the operator normally makes, such as displaying relevant data at

certain times when necessary. Also, we would like to increase the reliability of the voice recognition system by having the expert system consider the semantics of the sequence of voice commands rather than basing an acceptance/rejection decision only on a recognition threshold.

12 - SIMULATION

A number of different RMS and OMV simulations were performed using the LASS facility's 6 degree-of-freedom motion base. The Advanced Space Cockpit was used as the control station for all the simulations. Maneuvering of the simulated OMV and RMS was controlled by hand controllers. In some of the simulation scenarios, two 3 degree of freedom hand controllers were interchanged with a 6 degree of freedom hand controller for comparison purposes. Operations were conducted using views from a CCTV camera mounted on the motion base, in addition to other camera views. All modes were selected using the touch switches. Docking reticles and force/moment displays, overlaying the live camera image, were available in the center of the screen so that the operator never had to move his eyes from that point.

Real dynamical RMS and OMV math models were used to drive the LASS motion base so that it responded like the real system. Computer communications update times were approximately 50 milliseconds, and display update times were about 250 milliseconds.

Other simulation scenarios use the windowing display. Thermal Control and EVA monitoring, checkout, and servicing scenarios were performed. In Fig. 3, a typical situation using a multiple window display shows an out-of-tolerance pump speed in the Liquid Transport Circuit, in the EVA subsystem. Due to consistency of workstations, operators can easily move from OMV to RMS to TCS and finally to EVA without having to relearn the MMI protocol.

Note that although our workstations are used in simulations, based on our software architecture, it is possible, with a minimum software change, to connect the workstations to the hardware used in an actual

system. This can be done by replacing the "simulation tasks" with tasks that communicate with a host computer and real payloads.

13 - CONCLUSIONS

Traditional programs can be viewed as consisting of data and procedures operating on the data. However, modern programming systems using abstract models (such as the concept of "objects", which hides implementation, software layering, or finite state automations) makes the design of a complicated software project easier and more manageable.[7] Use of other popular modern concepts such as windows, expert systems, voice recognition and touch input improves the man-machine interface by decreasing operator training time and allows him to more naturally communicate with the computer. Future workstations should be easily reconfigurable, have a consistent human-machine communication protocol, and be intelligent, so that much of the operator's decision-making becomes automated.

14 - REFERENCES

[1] Daniel Ingalls, "Design Principles Behind Smalltalk", Byte, August 1981, Vol. 6 No. 8

[2] Ramachendra P. Batni, "Software Development Evolves Into Software Engineering", Computer Design, September 1984, Vol. 23, No. 10

[3] K.S. Booth, et al, "Anthropomorphic Programming", University of Waterloo computer Science Dept. monograph CS-82-47

[4] Intel Corporation, "iRMX^{TM} 86 Nucleus Reference Manual"

[5] A.D. D'Amico, "Simulation Test of Man-Machine Interface for Remote Manipulator System Operations in the Space Station", Proceedings of the Conference on Remote Systems and Robotics in Hostile Environments, March 31, 1987

[6] "An Introduction to Touch Technology", Carroll Touch, PO Box 1309, Round Rock, Texas, 78680

[7] R.E. Filman and D.P. Friedman, Coordinated Computing:Tools and Techniques for Distributed Software, McGraw-Hill, 1984

Consolidated Fuel Reprocessing Program

THE IMPLICATIONS OF FORCE REFLECTION FOR TELEOPERATION IN SPACE*

John V. Draper
Human Machine Interfaces, Inc.
1808 Blue Spring Lane
Knoxville, Tennessee 37932

Joseph N. Herndon
Fuel Recycle Division

and

Wendy E. Moore
Instrumentation and Controls Division
Oak Ridge National Laboratory
Post Office Box X
Oak Ridge, Tennessee 37831

ABSTRACT

This paper reviews previous research on teleoperator force feedback and reports results of a testing program which assessed the impact of force reflection on teleoperator task performance. Force reflection is a type of force feedback in which the forces acting on the remote portion of the teleoperator are displayed to the operator by back-driving the master controller. The testing program compared three force reflection levels: 4 to 1 (four units of force on the slave produce one unit of force at the master controller), 1 to 1, and infinity to 1 (no force reflection). Time required to complete tasks, rate of occurrence of errors, the maximum force applied to task components, and variability in forces applied to components during completion of representative remote handling tasks were used as dependent variables. Operators exhibited lower error rates, lower peak forces, and more consistent application of forces using force reflection than they did without it. These data support the hypothesis that force reflection provides useful information for teleoperator users.

The earlier literature and the results of the experiment are discussed in terms of their implications for space-based teleoperator systems. The discussion describes the impact of force reflection on task completion performance and task strategies, as suggested by the literature. It is important to understand the trade-offs involved in using telerobotic systems with and without force reflection. Force-reflecting systems are typically more expensive (in mass, volume, and price per unit), but they reduce mean time to repair and may be safer to use, compared to systems without force reflection.

*Research sponsored by the Office of Facilities, Fuel Cycle, and Test Programs, U.S. Department of Energy, under Contract No. DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

INTRODUCTION

The National Aeronautics and Space Administration (NASA) has embarked on an extensive national project to establish a permanent human-occupied space station in low earth orbit. In order to accomplish this, significant levels of dexterous, human-like handling tasks must be accomplished during orbit. This will include space station construction and planned and unplanned maintenance on space station. In addition, satellite repair and maintenance will be done. To meet the need for sharply increased levels of dexterous handling while decreasing the hours of human extravehicular activity required, NASA plans to utilize telerobotic hardware on the space station. The role of force reflection in these telerobotic systems is an important issue for developing NASA hardware.

The Oak Ridge National Laboratory's (ORNL's) Consolidated Fuel Reprocessing Program (CFRP) is responsible for developing systems for reprocessing nuclear fuel. This effort includes development of advanced systems for remote maintenance of process equipment developed by CFRP. The CFRP emphasis is on teleoperator systems featuring dexterous, force-reflecting servomanipulators, transporters for large-scale movement, television viewing of remote sites, and human-in-the-loop control. Significant research resources have been used by the CFRP to develop and understanding of the implications of force reflection for performance of teleoperators.

Because CFRP systems emphasize human control, the performance of the human operator is important for overall system performance and, in turn, for mean time to repair and plant availability. One important issue in this area is the question of how much sensory information is necessary for efficient performance of maintenance tasks. Monochromatic television seems to be a minimum requirement; enhancements to the system (e.g., color, increased resolution, and/or stereoscopic television) may improve task performance for certain classes of remote handling tasks.[1] Supplementary sensory channels (e.g., hearing or touch) could also be provided.

Force feedback may be one important supplementary sensory channel. Force feedback may be in the form of proportional force feedback or in the form of force-distribution feedback.[2] Force-distribution feedback provides a display of forces which matches the distribution of forces on the manipulator (usually the manipulator end-effector). It gives users a sense of touch, similar to human tactition. It allows perception of shapes and textures in the remote area. Proportional force feedback presents the operator with a display of force which is proportional to forces on the teleoperator. Force reflection is a type of proportional force feedback in which forces applied by the slave (remote) portion of a master/slave teleoperator are displayed to the operator through back-driving the master controller. The user feels forces through the action of the master controller on the teleoperator master handle. Proportional force feedback gives users a sense which is not directly analogous to any single human sense, but combines elements of tactition (touch) with kinesthesia (kinesthesia is the sense related to forces exerted by the limbs and acting on them).

LITERATURE REVIEW

Four studies have made direct comparisons of teleoperator performance with and without force reflection. D. A. Kugath[3] found some evidence for a beneficial effect of force reflection on teleoperator performance (defined as task time and collisions with equipment in the remote area) for simple tasks with a fairly large-scale manipulator (a General Electric CAM 1400 with 12- and 13-ft booms), but, in the author's words, "Not enough data . . . [were] taken to show conclusively that the lack of force feedback was detrimental." For the large manipulator employed, removal of force reflection following completion of a task several times with force reflection led to high rates of operator errors and seemed to cause manipulator instability. Kugath also noticed a change in the style of operation in his subjects when force reflection was removed. Without force reflection, users seemed to execute trajectories stepwise, making a movement and then checking manipulator position before making another input. This was in contrast to continuous motions observed with force reflection and seemed to lead to frequent target overshoots.

J. W. Hill[4] also reported data which seem to favor force reflection (performance was defined as time required to complete tasks), but his force reflection differences are confounded with differences between the manipulators used in the force reflecting and non-force reflecting conditions. In the latter, subjects performed a set of simple tasks with the NASA/Ames Arm, a unilateral system with an exoskeletal master controller and anthropomorphic (elbows-down) stance. In the force-reflecting condition, they used a Central Research Laboratories (CRL) Model H manipulator system, a mechanical master-slave manipulator with a "through-the-wall" stance. These systems seem too different (kinematically and in terms of performance) to be directly comparable, and indeed the author reports that the Ames Arm typically required 20% longer to complete some simple movements.

Hill and J. K. Salisbury, Jr.,[5] performed an experiment that compared a single manipulator system (the French-designed MA-23) with and without force reflection and also found average differences favoring force reflection in the time required to complete tasks. This is the most rigorous study of the topic to date. Unfortunately, the design of this study and the statistical procedures used to analyze its data were flawed. The experiment included only two subjects; they were administered force reflection conditions in reverse order. The sample size leaves the study vulnerable to threats to validity (for a discussion of threats to experimental validity, see ref. 6) from treatment by subject interactions; small experimental groups increase the likelihood that the subjects are not typical of the population as a whole, and they may have an atypical reaction to the conditions of the experiment. The inversion of treatment administration order does not seem adequate protection against treatment by practice interaction. In experiments that make repeated measurements on subjects, the comparison between conditions must be based on a within-subjects difference. In other words, the performance of one subject with force reflection should be compared to his own performance without force reflection. The total difference

should be the sum of these within-subjects differences. When the treatment administration order is reversed, it does not account for the within-subject effect of practice. Averaging across subjects (especially when there are only two) will not remove the effect because individuals learn at different rates and to different final performance levels. Furthermore, the data analyses used in the study were not appropriate. The authors used a simple factorial analysis of variance (ANOVA) model and assumed that all factors in the model were fixed factors. With repeated measurements on the same subjects, this is not appropriate. A model that considered subjects to be a random factor (a within-subjects model) is the correct one in this case. The mean square ratio (or F test) the authors used for the test of the main effect of force reflection had 1 and 504 degrees of freedom (D.F.); it had the mean square for the force reflection effect as its numerator and the mean square for error as its denominator. The appropriate test would use the mean square for the subject by force reflection condition interaction as its denominator, and would have only 1 denominator D.F. The results of the tests used by the authors are uninterpretable, and conclusions based on this study cannot be accepted with any confidence.

In a more recent study, data collected at ORNL in the course of teleoperator system comparisons failed to demonstrate any positive effect of force reflection on the performance of remote handling tasks.[7] The differences between tasks completed with and without force reflection using two different teleoperator systems were not statistically significant, although on average users required longer to complete tasks with force reflection than they did without it. However, the tasks and procedures used in that experiment were not designed to evaluate force reflection and may have been insensitive to its effects. The ORNL experiment compared a wide range of teleoperator systems, and tasks were designed to be simple enough to complete with relatively low-dexterity systems. Force information had no impact on efficiency within these simple tasks.

J. C. Bliss, Hill, and B. M. Wilber[2] studied performance of tasks with force-distribution feedback. These authors did not find significant differences in the rate of task performance with and without force-distribution feedback. However, the quality of performance, both in terms of the number of errors and failed attempts at the task and in terms of the strategy used by operators was different between the force feedback conditions. These authors had subjects perform a set of tasks with and without force feedback, and with varying degrees of occlusion of the camera line of sight. Without the force-distribution feedback, there were no differences in performance when operators had unobstructed views of the task. With increasing occlusion of the camera view, the number of failed attempts to grasp and operate task components increased more rapidly for users without force feedback. In addition, users without force feedback made more attempts to grasp components and were less careful. Users with force feedback tended to position the teleoperator more carefully. These users made fewer attempts per successful completion of the task, but because they were more careful (and more precise) about teleoperator position, their attempts tended to be of longer

duration. Although there were no differences in the mean time to complete the tasks, the quality of performance differed between force-feedback conditions. Without force feedback, users made frequent imprecise attempts to grasp and operate task components; users with force feedback made fewer attempts, and their attempts were more precise and longer in duration.

CONCLUSIONS FROM THE LITERATURE

It is impossible to draw precise conclusions concerning the effect of force on task performance from the studies which compared teleoperator performance with and without it. The results of these studies are uninterpretable because of their methodological inadequacies. However, certain hypotheses may be stated based on observations from this literature and on the characteristics of humans as processors of information. First, information provided by force reflection can be unique, or it can complement information available through other sensory channels. For example, an operator attempting to tighten a bolt to a criterion torque may be able to judge when the bolt reaches this torque by feeling the reflection of resistance to turning. The task can also be done by viewing the dial of a torque wrench. When force reflection provides information that complements operators' television views of the remote area, operators are not as likely to attend to force reflection since humans tend to favor vision over the other senses.[8] Force reflection is most helpful when it displays information that other senses (particularly vision) are unable to provide or when other displays are difficult to interpret. The greatest advantage for force reflection should occur when forces applied to the remote area are important; when task components require guidance or assembly in areas difficult to see with television cameras; and when viewing is degraded by dust, gases, lens browning, or other obscuration.

Second, there appear to be fundamental differences in the strategies employed by operators with and without force reflection. Kugath reports stepwise trajectory inputs without force reflection; Bliss, Hill, and Wilber report different approaches to grasping task components. It seems that operators without force reflection perform tasks more tentatively than they do with force reflection. The ability to detect contact through force reflection may give the operators a greater feeling of safety during operations. It may also allow operators to moderate force applied to task components as they work, so that they do not need to avoid contact.

The experiment described in this paper was concerned with the hypotheses that force reflection is helpful when it provides information that cannot be acquired through vision and when forces applied to tasks are important. The hypotheses were tested in a realistic remote maintenance simulation.

METHODS

The Remote Operations and Maintenance Development (ROMD) facility, which is located at the ORNL, was the site of the experiment. The ROMD

facility ;ts of a high-bay remote handling demonstration area filled with prot;al process equipment and teleoperator systems, along with a control for teleoperators and other remotely controlled equipment. Detailed ;ptions of the ROMD facility may be found in refs. 9 and 10. The CRL M-2 manipulator, which was used in the experiment, is housed in facility. The CRL M-2 has a digital control system that allows so; control over force-reflection levels and which provides a means fo;k switching between force levels. Details of the CRL M-2 and i;trol system may be found in ref. 11. In this experiment, operators the M-2 with 4 to 1 (four units of force at the slave produces ;it of force at the master controller) force ratio, 1 to 1 force rati;d without force reflection.

Four ; were included in the testing. The tasks were representative of c;al plant remote maintenance tasks requiring dexterity to complete. ;re 1 is a photograph of the tasks and task framework. The task f;ork is mounted on a force-torque table. Task 1 consisted of assembl; two pairs of electrical connectors. Two sockets were mounted on of a plate attached to the top of the task framework, and two more w;ounted underneath the top plate. To start the task, the connectors ; placed on top of the task framework. The operators picked up ;onnectors and plugged the ends into the sockets. After inserting a;our connectors, the operators unplugged the connectors and replace;em on top of the task framework.

Task 2 ;isted of a peg-in-hole task mounted within the task framework. ; hole was mounted at a 15° elevation and was offset to the left 15 ;rom the sagittal plane of the task framework. The high end was clo; to the teleoperator package and canted toward the left side (facing ;e task framework) of the package. The task was started with the peg ;ly inserted in the hole. Operators removed the peg, touched the ;k framework with the end of the peg, and reinserted it.

Task 3 ;isted of a pair of stainless-steel tubes with Swagelock-type tubing ;tings. One pair of fittings was mounted on vertical plates on th;op of the task framework. These plates were mounted 45° to the sagit; plane of the task framework and 90° to each other. The other pair o; fittings was mounted on a plate attached to the side of the framewor; The plate on the side of the framework was tilted 30° to the horizont; baseplate and 30° to the vertical side of the task framework. The fr;t (closest the teleoperator when it is in position to perform the t;ks) and outside (farthest from the framework) edges of this plate we; lower than the back and inside edges. To begin the task, the jum;r tubes were placed on top of the task framework. The operators pic;d up the jumper tubes, inserted the ends in the appropriate socket; and tightened the tubing fittings with a wrench. The wrench was pl;ed on the top of the task framework at the beginning of the task.

Task 4 c;sisted of a 3/4-in. nut welded to the plate on top of the task framewor; and an accompanying 3/4-in.-diam, 3-in.-long bolt. Operators screwed t;e bolt into the nut to a criterion depth. At the start of the task, t;e bolt was placed on top of the task framework near the vertical plate to which the nut was attached. The operators picked up

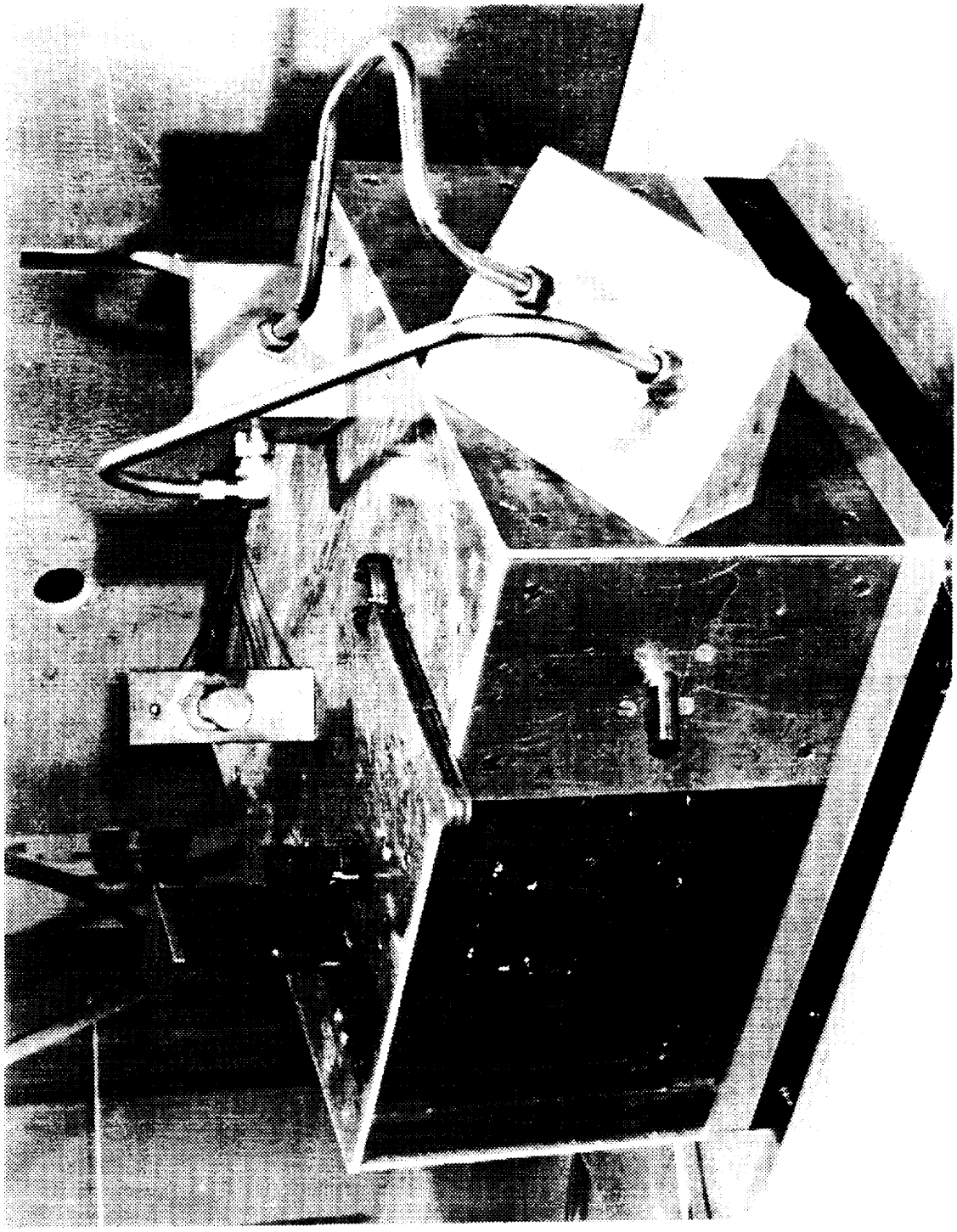


Fig. 1. Task framework, tasks, and load table.

the bolt, positioned it in the nut, and rotated it until it engaged the nut. The operators continued rotating the bolt until the criterion depth was reached. A light mounted on the task framework above the nut indicated when the criterion depth was reached.

Six qualified (according to standard ROMD facility training procedures) teleoperator users participated in the experiment. Every operator completed 1.5 h of practice prior to the start of testing, 0.5 h with each force-reflection level. After the practice sessions, each operator completed 15 testing sessions. A testing session consisted of six trials, two each with three experimental tasks (electrical connectors, peg-in-hole, and tubing jumpers tasks). Within sessions, task were administered in random order under the constraints that no task would be completed consecutively and no task would be repeated before each task had been done once. Six of the 15 total sessions also included one repetition of the bolt threading task. A rest period of at least 1 h between consecutive sessions for one operator was required, and no more than 96 h were allowed to elapse between testing sessions for an operator. These restrictions prevented operators from becoming overly fatigued or out of practice.

Television views were restricted to those available from cameras on board the M-2 package (see refs. 9 and 10). Two cameras were set up to give views from approximately 45° to either side of the tasks (outboard of the sagittal plane of the teleoperator). Operators were allowed to use either of these views and/or a camera view from between the teleoperator arms. Operators were not allowed to see views from cameras other than those onboard the M-2 package during trials, and they were not allowed to change the aiming, magnification, or position of cameras.

Three categories of dependent variables were recorded. The rate of task performance was measured by recording the time in seconds required to complete tasks, the quality of task performance was measured by recording the frequency of occurrence of each of 18 different types of errors, and the effect of the teleoperator on the remote area was measured by recording the forces applied to task components. This multimethod approach to performance quantification avoids bias which may result from defining performance as only one variable. Data were recorded using a Hewlett-Packard 9236 computer programmed in Multi-FORTH to scan 21 channels of A/D information and store the data on a hard disk/streaming tape drive system. Errors included 18 items such as collisions, dropping grasped items, items slipping in the grasp of the teleoperator, collisions, damage to teleoperator or task. A complete list may be found in ref. 7. Forces, torques, velocity of each joint of the right-hand slave, and motor currents of selected joints were recorded 20 times per second. Force and torque data were provided by the load table on which the tasks were mounted. These data were later reduced to resolved force and torque values for each data point. Preliminary analysis of force and torque data revealed that the correlation between these variables was high (for average force and average torque, $r = 0.78$; for maximum force and maximum torque, $r = 0.82$) and the averages and standard deviations of the two variables were similar (average force = 5.23, standard deviation = 2.58; average torque = 4.76, standard deviation =

2.11). These variables seemed to be measuring the same dimension of performance, so the torque data were not included in the statistical analyses.

RESULTS

This paper will concentrate on the results of analyses conducted on the time required to complete each task (converted to its logarithm to base 10), the rate (per minute) of errors, the maximum force exerted, and the force variance within each task repetition. The last variable is a measure of the consistency with which operators apply forces in the remote area. High scores indicate inconsistency in force application, and low scores indicate uniform use of force throughout a task repetition.

The dependent variables were submitted to repeated-measures multivariate analysis of variance.[12] Separate analyses were performed for each task. Details of the statistical analyses may be found in ref. 13. This paper will summarize results of the comparison between force reflection levels. In the sections to follow, significant effects of force reflection will be described but details of the tests will not be reported. The significance level of F tests (the statistic calculated by MANOVA and ANOVA) in the analysis was alpha <0.05 (alpha is the probability of making a mistake in declaring two averages different). Figures 2, 3, 4, and 5 illustrate the averages of each variable for the task and force level combinations.

ELECTRICAL CONNECTOR TASK

The MANOVA found a significant overall impact of force reflection for the electrical connector task. The ANOVA found this effect to be significant for the maximum force and force variance variables. Operators had higher peak forces and larger variance in forces without force reflection than they did with it. There was not a significant difference between 4:1 and 1:1 levels, although on average both variables were lower than in the nonforce condition.

The test for time to complete was very close to significance, with F reaching the 0.06 alpha level. On average, operators completed this task in less time with 4:1 than with either other level.

PEG-IN-HOLE TASK

The MANOVA found a significant overall impact of force reflection for the peg-in-hole task. The ANOVA found this effect to be significant for the error rate, maximum force, and force variance variables. Operators had higher error rates, higher peak forces, and larger variance in forces without force reflection than they did with it. There was not a significant difference between 4:1 and 1:1 levels.

TUBING JUMPERS TASK

The MANOVA found a significant overall impact of force reflection for the tubing jumpers task. The ANOVA found this effect to be significant for the error rate, maximum force, and force variance variables.

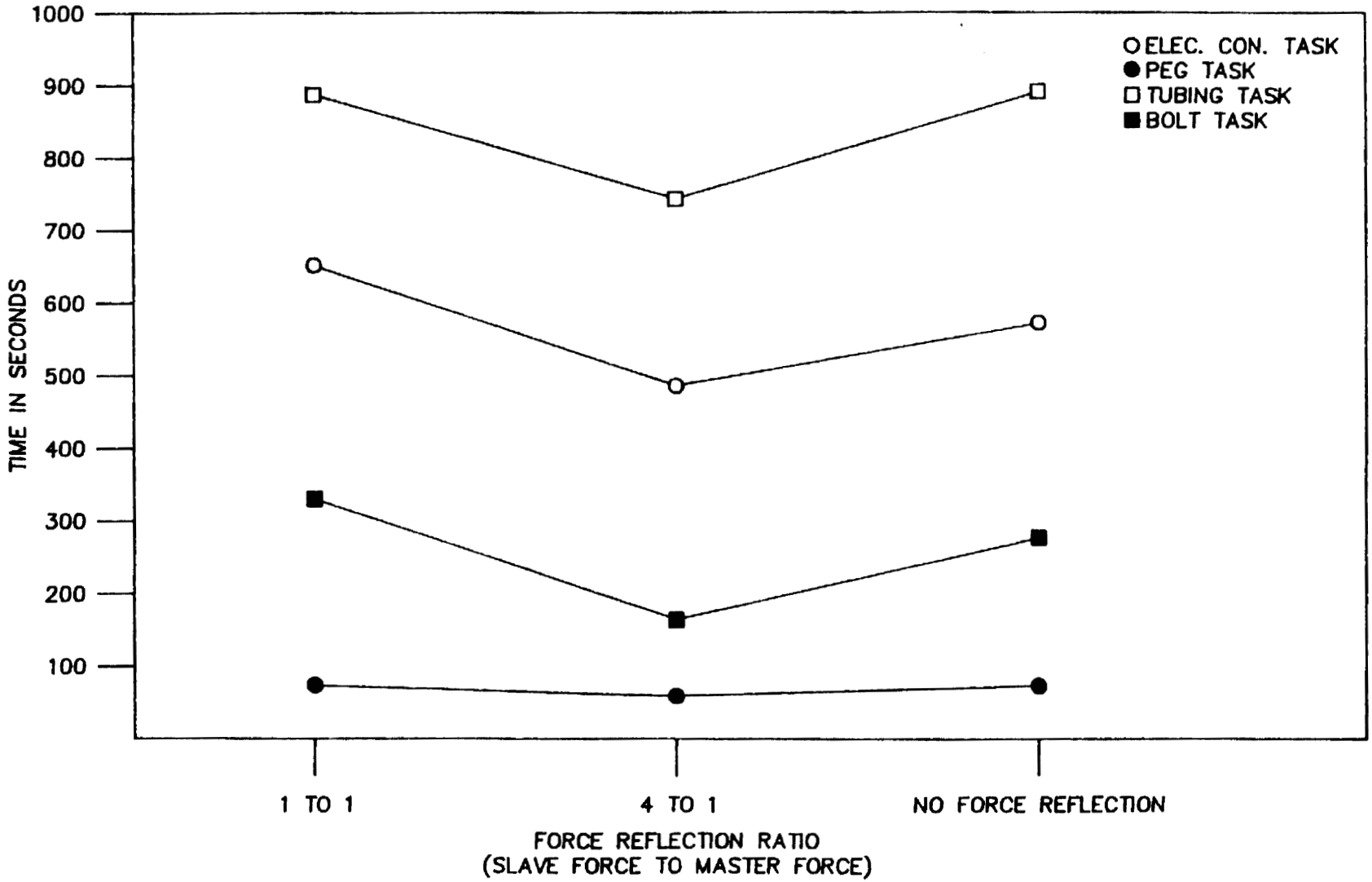


Fig. 2. Average time required to complete tasks.

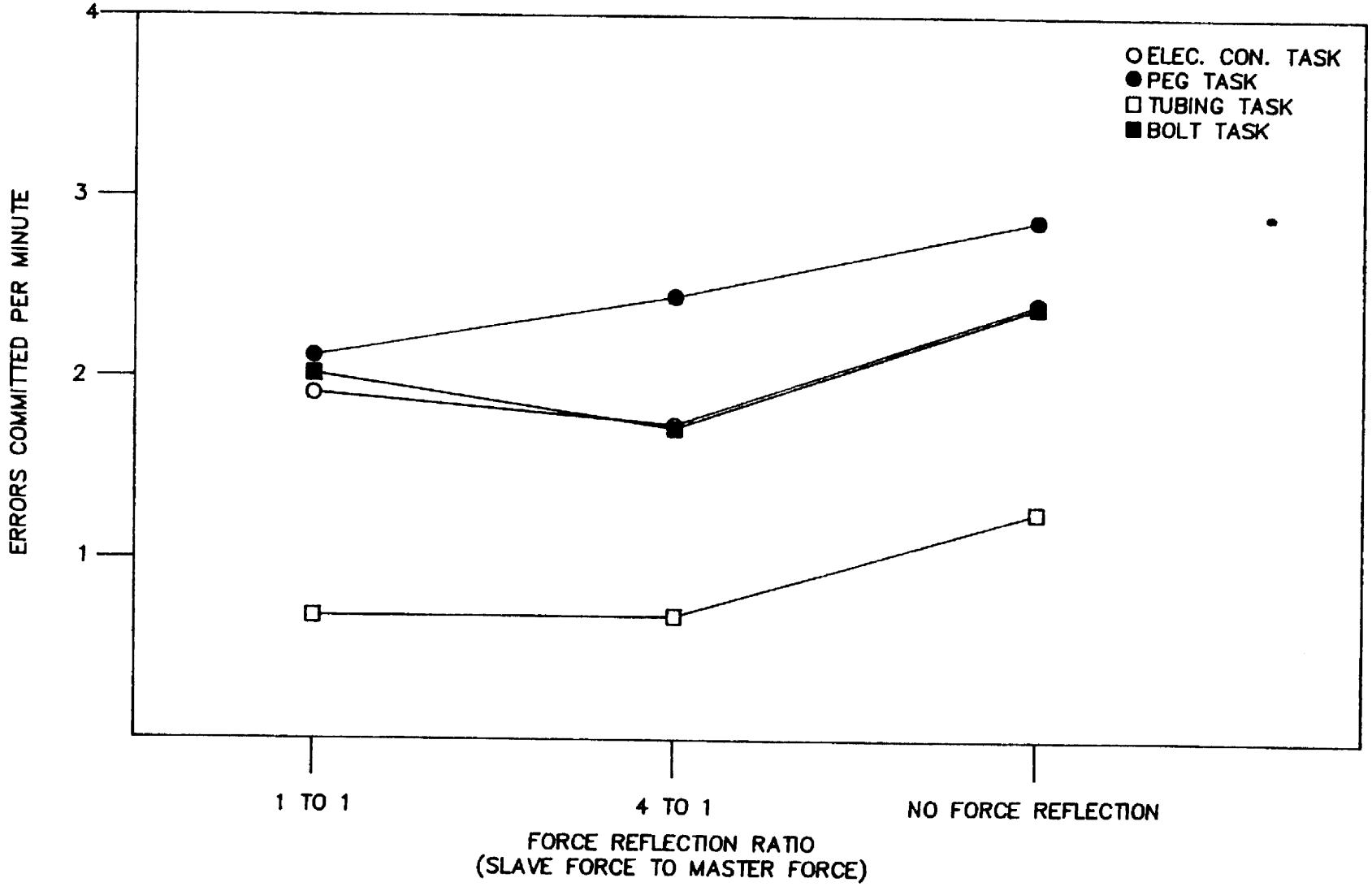


Fig. 3. Average rate of error occurence.

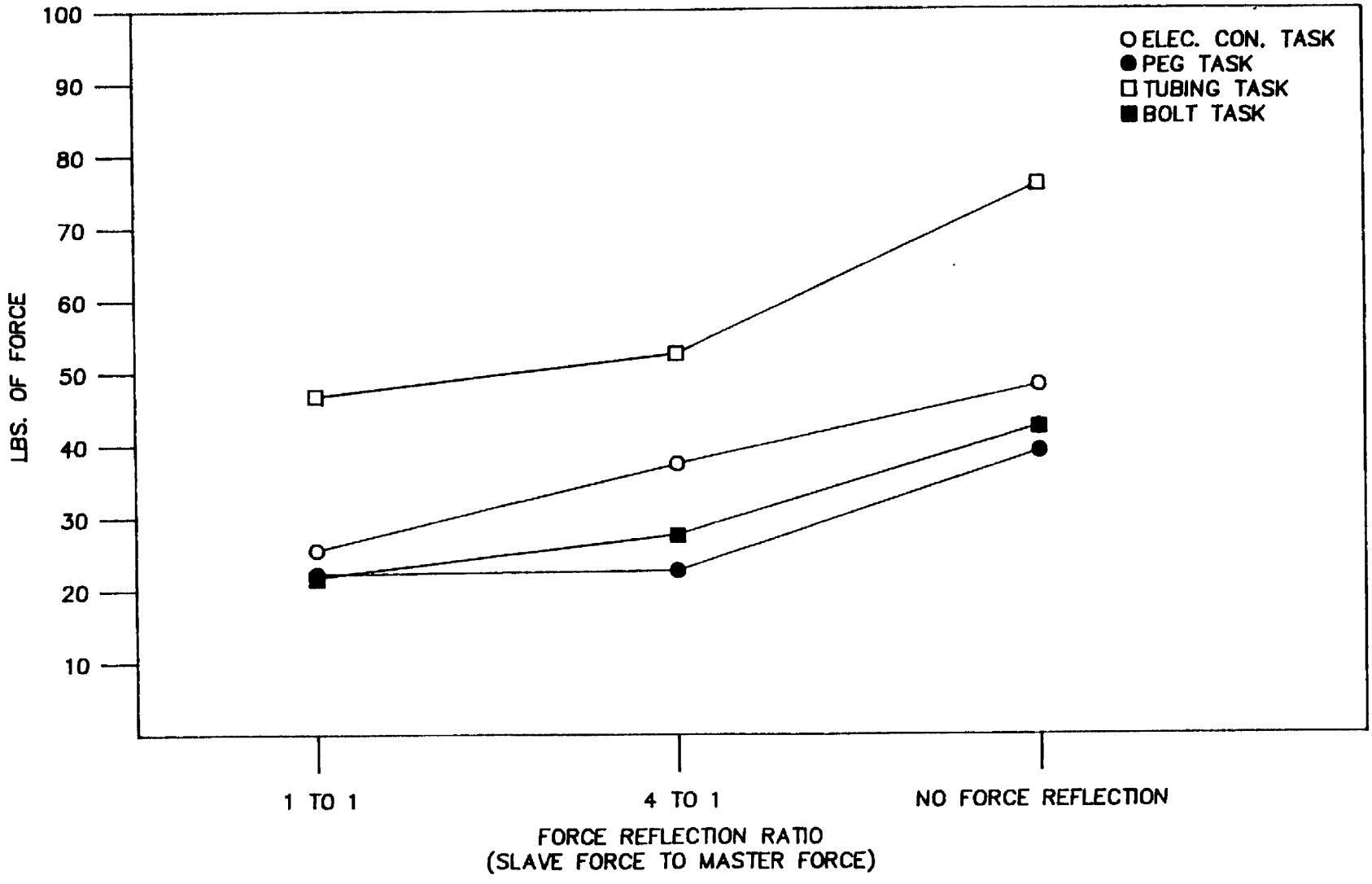


Fig. 4. Average of maximum force per task repetition.

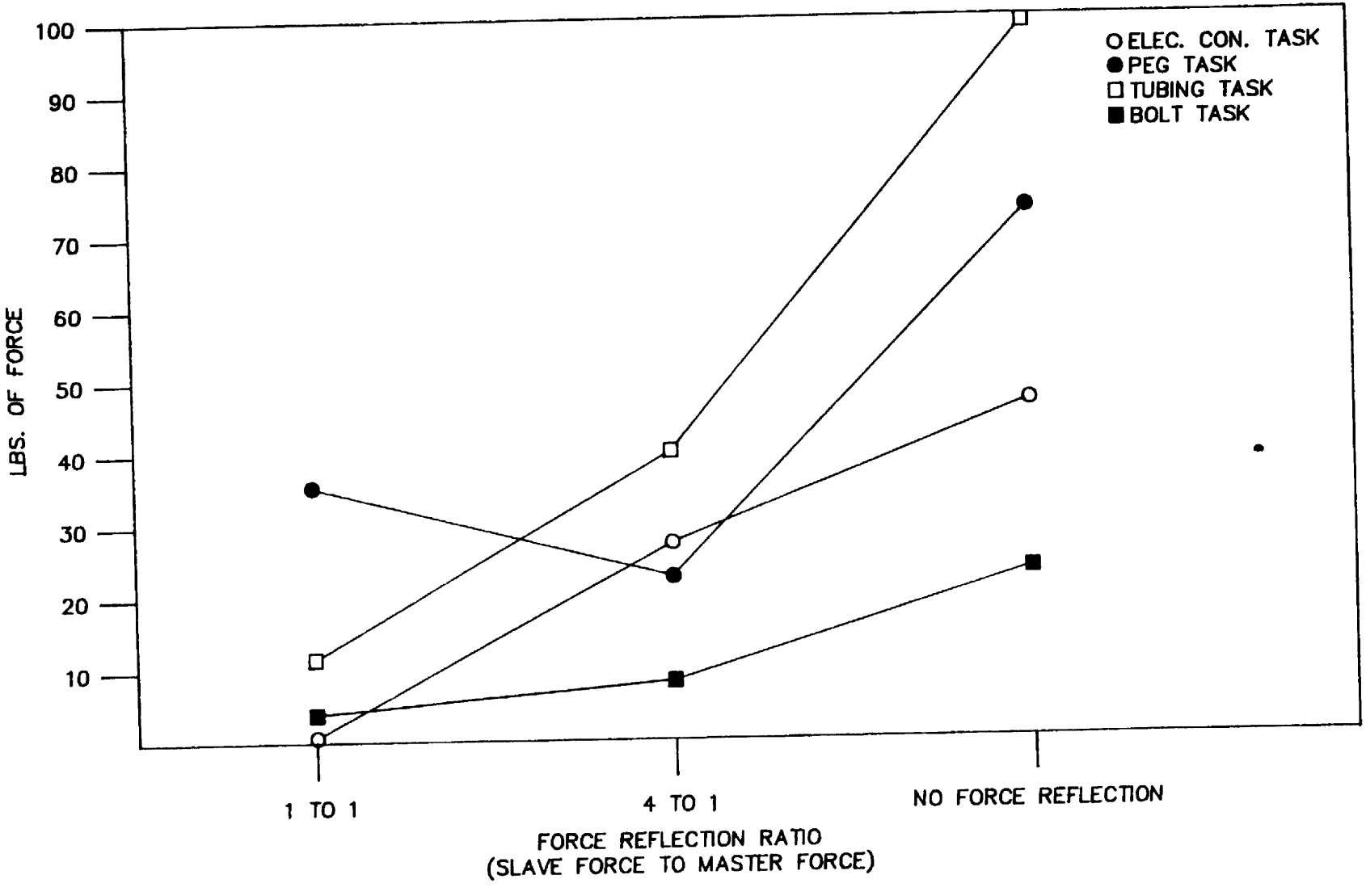


Fig. 5. Average reliability of exerted forces per task repetition.

Operators had higher error rates, higher peak forces, and larger variance in forces without force reflection than they did with it. There was not a significant difference between 4:1 and 1:1 levels.

The test for time to complete was very close to significance, with F reaching the 0.07 alpha level. On average, operators completed this task in less time with 4:1 than with either other level.

BOLT-THREADING TASK

The MANOVA found a significant overall impact of force reflection for the bolt-threading task. The ANOVA found this effect to be significant for the maximum force and force variance variables. Operators had higher peak forces and larger variance in forces without force reflection than they did with it. There was not a significant difference between 4:1 and 1:1 levels.

OPERATOR FATIGUE

Force reflection adds to the friction and inertia experienced by operators using teleoperators. Therefore, it can have an adverse impact on teleoperator ease of use. This could lead to greater operator fatigue while using force reflection, particularly if friction and inertia are high. However, comparison of performance of tasks the first and second time within testing sessions found no evidence of greater operator fatigue associated with force reflection in these data. This may result from to the low friction and inertia characteristic of the M-2 manipulator.

CONCLUSIONS

These data support the hypothesis that force reflection can be beneficial for operators performing remote handling tasks, when the information it provides has no visual analog (operators were unable to gauge deflection or other signs of force in this experiment). It can be particularly useful when forces in the remote area must be controlled. In general, force reflection allowed operators to complete the tasks in this experiment with greater efficiency than they did without force reflection. The time required to complete tasks was not significantly reduced by force reflection (the results are affected by an operator by force reflection condition interaction), but for two of the four tasks the presence of force reflection led to significantly lower error rates. For all four tasks, force reflection allowed the operators to reduce peak forces applied to task components and to be more consistent in the application of forces. Even if there is no difference in task completion time, maintenance campaigns conducted with force reflection can be expected to be completed more quickly than those conducted without it. Operators are less likely to cause damage to equipment during maintenance with force reflection because they are better able to control forces. In addition, operators using force reflection commit errors at a lower rate. These findings are especially important for teleoperation

in space. Control of exerted forces will be much more important in space-based remote operations because of the presence of fragile equipment and the necessity of avoiding unwanted dynamic effects. In addition, operator errors in space will be more difficult to recover from than in terrestrial applications.

It should be noted that the results presented in this paper represent the performance of experienced operators performing familiar tasks. These data do not address the possible impact of force reflection on novice operators or on experienced operators performing novel tasks. It may be postulated that force reflection will be beneficial under these conditions as well, but there are no data to support that hypothesis at this time.

REFERENCES

1. J. V. Draper, Y. Fujita, and J. N. Herndon, Evaluation of High-Definition Television for Remote Task Performance, Oak Ridge National Laboratory, ORNL/TM-10303, April 1987.

2. J. C. Bliss, J. W. Hill, and B. M. Wilber, Tactile Perception Studies Related to Teleoperator Systems, National Aeronautics and Space Administration, NASA CR-1775, April 1971.

3. D. A. Kugath, Experiments Evaluating Compliance and Force Feedback Effect on Manipulator Performance, National Aeronautics and Space Administration, NASA-CR-128605, August 1972.

4. J. W. Hill, Study of Modeling and Evaluation of Remote Manipulation Tasks with Force Feedback, National Aeronautics and Space Administration, NASA-CR-158721, July 1979.

5. J. W. Hill and J. K. Salisbury, Jr., Study to Design and Develop Remote Manipulator Systems, National Aeronautics and Space Administration, NASA-CR-152092, November 1977.

6. T. D. Cook and D. T. Campbell, Quasi-Experimentation: Design and Analysis Issues for Field Settings, Houghton Mifflin Company, Boston, 1979.

7. J. V. Draper, S. J. Handel, E. Sundstrom, J. N. Herndon, Y. Fujita, and M. Maeda, Final Report: Manipulator Comparative Testing Program, Oak Ridge National Laboratory, ORNL/TM-10109, February 1987.

8. C. D. Wickens, Engineering Psychology and Human Performance, Charles E. Merrill Publishing Company, Columbus, Ohio, 1984, p. 253.

9. J. N. Herndon, "The State-of-the-Art Model M-2 Maintenance System," Proceedings of the Robotics and Remote Handling in Hostile Environments National Topical Meeting, American Nuclear Society, Gatlinburg, Tennessee, April 23-27, 1984.

10. T. W. Burgess, "The Remote Operation and Maintenance Demonstration Facility at the Oak Ridge National Laboratory," Proceedings of the International Topical Meeting on Waste Management and Decontamination and Decommissioning, American Nuclear Society, Niagara Falls, New York, September 14-18, 1986.

11. P. E. Satterlee, H. L. Martin, and J. N. Herndon, "Control Software Architecture and Operating Modes of the Model M-2 Maintenance System," Proceedings of the 1984 National Topical Meeting on Robotics and Remote Handling in Hostile Environments, American Nuclear Society, Gatlinburg, Tennessee, April 1984.

12. R. E. Kirk, Experimental Design, 2d ed., Brooks/Cole Publishing Company, Belmont, Calif., 1982.

13. J. V. Draper, S. J. Handel, W. E. Moore, J. N. Herndon, and E. Sundstrom, Final Report: Force Reflection Testing Program, Oak Ridge National Laboratory, in preparation.

Issues, Concerns, and Initial Implementation Results for Space Based Telerobotic Control

D. A. Lawrence*, J. D. Chapel**, and T. M. Depkovich†

ABSTRACT

Telerobotic control for space based assembly and servicing tasks presents many unique problems in system design. Traditional force reflection teleoperation schemes are not well suited to this new application, and the relatively new approaches to compliance control via computer algorithms have yet to see significant testing and comparison. These observations are discussed in detail, as well as the concerns they raise for imminent design and testing of space robotic systems.

As an example of the detailed technical work yet to be done before such systems can be specified, a particular approach to providing manipulator compliance is examined experimentally and through modeling and analysis. This yields some initial insight into the limitations and design trade-offs for this class of manipulator control schemes. Implications of this investigation for space based telerobots are discussed in detail.

* Staff Engineer, Martin Marietta Denver Aerospace
** Senior Engineer, Martin Marietta Denver Aerospace
† Senior Staff Engineer, Martin Marietta Denver Aerospace

This work was performed under IRAD D-75D.

1. INTRODUCTION

Although there has been an intense amount of activity in the area of control system design for robotic manipulator systems over the past ten years, little has surfaced in the way of a unified body of theory that can be readily applied for a given application. For example, the tremendous amount of work in the area of nonlinear systems theory has failed to produce general techniques for control system design that clearly outperform conventional independent joint control techniques. In a similar vein, the work in the area of compliant control systems has failed to produce a consensus on a method for control when environmental interaction is required. All of these problems are further compounded when system requirements dictate the need for both teleoperator and autonomous operational modes. Unfortunately, some solid answers and directions are required now since both NASA and the DOD are ready to embark on major system developments.

Our research and development work at Martin Marietta has focussed on a hierarchical approach to the development of robotic control systems. A general task breakdown is shown in Figure 1. While research continues in the area of nonlinear control methods, we have nonetheless been able to implement reliable compliant control structures that have demonstrated a great deal of promise. This has, in turn, allowed work to proceed in both teleoperation and autonomous coordinated dual arm control structures. The benefits of this approach are twofold. First, it has forced us to make maximum usage of existing theory to develop an overall control structure that supports both autonomous and teleoperated operations for both single and dual arm systems. Secondly, it has consistently forced the issue of hardware implementation. As a result, all key aspects of system performance have been demonstrated on a dual arm laboratory testbed.

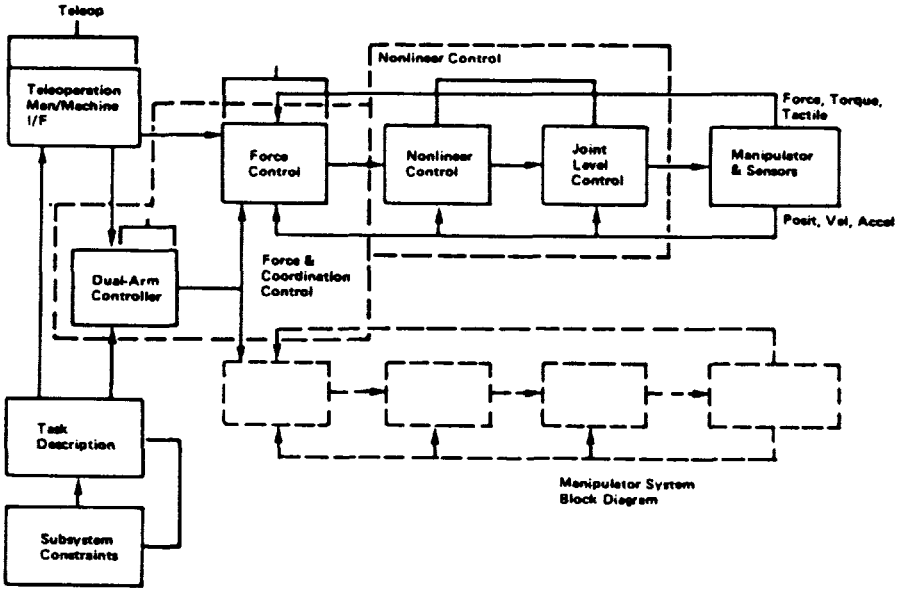


Figure 1. *Overall manipulator control block diagram.*

This paper focusses on the heart of our system--single arm compliant control strategies. The control structure to be described in detail in the following sections relies on independent joint level control but allows specification of the dynamic impedance of the manipulator in any specific reference frame. We view this structure as both a reasonable compliant controller baseline and a tool for more advanced studies. In particular, we present some detailed results on the modeling, analysis, and limitations of this approach to providing compliance for manipulator/task interactions.

2. BACKGROUND AND MOTIVATION

Although advancements in artificial intelligence and computer vision promise fully autonomous robotic servicing in the future, near term space servicing systems, such as the flight telerobotic servicer (FTS), will be primarily teleoperated systems. Because these space servicing systems will be manually operated by a remote operator, small errors in aligning the telerobot with the worksite are almost unavoidable. As shown in an experimental module removal task presented later in this paper, these small errors in alignment and positioning can impart large forces upon the worksite, depending upon both the environmental and manipulator stiffnesses. Clearly, a key aspect of these systems will be the control of these environmental interaction forces.

Both nuclear and underwater teleoperation systems have extensively utilized the technique of bilateral force reflection or, simply, force reflection to control interaction forces. This technique provides force information to the operator by backdriving motors on the master arm with signals proportional to sensed forces. Because the master arm controls the position of the telerobot's manipulators, backdriving the master arm modifies the postion command to the manipulators. Thus, the dynamics of the teleoperator system using this force reflection scheme are determined by the characteristics of the hand controller/human arm combination. By using this scheme, teleoperation task completion times have been shown to be reduced by up to 40% [1].

There are substantial differences between these teleoperation systems and those envisioned for space teleoperation. Because space telerobotic servicing systems will be all electric, the damping inherent in hydraulic systems, which helps stabilize them, will not be available. Size and weight constraints, as well as scaling problems, will preclude the use of replica master controllers in space. The resulting D/A and A/D conversions and kinematic mapping calculations needed for digital control of the telerobot can introduce significant time delay into the system. Finally, the communications link may have minimal bandwidth resulting in a slowly sampled system controlled by a low bandwidth controller. Because all of these factors are expected to adversely affect system performance, review of the traditional teleoperator control techniques is warranted to assess their applicability to space telerobotics.

One alternative method of providing telerobot compliance can be realized by replacing the dynamics of the force reflection hand controller with a digital filter. In other words, the measured contact forces can be processed through a digital filter to provide the desired dynamic relation between experienced forces and manipulator motion commands. Because this control scheme effectively modifies the mechanical impedances of the manipulator according to the parameters of this digital filter, it is

known in the robot control literature as impedance control or force control [2,3]. Using this scheme, the stiffness, damping and inertia parameters, or mechanical impedance, can be programmed to vary the dynamic characteristics of the telerobot system. In particular, the filter parameters can be programmed so that the impedance controller has approximately the same closed loop dynamics as the bilateral force reflection controller. It is this equivalence between the two controllers that we wish to exploit to further understand the control of environmental interaction forces for telerobots.

PRACTICAL IMPLEMENTATION

The robotic laboratory facilities used to examine these two telerobot force control schemes at Martin Marietta are shown in Figure 2. The Cincinnati-Milacron T^3-726 industrial robots are essentially position-control devices, servoing about a given reference position. The operator can modify the reference position by moving a compact, 6-DOF hand controller, which has the capability for force feedback. Because the hand controller has only ± 1 inch travel in the three translational degrees of freedom and only ± 30 degrees travel in the three rotational degrees of freedom, it must be indexed or "racheted" to produce large changes in the robot's position. Each robot has a 6-DOF force-torque sensor attached to the mounting plate of the gripper. The force information from this sensor can be used to drive the force feedback motors on the hand controller thereby providing bilateral force reflection, or can be used as inputs to the impedance filter whose outputs are used to modify the robot's reference position directly.

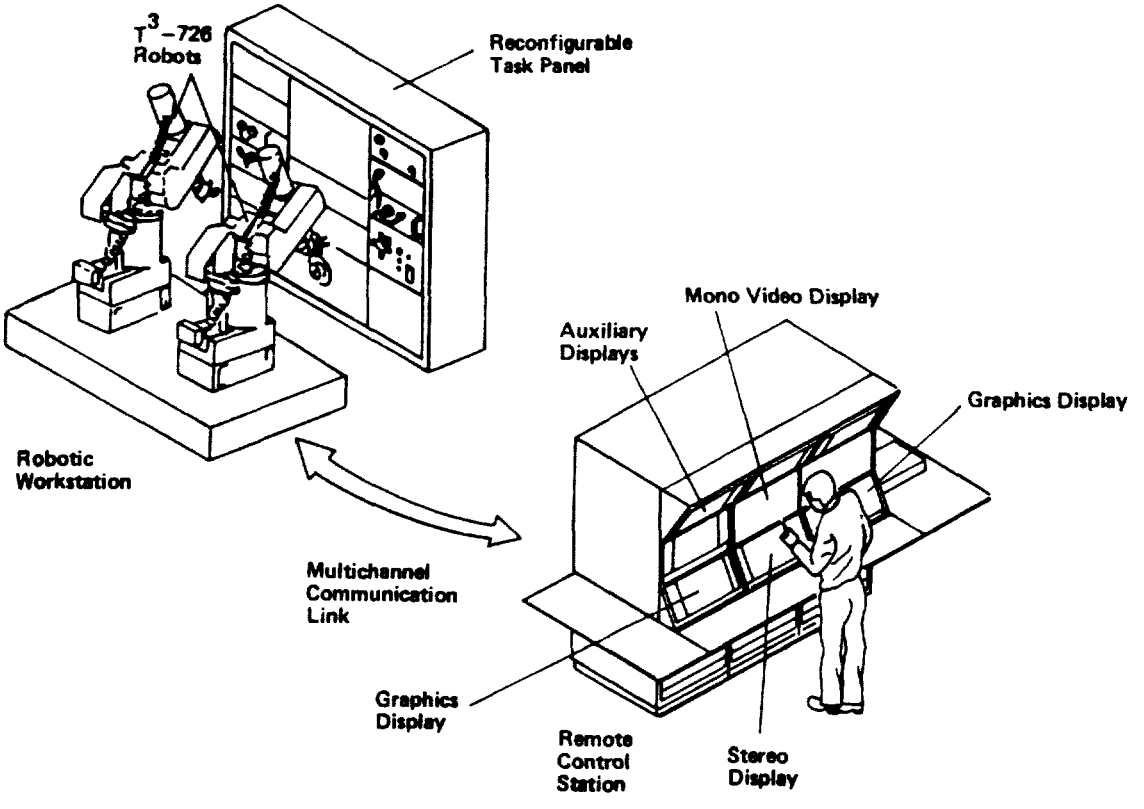


Figure 2. Martin Marietta telerobotic research facilities.

To compare these two control techniques, a module removal task was chosen as representative of spacecraft servicing. In the laboratory, this task is simulated by a task panel drawer that slides along an axis, x_p, perpendicular to the task panel's face. To demonstrate the performance in compensating for small misalignments, the task panel is set up so that the
task panel's coordinate system must be rotated by an angle θ to align the task panel's x_p axis with the robot's x coordinate axis, as shown in Figure 3. This angle is only 4 degrees, so it is difficult, at best, for the operator to determine that any misalignment exists. To perform the task successfully, the operator must command the manipulator to grasp the handle on the drawer, pull the drawer out to its full extension, hold it at that point for 10 to 15 seconds, and then push the drawer closed. As mentioned earlier, the hand controller must be racheted to move the manipulator from the initial grasp point to tthe maximum x displacement, which is about 10 inches. Because of this racheting action, the position profile in the x direction is expected to be as shown in Figure 3. The manipulator and the environment are assumed to have stiffnesses K_m and K_e, respectively in the directions of interest. Therefore, as the minipulator is given a commanded position trajectory to pull the drawer along the manipulator's x axis, the misalignment angle θ will produce a static force in the y direction of

$$D\left(\frac{K_e K_m}{K_e + K_m}\right)\sin\theta \approx DK_e\sin\theta$$

where D is the distance measured along the x axis from the initial grasp point. The expected force profile in the y direction is also given in Figure 3.

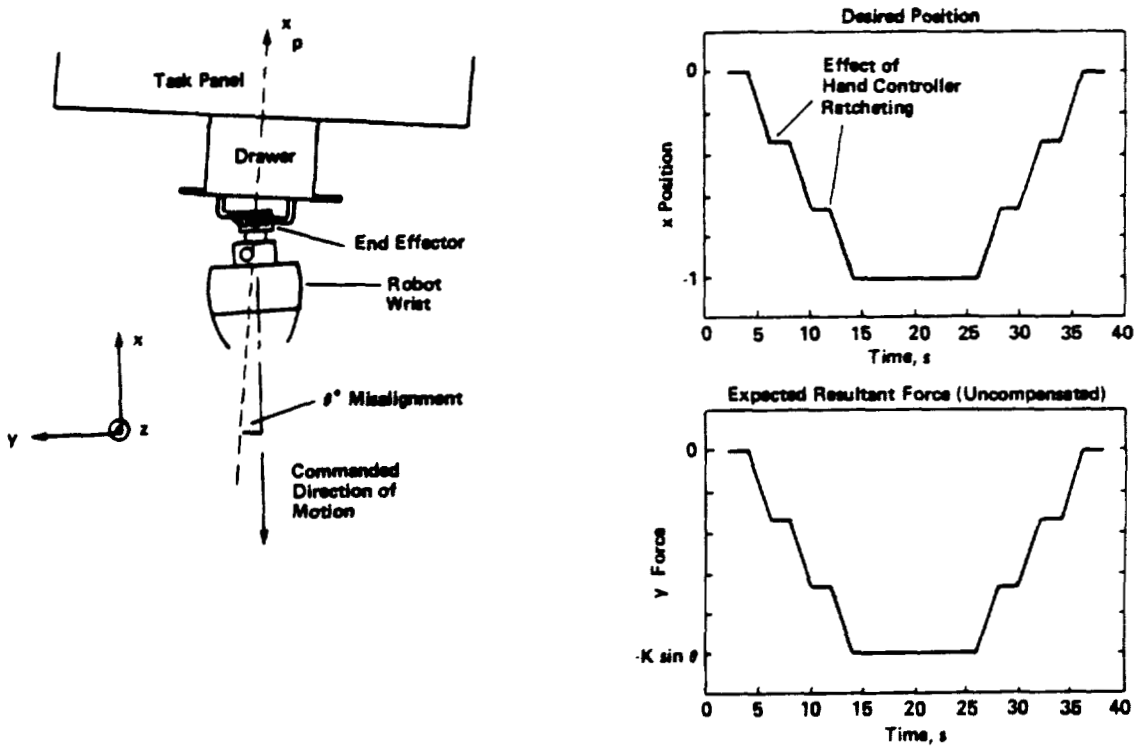


Figure 3. *Misaligned module removal task.*

Three independent trials were made using this experimental setup. In the first trial, the force information was not used in any way so the operator had only visual information to perform the required task. In the second trial, the technique of bilateral force reflection was used to drive the motors on the hand controller proportionally to the sensed forces. In the third trial, the force information was used as input to an impedance filter, and the output of this filter was used to modify the position references. The results of this experiment are shown in Figure 4.

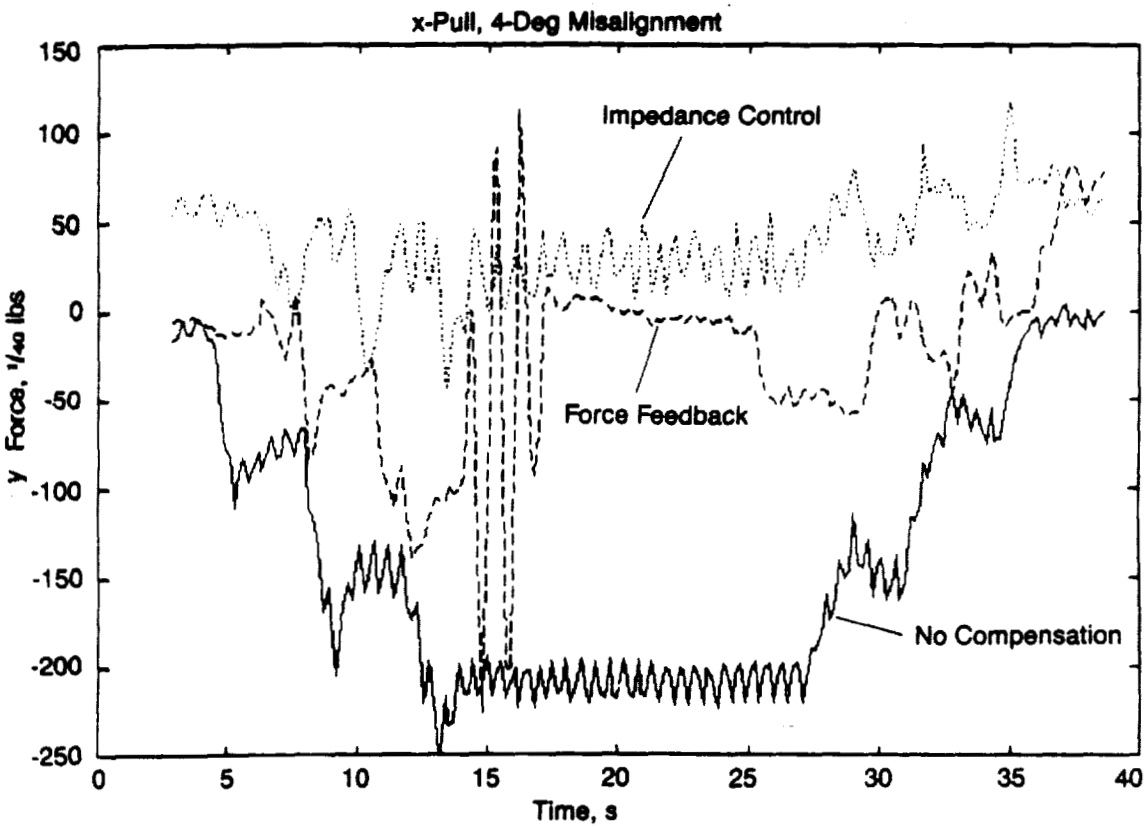


Figure 4. *Misaligned task experimental results.*

Because the manipulator itself is very stiff, the y position in the first trial is nearly constant. As a result, the task panel drawer must comply to compensate for the misalignment. The experimental parameters produce a steady state force in the y direction of more than 5 lbs at the maximum displacement. This force level is certainly more than is desirable. If this force level could be reduced by an order of magnitude by using some form of compliant control, the performance of this simple task would be much more acceptable.

The results of the second trial with force reflection were somewhat unexpected. By making the operator conscious of the force in the y direction, the operator is able to compensate for the forces by moving the hand controller in the opposite direction of the sensed force. The steady state force in the y direciton is reduced to less than 1 lb by using the force information in this way. However, the oscillations occurring in

the force profile at approximately 15 seconds into the trial produced forces equivalent to those observed in the first trial. These oscillations reflect the difficulty an operator has in stabilzing the system. A similar oscillation was observed in the y position profile. The only time delay in the system results from the computational requirements for the digital controller, so no stability problems were anticipated due to time delay. An initial response to these oscillations is to decrease the gain in the force reflection, but the operator loses "feel" for the task when this gain is reduced. Clearly there are additional dynamics in this system that need to be better understood.

In the third trial, the impedance control structure is seen to be as effective as the force reflection controller in reducing interaction forces. However, the impedance controller does not exhibit the oscillatory tendancies of the force reflection controller. Because of the difficulty in estimating the stiffness/damping parameters for the force reflection hand controller/human operator system, it is nearly impossible to choose parameters for the impedance controller to precisely emulate the dynamics of the hand controller. Instead, the impedance control parameters were selected to make tthe system fairly compliant, but viscous enough to maintain stability. Oscillatory behavior was observed when the stiffness and/or damping parameters in the filter were chosen to be to low. In fact, some combinations of impedance control parameters were seen to produce an oscillatory system with oscillations that were growing with time.

The similarity in performance between these two force control methods suggests further similarities in their dynamic models. In both cases, oscillatory behavior has been observed with certain parameters. Furthermore, this oscillatory behavior can be modified to produce acceptable system behavior by altering parameters within either controller. In the force reflection control system, the operator has difficulty stabilizing the system, even with small time delays. While in contact with rigid objects, this form of compliant control requires intense concentration and a fair amount of physical effort. The operator is able to stabilize the system by exerting more force on the hand controller to stop the oscillations. He is essentially modifying the dynamics of the force reflection controller to stabilize the system. He is certainly adding more stiffness to the system by exerting force on the hand controller, but our understandings of physical systems tell us that damping is also needed to damp out these oscillations. Similarly in the impedance control case, we have found that making the coefficient on the filter's damping term larger tends to stabilize the system. The trend observed by adding to the stiffness term is less clear, but it is clear that when the manipulator is stiffer than the environment there is no stability problem (although the manipulator will not comply in response to force inputs). Intuition also tells us that this situation is aggravated by adding time delay into the system.

3. MODELING AND ANALYSIS

The behavior of the bilateral force reflection and the impedance control observed in experimental tests was perplexing, since the prevalent theory behind the implementation did not suggest that stability could be a problem. However, the data suggested that the system could become unstable for certain combinations of programmed impedance and environmental conditions, or when the human operator was not "stiff" enough. This situation raised some important questions. Is the anamolous behavior inherent in the basic control technique, the implementation, or

both? Are there "safe areas" of operation which can be designed around? Are there other control techniques which are more suitable?

To gain a deeper understanding of the problem, a detailed investigation was initiated. The goals were to first understand the essential cause of the observed behavior, and then to build on this insight using more detailed models and more thorough experimental verification. The remainder of this paper describes the results obtained in the first phase of this investigation. The second phase is the subject of ongoing research.

Analysis of the observed behavior may seem a complex problem at first. The interaction of a six DOF robot with a task containing the various dynamics of the human and hand controller is a complex process indeed. A full dynamic model of this system would contain far too many parameters to lend any insight into its behavior. Such an approach is the undoing of many proposed manipulator control schemes in the literature--the essential benefits are masked by unjustified complexity/

The approach taken here, in contrast, seeks to discover the most fundamental models which can predict the observed behavior. Both the force reflection teleoperation and impedance control are considered using a single dynamic model. In either case, they act to process sensed forces and provide manipulator position commands. In the impedance control case, the position commands are related to the forces by a programmed filter. In the teleoperation case, the force/position relation is governed by the dynamics of the hand controller, human hand and arm. At the simplest level, both systems have the same model structure: a spring/damper/inertia combination. With respect to dynamic performance, these systems are essentially the same. We will refer to both cases as a compliance control system in the following discussion.

Since the performance objective can be thought of as assuring some desired behavior in each cartesian axis, including rotations about these axes, it would seem that satisfactory behavior in each axis would be a necessary condition for acceptable behavior overall. This intuitive principle provides motivation for the simple modeling and analysis below, where a single cartesian axis model is investigated in detail. To indicate the validity of the results obtained, experimental results of a full six DOF implementation along one axis are compared with the analysis. While experimental error prevented verifying exact correspondance, the trends predicted by the analysis were clearly verified by the hardware data. This experimental test is also discussed in detail below.

Before describing the details of the analytical results, it may be helpful to place the approach in context with others from the control literature. There seem to be two main approaches to specifying objectives for the control scheme. The first considers specific objectives on either position or force in a particular cartesian direction. Typical goals are zero steady state error in these individual quantities, or to minimize these errors in other senses. The basis of this approach relies on a separation of the six axes into "force control" and "position control" axes, which depend on the particular geometry of each task [4,5]. This approach is often called the "hybrid" control technique.

The second main approach considers the control objective to be a constraint on position and force in each cartesian direction. This constraint is often expressed as a desired mechanical impedance [2,6]. For example the constraint

$$F = Kx + B\frac{dx}{dt} + J\frac{d^2x}{dt^2}$$

would cause the robot and effector to behave as a spring of stiffness K, a damper of damping B, and an inertia of value J. The resulting impedance is expressed as

$$Z = K + Bs + Js^2$$

where s is the Laplace derivative operator. Actually, this view contains the hybrid approach as a special case, since small position errors are achieved by large stiffnesses, and force error zeroing can be achieved by zero stiffness and non-zero damping or inertia. For the case of teleoperation with force reflection, the human operator cannot provide wide ranges of impedances simultaneously in various directions, due to coupling in the arm and hand. Hence, this is more of an impedance control technique rather than a hybrid approach.

Another distinguishing feature of the different methods is the particular way the objective is implemented via feedback control. This is perhaps the most confusing issue in the field. We consider the impedance objective to have two main implementations. The first seeks to measure the cartesian position and orientation of the robot end effector, then command joint torques to supply the desired force according to the impedance constraint [2,5,6]. This is termed "torque based" impedance control, since the robot commands are in the form of torque. The second approach measures end effector forces and commands the robot cartesian position, again according the impedance objective. This is the "position based" approach to impedance control, since robot commands are in the form of positions [7,8,9]. Some suggested control schemes [3] differ slightly from these two main approaches, and some contain a mixture [2,6] of these ideas. However, this view captures the essential ideas behind the various approaches. Of course, force reflection teleoperation is necessarily a position based approach.

While there are many issues in the relative advantages of these methods when implemented via computer control, a key issue exploited here is that the joint control used to follow position commands also provides a large degree of joint dynamic decoupling. This greatly simplifies the control design task, since the heirarchical successive loop closure approach can be used. Hopefully, a more complete understanding of the limitations of this approach, together with other approaches, can be gained. Only then can rational choices be made between approaches and implementation details for particular applications. The analysis presented below yields some insight into the limitations of position based impedance control.

We consider the following physical situation. The manipulator base is attached to a common mounting with a task fixture. The end effector contains a six axis force sensor, which measures the forces and torques occuring at the contact point when the manipulator interacts with the task. The manipulator is under joint position control, and can be commanded to move according to cartesian reference commands by passing these commands through the inverse kinematic transformation to arrive at joint position commands. The actual cartesian position is available by passing the measured joint positions through the foreward kinematic transformation.

If the joint position control were perfect, and the kinematics were computed instantaneously, then the manipulator motion in each cartesian direction would be exactly as desired, and it would be decoupled from motion in all other cartesian axes. Practically, this behavior is approached at low frequencies, but degrades at higher frequencies. Thus we model the behavior of the manipulator in a particular cartesian direction as a positioning system with second order dynamics. This provides a basic model of the positioning fidelity as the frequency increases, but does not include coupling effects from other cartesian axes. As it turns out, the unstable behavior we wish to model occurs at low frequencies relative to those at which our model loses significant accuracy.

The impedance control is implemented in an outer feedback loop which measures interaction forces, passes these through an impedance specification filter, and updates the commanded cartesian position. Again, this models both the teleoperation and computer control implementations. The sensed foreces are dependent on the position of the contact point as well as the dynamic properties of the manipulator and task. For many tasks, their dominant character is represented by a pure stiffness, since it is often very large compared to any damping present in the task. Also, in the present case where the task is rigidly attached to the manipulator base frame, no inertial effects in the task are significant. The resulting model for the interconnected system is shown in block diagram form in Figure 5. The manipulator dynamics are parameterized by K_m, B_m , and J_m. The environmental (task) stiffness is K_e, and the desired impedance specification has stiffness K and damping B. X is the cartesian position along the axis in question, and F is the sensed interaction force. Note that if X accurately tracks the impedance update X_i, then the desired manipulator impedance will be achieved, ie. the constraint between F and X will be realized. This impedance is independent of the environmental impedance K_e, but the total impedance seen at the contact point is the sum of the manipulator and environmental impedance. This fact suggests that the system will be stable if the parameters are all positive [10]. The system should act like a connection of positive springs, masses and dampers.

However, the implementation of various kinematic transforms, rotations into the correct reference frame, etc. cause time delays in practical systems. Together with potentially large feedback gains due to small desired stiffness K and damping B, this time delay can cause instabilities in the overall system behavior. Due to the simplicity of our model, the precise relationship between the desired impedance, the environmental impedance, the time delay, and manipulator dynamics can be determined in order to retain stability.

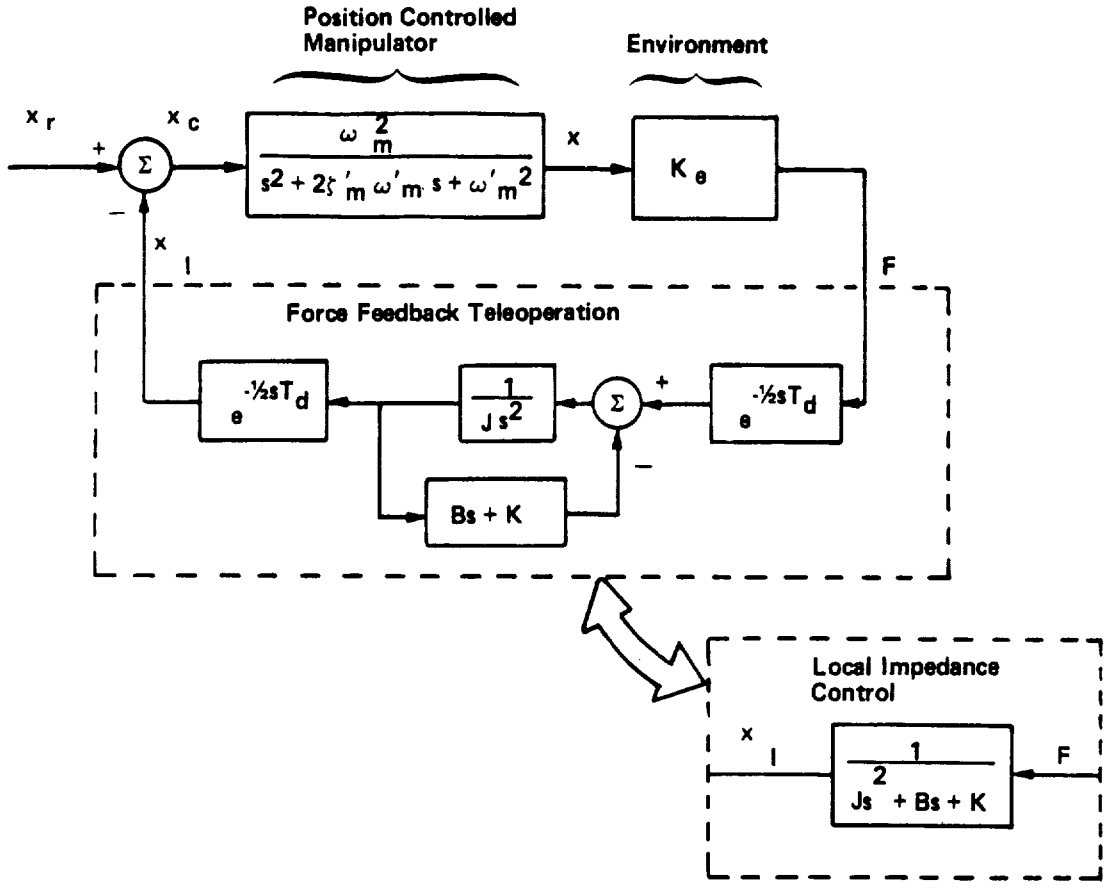


Figure 5. *Position Based Impedance Control Block Diagram.*

Since the system is infinite dimensional when time delays are present, gain and phase margins are used to determine the stability boundaries as parameters are varied. Based on the practical assumption that the manipulator dynamics are well damped, the open loop transfer function of the model in Figure 5, given by

$$G_{OL} = \frac{K_m K_e}{\left(J_m s^2 + B_m s + K_m + K_e\right)\left(Bs + K\right)}$$

can be shown to represent a stable closed loop system if and only if the phase margin is positive [9]. For a given set of manipulator dynamics parameterized by K_m, B_m, and J_m, and for a given environmental stiffness K_e, the time delay nT_S and desired impedance parameters K and B must satisfy both

$$K_m^2 K_e^2 = \left[\left(K_m + K_e - J_m\omega^2\right)^2 + B_m^2\omega^2\right]\left(K^2 + B^2\omega^2\right)$$

and

$$\pi = \tan^{-1}\left(\frac{B_m\omega}{K_m + K_e - J_m\omega^2}\right) + \tan^{-1}\left(\frac{B\omega}{K}\right) + \omega n T_s$$

for marginal stability of the closed loop system. This stability boundary in K and B is shown in Figure 6 for particular values of the other system parameters, and for various time delays nT_s. This figure also shows the stability boundary for a discrete time model of the impedance control loop. Both boundaries are quite close, except when B is very small. For further details on these results, see [9,11]. Observe that the region where both K and B are small is excluded for stable behavior. This restricts how "soft" the manipulator can be made to appear, ie., how small its impedance specification can be. Also, note that small K or small B can be achieved only if the other is relatively large. Finally, the larger the time delay, the larger K and B generally have to be to remain in the stable region. The implications of these stability results are discussed in detail in section 4.

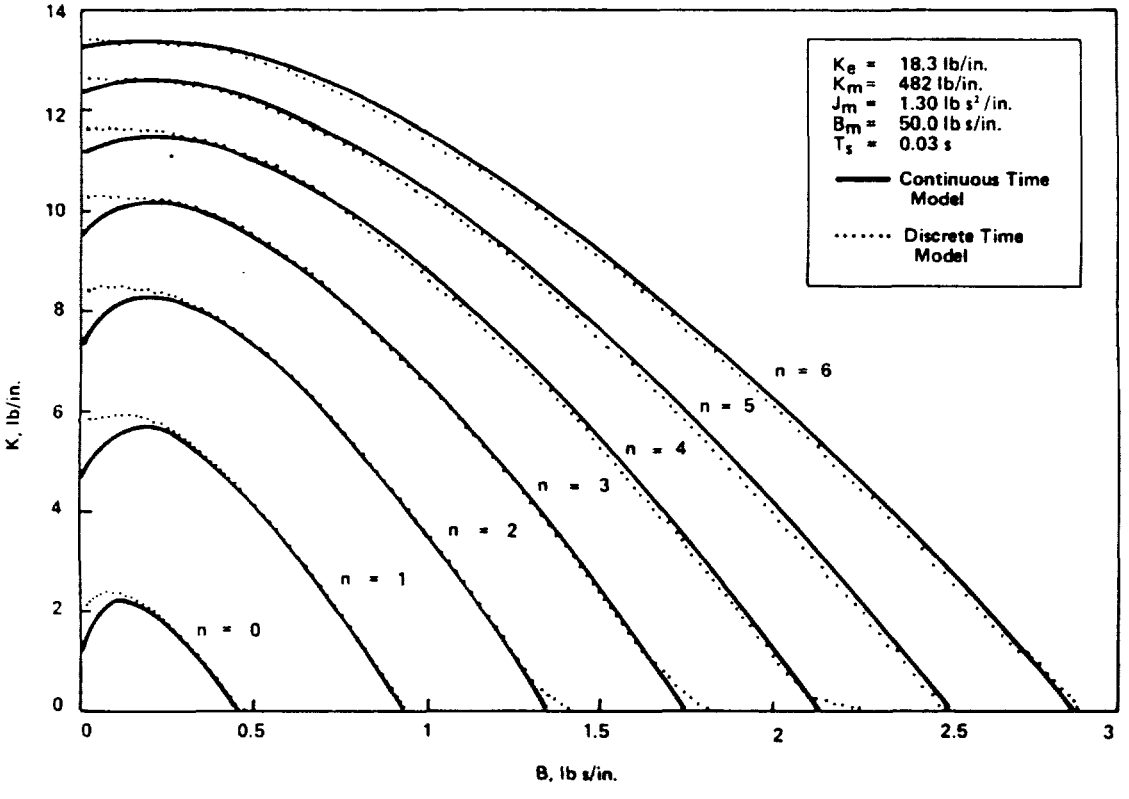


Figure 6. *Stability boundary in K and B for various time delays, nT_s.*

As an indication of the validity of the model and its prediction of the stability/instability boundary, consider the following comparison with an experimental determination of the boundary. For the test, the manipulator was programmed to have very large stiffness and damping in all cartesian directions but one. In these directions, the feedback gain from the force sensor to position update is essentially zero, and the stiff manipulator behavior is not modified. In the direction in question, however, various B and K values were used in the impedance specification block to find the values which caused unstable behavior. The task in this case consisted of a simple linear spring with known spring constant, as shown in Figure 7. The measured stability boundary is plotted in Figure 8, along with the theoretical boundary for the manipulator parameters and software time delay obtained from previous identification work. Here,

$$\omega_m = \sqrt{\frac{K_m + K_e}{J_m}}, \qquad \zeta_m = \frac{B_m}{2\sqrt{J_m\left(K_m + K_e\right)}}$$

The large measurement uncertainty bands are due to the presence of limit cycles near the stability boundary. Rather than a sharp division between growing oscillations and decaying oscillations, as the linear model above would predict, limit cycles of varying sizes were seen along the stability boundary. Outside this band, decaying and growing oscillations were observed. Thus, the overall trend predicted by the anlaysis is clearly observed, but close agreement in actual values was not.

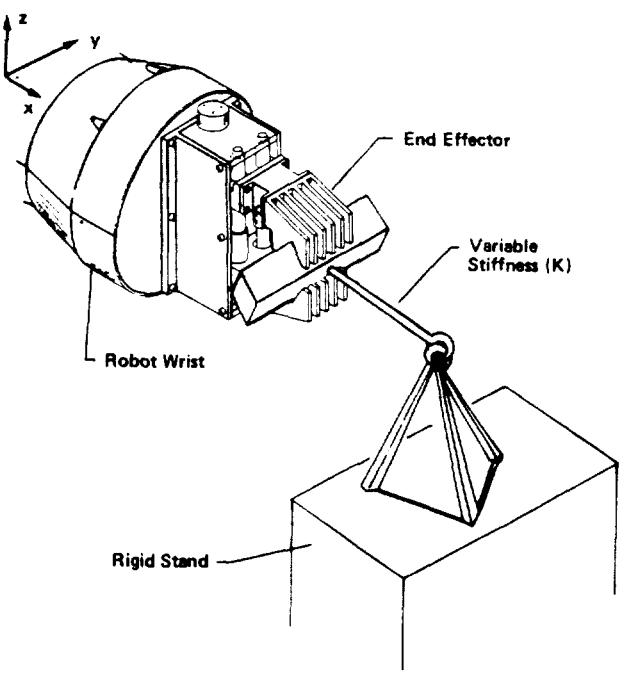


Figure 7. *Experimental verification test fixture.*

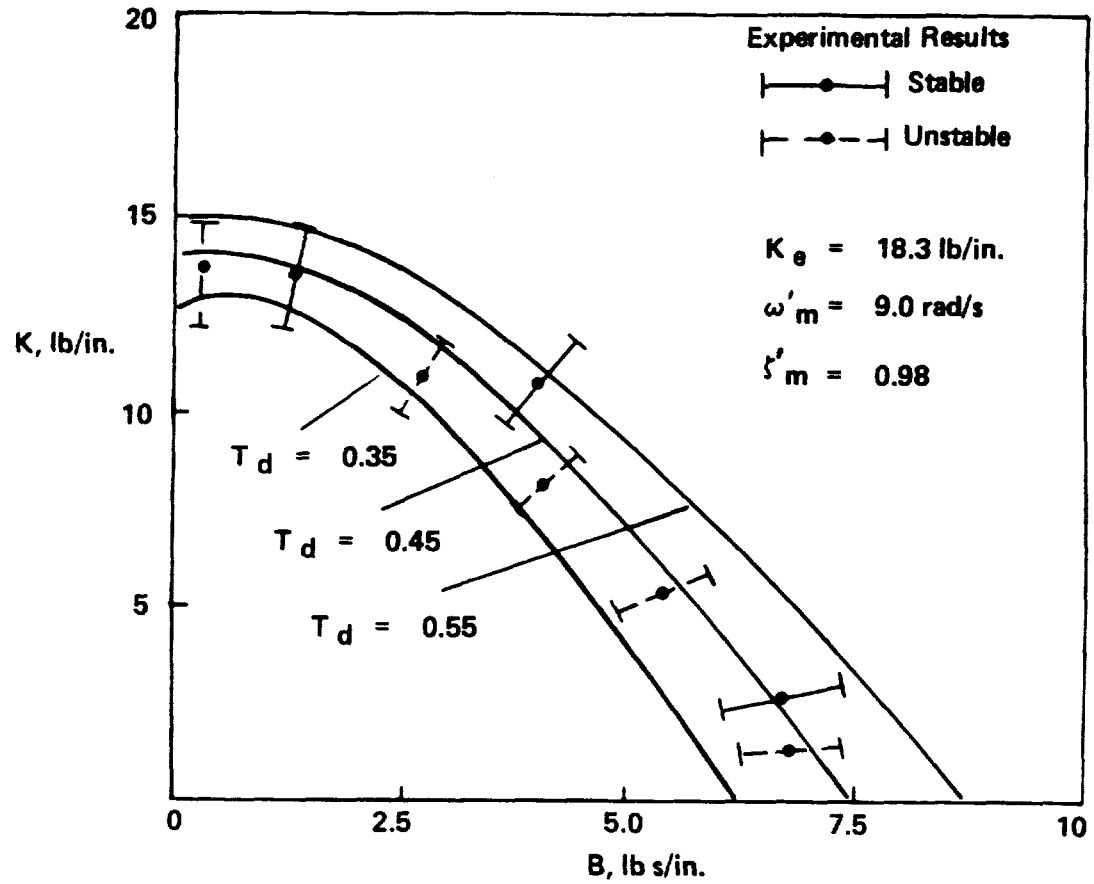


Figure 8. *Comparison of experimental and theoretical results.*

The essence of the problem is now clear---time delays and manipulator dynamics can cause undesired behavior, which prevents realization of very small manipulator impedances. More exact correspondence with measured data will depend on more detailed modeling, but the basic intuition has been established.

4. IMPLICATIONS FOR SPACE-BASED TELEROBOTS

One of the most desireable aspects of good bahavior during interaction between manipulator and environment is the reduction of contact forces. This provides a measure of safety in general, but also reduces distortion in components and disturbances to the overall system. Force disturbances may play a crutial role in space manipulation systems, particularly when supported by larger, more flexible platforms, such as the RMS manipulator.

If we consider the manipulator to have stiffness K and the environment (task) to have stiffness K_e along a particular cartesian direction, then the resulting force due to a change in the manipulator command position, x, is given by

$$F = \frac{KK_e}{K + K_e} x$$

This represents the force that would occur if the position of the task were not known precisely, and the command attempted to position the end effector inside the task surface. Note that when K is small compared to K_e, the resulting force is given approximately by

$$F \approx Kx$$

In fact, if $K < K_e/10$ then the resulting force is less than 1/11 of the force when the manipulator is perfectly stiff, ie. when $F = K_e X$. Thus, a usual objective for the control scheme is to require that K is small compared to the environmental stiffness in those directions where K_e is large. From Figure 6, stability of the system when this condition on K is imposed requires that the specified damping B lie near the K=0 intercept of the stability boundary. That is, relatively large values of damping are required. If other performance restrictions are imposed, such as the absense of overshoot in the response to surface contact, then even more damping is required. See [11] for more discussion of performance issues.

Even if static forces are sufficiently reduced by specifying small K, dynamic forces due to motion can cause large transient forces when B is large. In the case where the specified manipulator impedance is Bs + K, the force at the contact point is given by

$$F = K_e \, x \, \frac{(Bs + K)}{Bs + K + K_e}$$

If K_e is large compared to K, and large compared to Bs up to some maximum frequency of interest, then this is given approximately by

$$F \approx (Bs + K)x$$

From the stability analysis results for the case where K_e = 5 lb./in., and K = 0.5 lb./in., and the time delay is 200 ms a reasonable value for B would be 2 lb. sec./in. Given a position profile X(t), the resulting force at the contact point can be computed using the approximate relation above.

The full effect of large damping values can be more easily seen in the following example. As in the experimental tests of section 2, let the manipulator extract a drawer from a fixture along some direction which is slightly different from the nominal direction given by the task geometry. Figure 9 shows a typical smooth position trajectory in the pull direction x. The misalignment causes a similar position trajectory in the y direction, which is interpreted as a position error with respect to the task geometry. Due to the small alignment error, the y position only changes 1 inch, while the drawer is pulled 1 foot in the x direction. The forces due to this 1 inch position error are shown in Figure 10 for three manipulator impedances. The first represents the case where no control compensation is present. There, the force is simply K_e times the position error. The second case represents the ideal case where the manipulator is programmed to behave as a pure stiffness K. Since K is 1/10 the size of K_e, the force is approximately given by K times the position error. Note the order of magnitude reduction in force compared to the uncompensated case. The third case represents the practical situation where significant damping is also required to retain stability. While static forces are the same as in the ideal (stiffness only) case, the transient forces exceed those of even the uncompensated case.

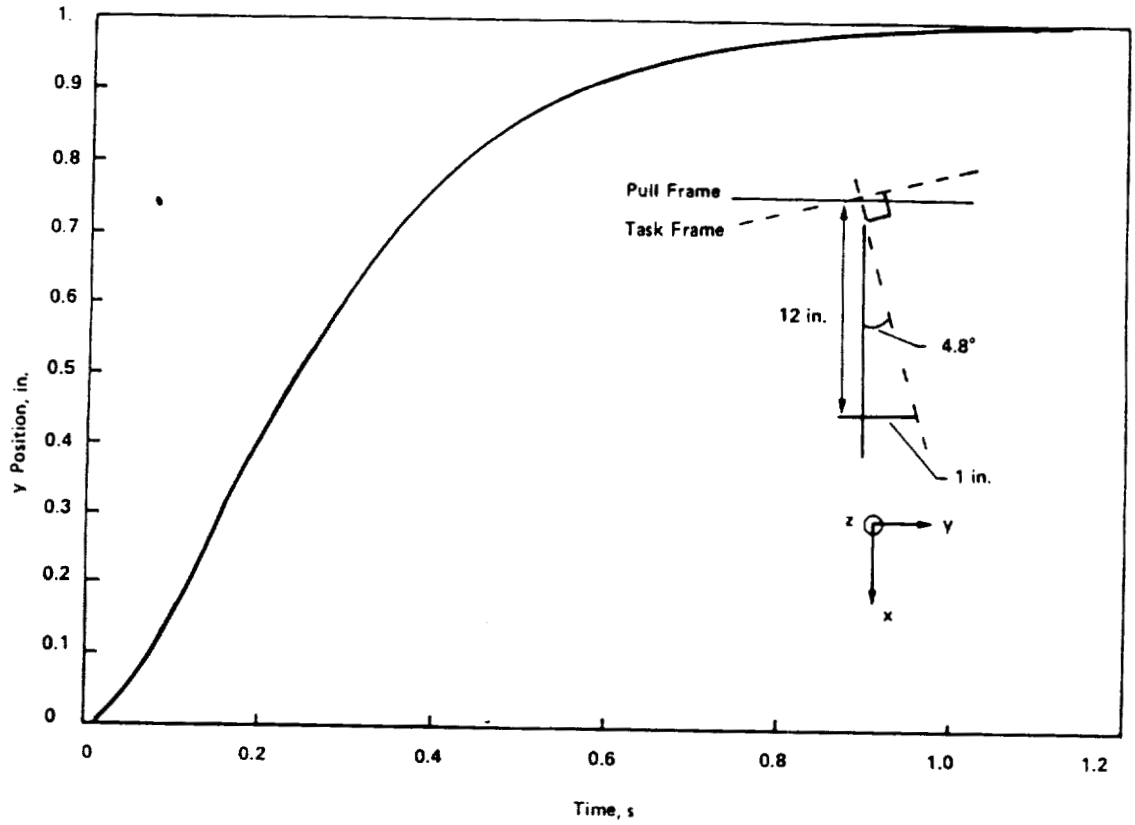


Figure 9. *Uncompensated y position, misaligned pull.*

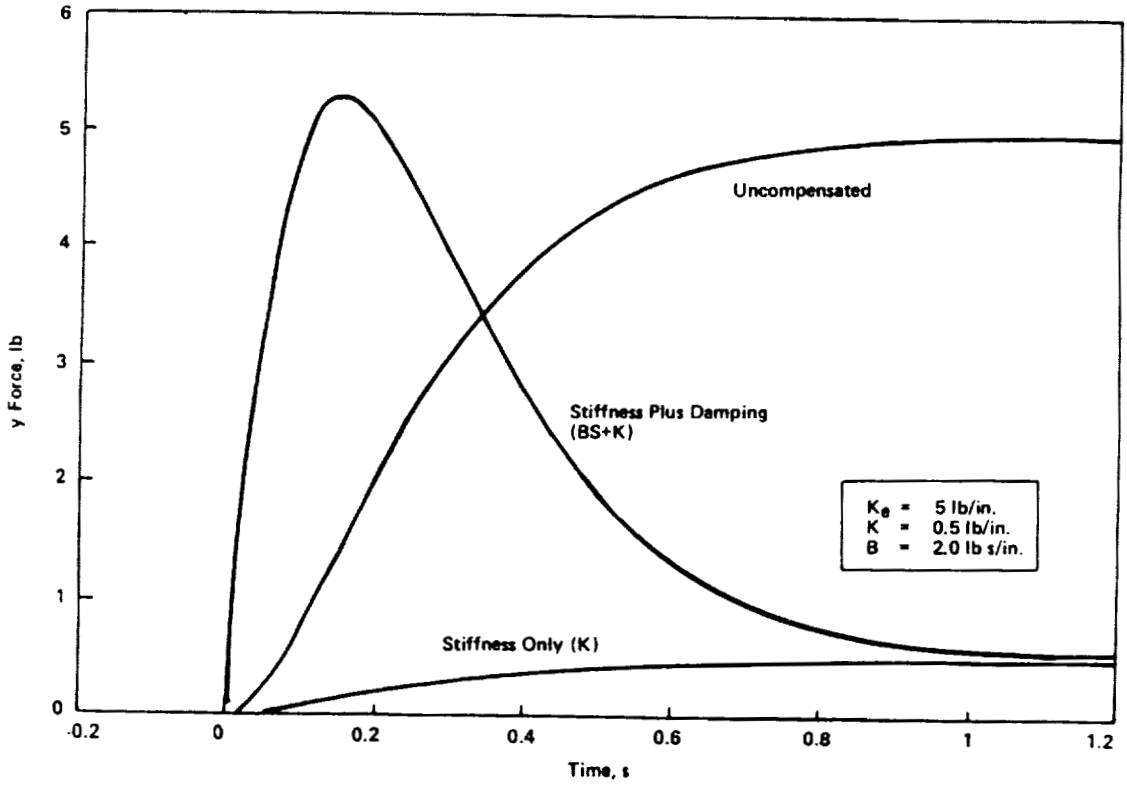


Figure 10. *Force versus time, for three manipulator impedances.*

Since the transient forces are proportional to the speed of the motion, these forces can be reduced by slowing the motion. Hence, given a desired level of force reduction compared to the uncompensated case, the stability constraint can be viewed as imposing limitations on speed of task execution. This is the essential difficulty with large amounts of damping in the impedance specification when the control is implemented via computer.

When implemented via force reflection teleoperation, other implications of large amounts of damping appear. There are two ways of supplying the required damping at the hand controller. The most obvious is to require the human operator to supply it. Alternatively, feedback control can be implemented local to the hand controller to add damping. In the first case, large amounts of damping are fatiguing because the operator must constantly keep muscles tensed, and concentation must not lapse. The difficulty involved was indicated by the experimental data and discussion in Section 2. In the second case, any motion not due to sensed interacition forces must be supplied by the operator in opposition to the hand controller damping. This can also be quite fatiguing if any prolonged motion is required. For example, the 2 lb. sec/in. level requires the exertion of 2 pounds of force in order to move the end effector at a speed of 1 in./sec. Practically, if motion is required when no environmental contact is made, this damping should be reduced or removed. The emphasis is then placed on some mode switching technique which depends on the detection of contact, proximity sensing, etc. One operational difficulty with hand controller damping is that surface contact can be difficult to "feel", since forces on the hand are not primarily dependent on contact forces. Clearly, many of these issues will require more extensive study, testing, and operator evaluation.

One final implication of the results presented above concerns the source of time delay in the control loop. An advantage in this respect can be obtained by implementing impedance control in computers local to the manipulator. Since the command updates depend on forces in simple ways, safety and error checking can be relatively fast. Also, communication delays can be reduced to insignificant levels. However, the use of teleoperation implies closure of the control loop via a remote control station. Communication delays, additional hand controller kinematic computations, mechanical properties of the hand controller, and relatively unpredictable operator reactions which require extensive safety monitoring all contribute to increasing time delays. These translate into increasing limitations of the control scheme as discussed above. What is obviously needed is a battery of laboratory tests using space--realistic manipulators, hand controllers, and functional tasks to determine the relative advantages of various control methods and implementation details. The results presented here represent initial data on a specific class of control approaches, and should be considered only as a basis on which to build.

REFERENCES

[1] Hill, J. W., Study to Design and Develop Remote Manipulator Systems, Quarterly Reports 5 and 6, Stanford Reserach Institute, January 1977.

[2] Hogan, N., "Impedance Control: An Approach to Manipulation," ASME Journal of Dynamic Systems, Measurement and Control, March, 1985, pp 1-24.

[3] Whitney, D. E., "Force Feedback Control of Manipulator Fine Motions," ASME Journal of Dynamic Systems, Measurement and Control, June, 1977, pp 91-97.

[4] Mason, M. T., "Complinance and Force Control For Computer Controlled Manipulators," IEEE Transacations on Systems, Man and Cybernetics, June, 1981, pp 418-432.

[5] Raibert, M. H., and Craig, J. J., "Hybrid Position/Force Control of Manipulators," ASME Journal of Dynamic Systems, Measurement and Control, June, 1981, pp 126-133.

[6] Salisbury, K. J., "Active Stiffness Control of a Manipulator in Cartesian Coordinates," Proceedings 1980 Conference on Decision and Control, December, 1980, pp 95-100.

[7] Maples, J. A. and Becker, J. J., "Experiments in Force Control of Robotic Manipulators," Proceedings IEEE International Conference on Robotics and Automation, April 1986, pp 695-702.

[8] Septien, T. M., et al, "Control of Tool/Workpiece Contact Force with Application to Robotic Deburring," IEEE Journal of Robotics and Automation, Vol RA-3, No. 1, February, 1987, pp 7-18.

[9] Lawrence, D. A. and Stoughton, R. M., "Position-Based Impedance Control: Achieving Stability in Practice," to appear Proceedings AIAA Conference on Guidance, Navigation, and Control, August, 1987.

[10] Kazerooni, H., Sheridan, T. B., and Houpt, P. K., "Robust Compliant Motion for Manipulators, " IEEE Journal of Robotics and Automation, Vol. RA-2, No. 2, June 1986, pp 83-92.

[11] Chapel, J. D. and Lawrence, D. A., "Stability Analysis For Alternative Force Control Schemes as Applied to Remote Space Teleoperation," Proceedings, 10th Annual AAS Guidance and Control Conference, January, 1987.

A Shared Position/Force Control Methodology for Teleoperation

Jin S. Lee

RCA Corporation, Advanced Technology Laboratories
Moorestown, New Jersey 08057

ABSTRACT

A flexible and computationally efficient shared position/force control concept and its implementation in the Robot Control C Library (RCCL) are presented from the point of teleoperation. This methodology enables certain degrees of freedom to be position-controlled through realtime manual inputs and the remaining degrees of freedom to be force-controlled by computer. Functionally, it is a hybrid control scheme in that certain degrees of freedom are designated to be under position control, and the remaining degrees of freedom to be under force control. However, the methodology is also a shared control scheme because some degrees of freedom can be put under manual control and the other degrees of freedom put under computer control.

Unlike other hybrid control schemes, which process position and force commands independently, this scheme provides a force control loop built on top of a position control innerloop. This feature minimizes the computational burden and increases disturbance rejection. A simple implementation is achieved partly because the joint control servos that are part of most robots can be used to provide the position control innerloop.

Along with this control scheme, several menus have been implemented for the convenience of the user. As a result, the user can define a center of compliance on any point in the workspace and a selection matrix that assigns certain axes under position control and other axes under force control. Finally, the user can define force gains in the force control strategy to ensure overall system stability and to avoid overshoots and oscillations in the robot motions.

The implemented control scheme has been successfully demonstrated for the tasks of hinged-panel opening and peg-in-hole insertion.

INTRODUCTION

Over the past several years, many compliant control techniques have been proposed and developed to extend robot applications to a wide variety of tasks requiring compliance. Compliance is almost inevitably required when the robot manipulator comes into contact with the environment and its position is constrained. Slight position errors of the robot may produce enormous forces and cause serious damage both to the robot and the workpiece. While compliant motion can be provided by a passive mechanical compliance device such as a remote center compliance (RCC), the development described in this paper is centered around digitally

implemented active compliance, which is more flexible and can be reconfigured in realtime.

Most active compliance techniques, however, have been developed for autonomous operations, in which a complete task is executed under computer control. Automated compliant control, while effective for structured tasks, does not provide the operator with direct control of the position at which the operation is performed. As a result, it is not suitable for unstructured tasks. Autonomous control techniques at the present time are neither intelligent nor reliable enough to perform any but the simplest and most routine tasks.

In teleoperation, force reflecting master/slave hand controllers have been used for many years to control remote robots in unstructured hazardous environments [1]. They provide an operator with an accurate experience of forces encountered by the robot and the illusion of doing the task directly. They are widely used in the nuclear and undersea environments, where no restrictions are imposed on size, and no communication time delays exist between control station and robot manipulator. These devices, however, cannot be readily extended to unmanned-orbit servicing tasks because of limitations on the size of the control station and communication time delays between control station and servicer.

Another alternative in teleoperation is the use of the resolved rate joystick. The joystick's compactness and the operator's familiarity with it makes it a logical candidate for use in telerobotic tasks. It is, however, a strict rate-control device without any force feedback and, therefore, not suitable for compliant control tasks.

This paper reports on the extension of compliant control techniques to teleoperated systems and the development of new concepts to accomplish compliant control tasks under joystick control. Developed at RCA's Advanced Technology Laboratories, the control mode presented is termed "shared position/ force control" or simply "shared control". Under shared control, the operator retains realtime control of robot motion while leaving the responsibility for compliance to a computer local to the robot.

An important potential application of shared control is in space telerobotic servicing. When the servicer is controlled from a shuttle or from the space station, the use of a joystick in shared control reduces the physical size of the master-slave hand controller, thereby making it possible to reduce the size of the control station. When the servicer is controlled from a ground station, excessive time delays (estimated at two to five seconds), resulting from space and ground communication links, prohibit realtime control of the servicer.

Shared control may be an important complement to the incremental "move-and-wait" tactic currently demonstrated for motions without force feedback. With an on-board controller, shared control provides for task adaptability at the work site, in which the tool, under local control, adapts to any excessive force. Initial investigations and tests of shared control techniques have opened promising new possibilities for

more efficient and safe teleoperation control for both short-term and long-term satellite servicing.

Compliant Control

The behaviour of compliant motion can be systematically described by specifying a center of compliance and its compliance frame. A center of compliance is a point in the workspace in which a force applied to the point causes a motion in the direction of the force. Its compliance frame is an orthogonal coordinate frame, with its origin at the center of compliance. Thus, a compliant task is described in terms of desired position trajectories or force/torque profiles for each of the compliance frame axes. In many cases, the center of compliance is placed at the center of the end-effector.

There are two prevalent approaches in active compliant control: explicit feedback and hybrid control [2]. Explicit feedback specifies a linear relation between sensed forces* and the corresponding positions* accommodations [3, 4]. This is typically modelled by the equation:

$$f = K(p - p0)$$

where:

- f = sensed force
- p = the current position
- $p0$ = the predefined nominal position
- K = the gain matrix that relates sensed forces linearly to deviations from the nominal position.

This explicit feedback scheme is built around the joint servos, which process joint set points and drive joint actuators of the robot (Figure 1). It is computationally efficient and simple to implement on the joint servos provided with most robots. However, the force control gains must be appropriately selected for each task to ensure system stability and desirable performance.

The hybrid control approach, on the other hand, processes position and force commands independently through their own control loops (Figure 2) [5, 6]. It first selects certain degrees of freedom to be under position control and the others to be under force control, and then drives each actuator according to the sum of its contributions, whether force or position. The hybrid control scheme is computationally expensive but conceptually elegant because it can process force commands under any environment. It usually involves significant modification of existing joint servos to implement a force servo loop. Because of joint friction/stiction in some robots, force servo loop are harder to design and implement.

---

*In this paper, force implies force and torque and position implies position and orientation.

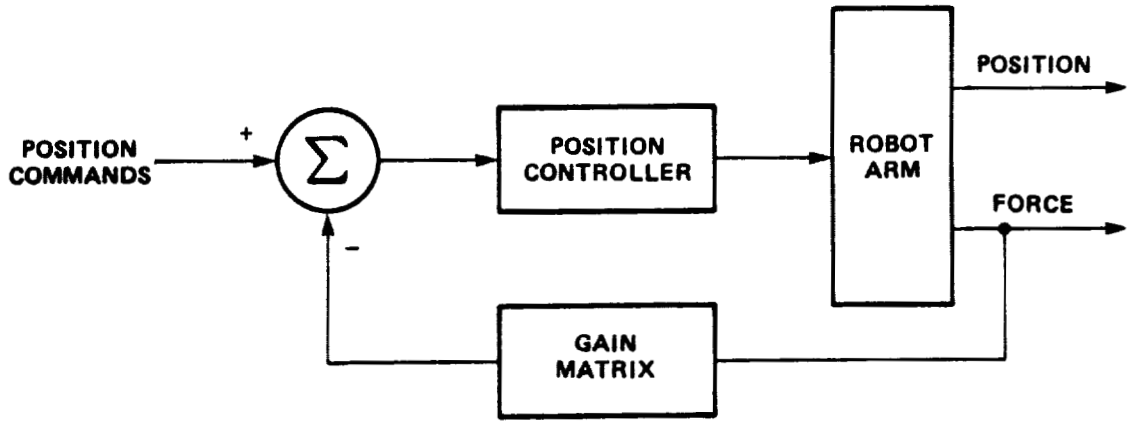


Figure 1. Explicit feedback block diagram.

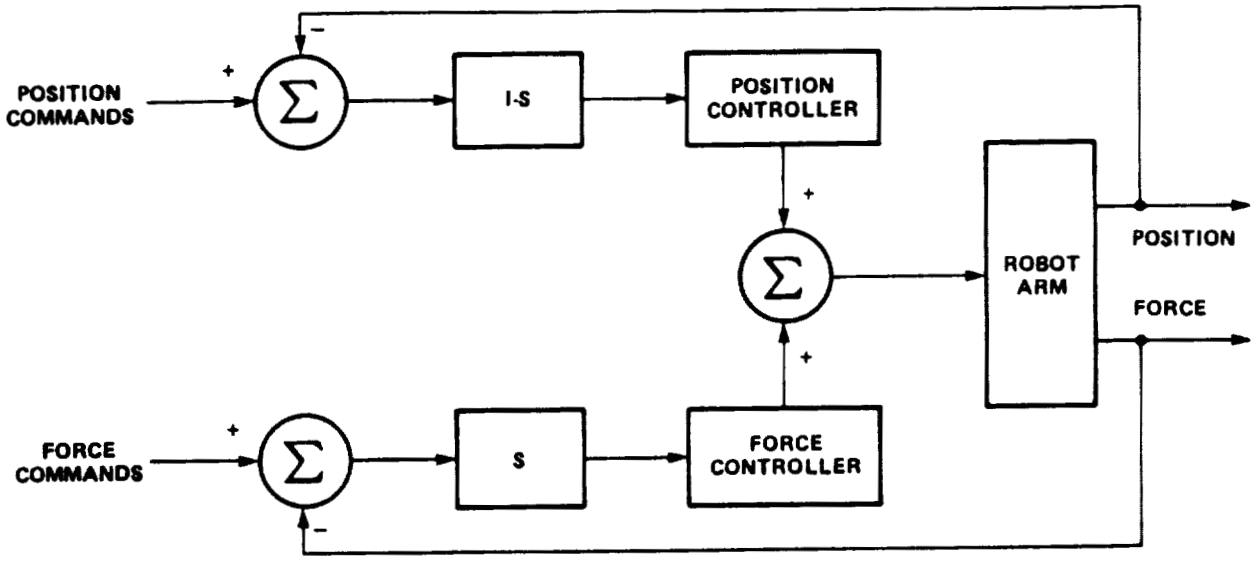


Figure 2. Hybrid control block diagram.

The compliant control algorithm in the shared position/force control is functionally a hybrid control scheme in that certain axes of the compliance frame are assigned for position control (rate control); while the other axes are assigned for force control. But the algorithm's force control loop is built on top of its position innerloop; hence, it is technically an explicit feedback scheme. As in explicit feedback, it is computationally efficient and simple to implement. The fast joint-based position innerloop automatically rejects disturbance torques arising from any source, even gravity and joint frictions/stictions [7].

Shared Position/Force Control

Figure 3 outlines the Cartesian-based shared control scheme currently used at RCA's Advanced Technology Laboratories to control a PUMA 762 robot. The implemented algorithm allows the operator to control selected position axes through realtime manual inputs or through predefined trajectories providing fully automated compliant task functions. The desired position values are derived from the joystick rate commands by integrating them. The control system itself consists of two feedback control loops: the inner PID joint servo loop and the outer force feedback loop. For the innerloop, it uses the standard PUMA 762 PID joint servos with a sampling rate of approximately 500 Hz. For the outer loop, it feeds sensed forces at the wrist sensor back to the position control loop with a sampling rate of approximately 36 Hz.

The position control scheme simply receives six rate values from the joysticks, selecting only a subset of those that correspond to position axes. Position axes are determined with a 6x6 diagonal selection matrix S. Each diagonal element of S, which is Boolean, is associated with each axis of the compliance frame. When its ith diagonal element is 0, the corresponding axis is under position control. When it is 1, the axis is under force control. The selected rate values are appropriately scaled with a 6x6 diagonal matrix K, which determines the desired robot velocity. The scaled rates are combined with the compensatory rates described in the text that follows, to form the combined rates at the compliance frame. These rates are integrated to derive the next desired Cartesian set points, which are then resolved to the robot end frame. These set points are then transformed to the joint set points, which are then input to the inner PID joint servos to drive the joint actuators of the PUMA 762 robot.

The force control loop is implemented around the inner joint PID servo loop, which is shared by the position control loop. The outer force control loop receives the predefined bias forces and drives position changes to produce those forces in the selected directions. It subtracts the sensed forces from the commanded forces and updates the force errors in the selected directions. The sensed forces at the compliance frame are computed from the sensed forces at the wrist force sensor and from the transformation T between the compliance frame and the sensor frame. The force control loop then scales the force errors via a force gain matrix to compute the compensatory rates. As described earlier, these rates are combined with the scaled rates to form the combined rates. The combined rates are then integrated to compute the next Cartesian set points. The gains in the force gain matrix must be

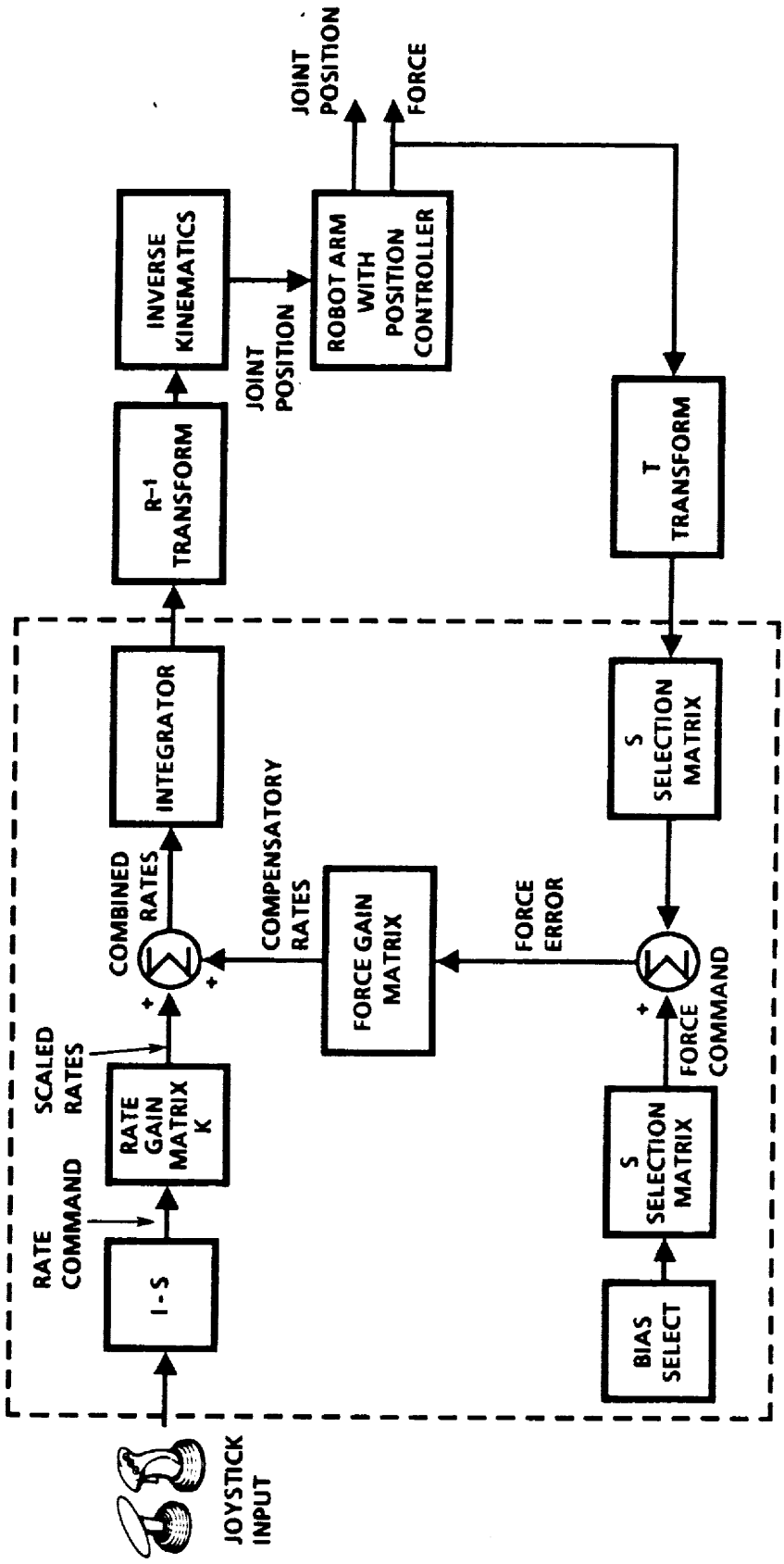


Figure 3. Shared position / force control scheme.

carefully tuned, not only to ensure system stability but also to prevent the closed loop system from overshoot and oscillation.

The inner joint servo loops interpolate between the commanded joint set points and the current joint set points, and they update the joint error actuating signals from the joint-interpolated set points and the joint encoder values. Next, each of these joint errors is regulated by each joint PID servo control algorithm. The gains of the PID control loops are tuned to make the closed loop systems sufficiently overdamped so that small perturbations and/or disturbances do not jeopardize the closed loop system stability.

Implementation in RCCL

RCCL is a general-purpose robot programming system originally developed at Purdue University by V. Hayward under the direction of R. P. Paul [8, 9] and later improved by J. Lloyd at McGill University [10]. Currently at RCA, RCCL is installed on a microVAX II to control a PUMA 762 [11].

RCCL's environment consists of two levels: the planning level and the control level. The control level is a C language software facility under the UNIX operating system that generates realtime Cartesian trajectories. Built on top of this is the planning level, which is a collection of C primitives and data structures that provide robot motion in Cartesian coordinates and joint coordinates. RCCL allows a user to modify the Cartesian trajectory in realtime by means of control level user functions. This facility is used to implement a shared control algorithm in the RCCL.

Figure 4 outlines the RCCL implementation of the shared control algorithm. In the RCCL planning-level user program, the user defines a simple transformation equation, which drives the robot according to external inputs such as joystick commands and sensed forces. The equation is described by T6*E = G, where T6, E, and G are the homogeneous transforms describing, respectively, the robot end, the compliance center from the robot end, and the desired set point of the compliance center. Predefined at the current compliance center position, G is functionally defined and computed in realtime in the control-level user functions. When the execution begins, the control function first collects joystick commands and sensed forces from the LSI 11/73, and then runs the control algorithm to compute the new combined rate V and the corresponding Cartesian set point G from the equation G = G*V. The RCCL trajectory generator then updates a new goal robot end transform T6 according to the new G and computes corresponding joint set points to drive the joint actuators.

Demonstrations

The shared control algorithm has been demonstrated for two compliance tasks: hinged-panel opening and peg-in-hole insertion.

(1) Hinged-Panel Opening

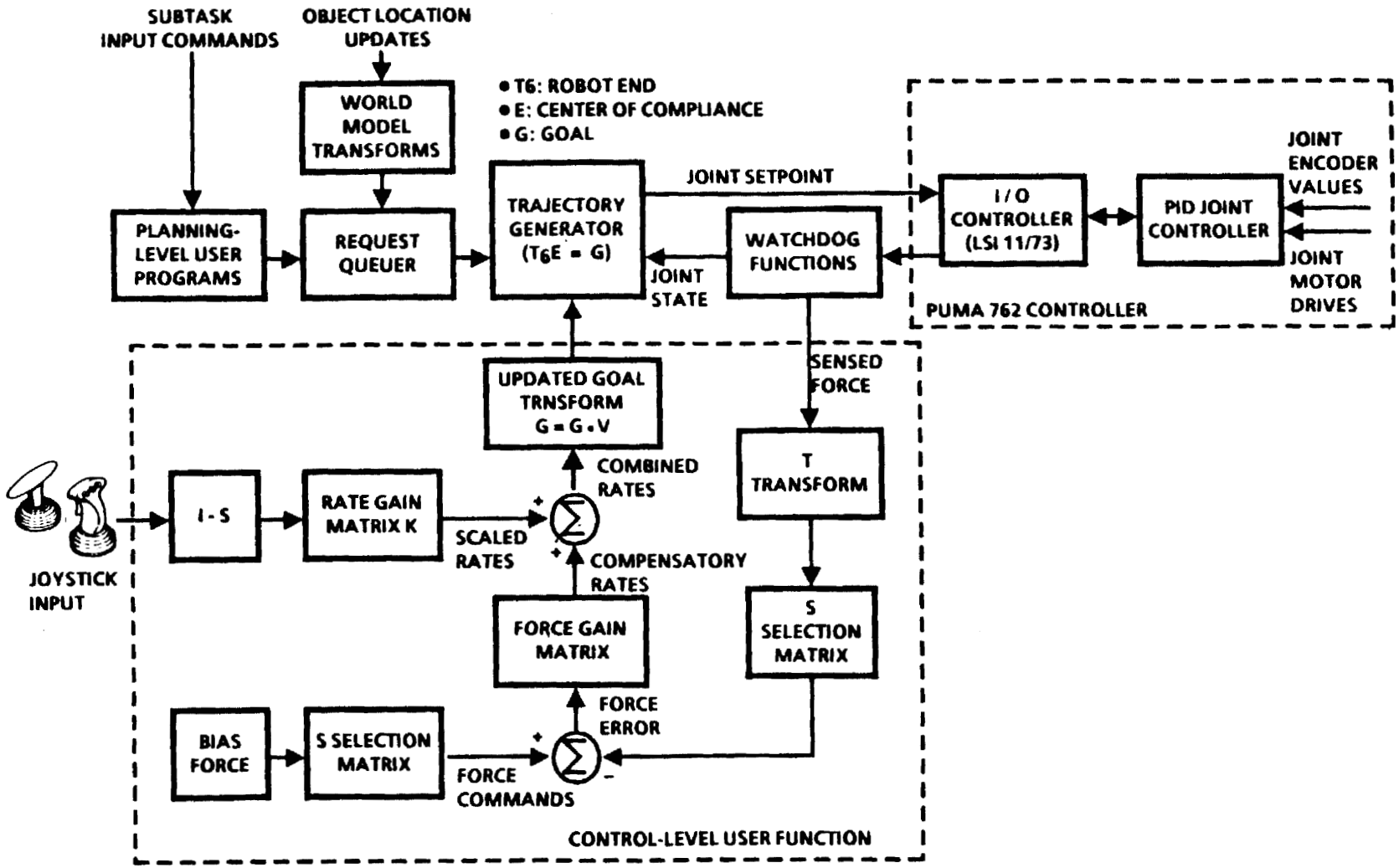


Figure 4. Shared position / force control algorithm implemented in the RCCL.

Figure 5 illustrates a hinged-panel opening task consisting of two subtasks: aligning the gripper to the doorknob and opening the hinged-panel. The compliance frame is defined at the center of the gripper, with the z-axis as the robot approach direction and the y-axis as the parallel gripper open/close direction.

When the robot grips the doorknob, a misalignment always occurs between the gripper and the doorknob. This misalignment is detected by a wrist sensor in terms of nontrivial force values along certain axes. By selecting these axes to be under force control, compensating positions are generated to accommodate to the geometric constraints of the doorknob.

When the robot opens the hinged-panel, the operator is concerned only with how far the hinged panel is open as he turns the doorknob in a direction normal to the current position, that is, along the z-axis of the compliance frame. The other axes are left to comply. A single axis of the joystick is then used to rate control the z-axis motion. Other motion adjustments are performed automatically to relieve forces in realtime. The force gain matrix must be carefully chosen to keep the gripper fully aligned to the doorknob. Otherwise, the misalignment may be increased, causing the system to be unstable.

The operator can arbitrarily stop the hinged-panel at any position and resume motion under manual control at will. The task is performed in realtime, with the natural motions and flexibility inherent in manual control. The same algorithm can be used to close the hinged-panel. This differs from automated compliant control, in which the program must be aborted to stop the motion, and can only be resumed by issuing a restart motion.

(2) Peg-In-Hole Insertion

The classical peg-insertion task is representative of assembly operations likely to be undertaken by robotic servicers. Figure 6 illustrates a peg-in-hole insertion task with a round tapered peg and a round hole with approximately a 1-mil clearance. The task is performed in two phases: the taper-crossing phase and the side contact phase. The compliance frame is defined at the end of the tapered peg, and again the z-axis is the robot approach direction. Here, the z-axis is a natural choice for position control, since the depth of insertion is the primary task parameter.

The operator approach the point of insertion in a purely manual mode. With the initial surface contact having been made, the taper-crossing phase begins. Still in the manual mode, the operator begins insertion with the rate under joystick control. Here, accommodation is made only in x and y-axes to slide the peg into the center of the hole. This phase in shared control mode continues until jamming occurs, which is indicated by excessive forces in the z direction. Then, the side-contact phase begins.

In this phase, angular alignments are the most important adjustments to release the peg from jamming and enable another insertion attempt.

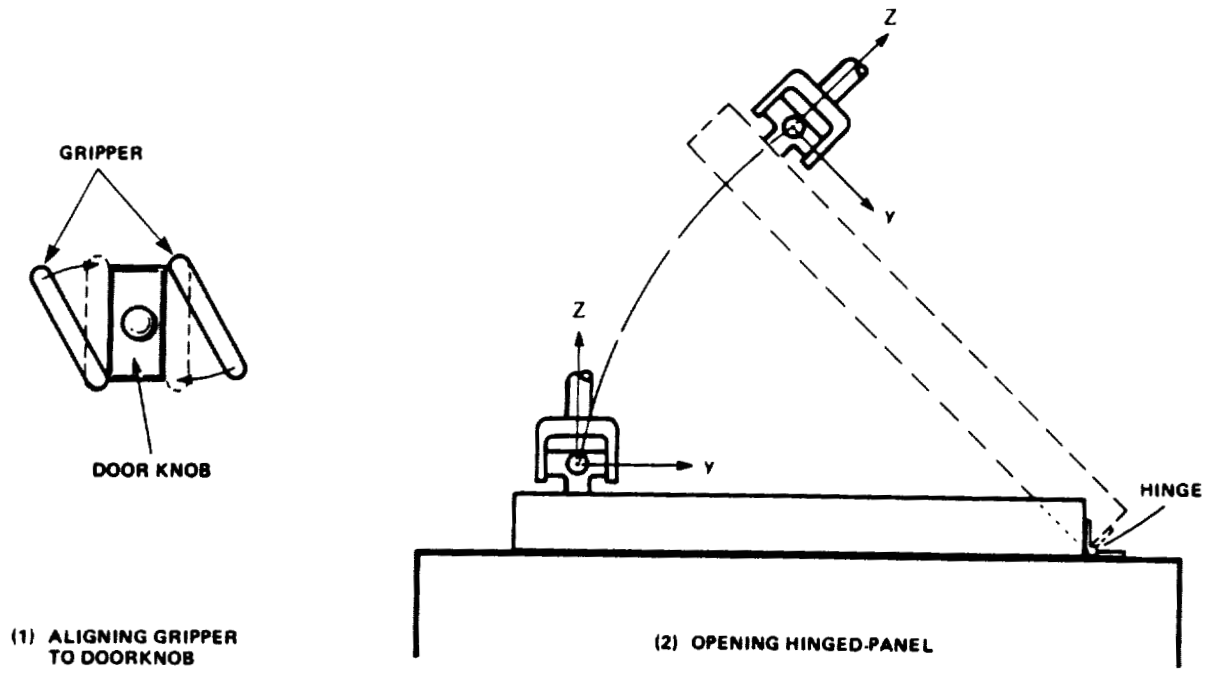


Figure 5. Hinged-panel opening task.

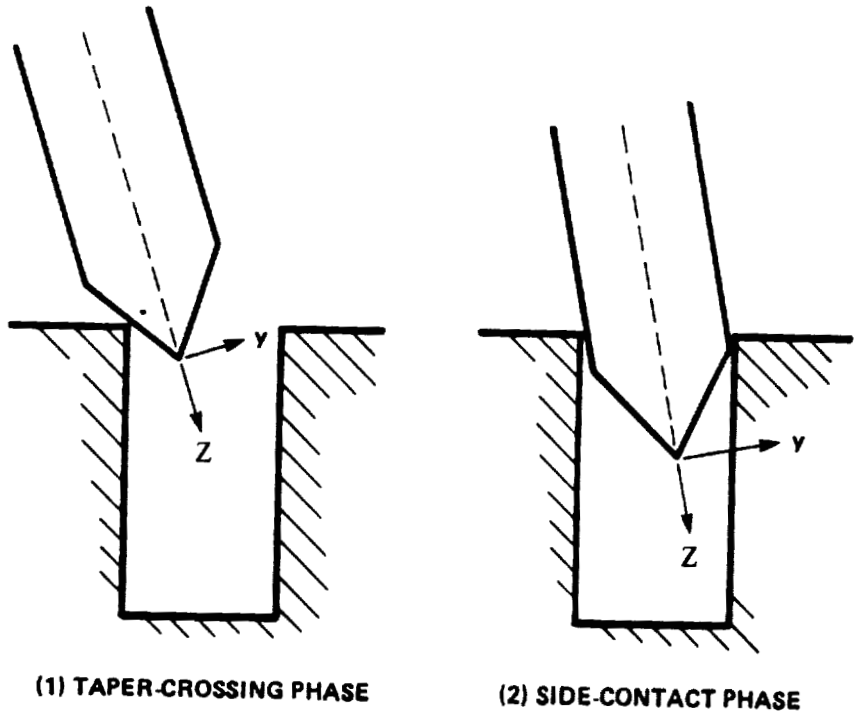


Figure 6. Peg-in-hole insertion task.

Except for the z-axis, which is under manual control, all other axes are selected for force control to provide both angular alignments and positional accommodations. With these arrangements, the operator can continue insertion, with all other axes complying to the geometric constraints of the hole. Because of tight geometric constraints, the force gains can be lowered significantly to boost system stability and performance.

Recommendations for Further Research

Under the current shared control scheme, force gains must be readjusted for each task to ensure stability and the efficient performance of the control system. To improve the current scheme, an adaptive algorithm should be developed to adjust these gains automatically for each task. The force control algorithm in the current shared control scheme can also be replaced by Raibert and Craig's hybrid control [5], where the adjustment of force gains is not necessary. This will involve the design and implementation of a PID-type force control algorithm similar in structure to joint position servos with a 500 Hz sampling rate.

References

1. A. K. Bejczy and J. K. Salisbury. "Kinesthetic Coupling Between Operator and Remote Manipulator," Proceedings of ASME Conference on Computer Technology, Volume 1. San Francisco, CA, August 1980.

2. M. T. Mason. "Compliance and Force Control for Computer Controlled Manipulators," IEEE Transactions on Systems, Man, and Cybernetics SMC-11, June 1981. pp. 418-432.

3. D. E. Whitney. "Force Feedback Control of Manipulator Fine Motions," Journal of Dynamic Systems, Measurement, and Control, June 1977. pp. 91-97.

4. J. K. Salisbury. "Active Stiffness Control of a Manipulator in Cartesian Coordinates," IEEE Conference on Decision and Control, November 1980.

5. M. H. Raibert and J. J. Craig. "Hybrid Position/Force Control of Manipulators," Journal of Dynamic Systems, Measurement and Control, 102, June 1981. pp. 126-133.

6. O. Khatib. "The Operational Space Formulation in Robot Manipulator Control," 15th ISIR, September 1985. Tokyo, Japan.

7. J. A. Maples and J. J. Becker. "Experiments in Force Control of Robotic Manipulators," Proceedings of IEEE Conference on Robotics and Automation, Volume 2, April 1985. San Francisco, CA.

8. V. Hayward and R. P. Paul. "Introduction to RCCL: A Robot Control C Library," IEEE First International Conference on Robotics, June 1984. Atlanta, Georgia.

9. V. Hayward and R. P. Paul. "Robot Manipulator Control under UNIX: RCCL, A Robot Control C Library," International Journal of Robotics Research, 5(3), Fall 1986.

10. J. E. Lloyd. "Implementation of a Robot Control Development Environment," Master of Engineering Thesis, Department of Electrical Engineering, McGill University, December 1985. Montreal, Canada.

11. J. S. Lee, S. Hayati, et al. "Implementation of RCCL, a Robot Control C Library on a MicroVAX II," Proceedings of the SPIE Conference on Advances in Intelligent Robotics Systems, Vol. 726, October 1986. Cambridge, MA.

MULTIPLE SENSOR SMART ROBOT HAND WITH FORCE CONTROL

Richard R. Killion
Lee R. Robinson
Antal Bejczy

JET PROPULSION LABORATORY
California Institute of Technology
Pasadena, California

I. INTRODUCTION

Analysis of anticipated space assembly, servicing and repair tasks to be performed by robot arms motivated the work at the Jet Propulsion Laboratory (JPL) for the design and development of multifunctional and smart robot hands. Here the term "multifunctional" refers to the hand's mechanical capabilities, while the term "smart" refers to the hand's sensing and control capabilities. The analysis also led to the conclusion that an evolutionary approach to the design and development of robot hands can generate important and needed capability increases. The first step in this evolutionary development effort was the consideration of one degree-of-freedom parallel claw end effectors equipped with force/torque balance and grasp force sensors, and capable of being servoed in position, rate, and grasp force modes of control.

This paper describes a smart robot hand developed at JPL for the Protoflight Manipulator Arm (PFMA) at the Marshall Space Flight Center (MSFC). The development of this smart hand was based on an integrated design and subsystem architecture by considering mechanism, electronics, sensing, control, display and operator interface in an integrated design approach. The mechanical details of this smart hand and the overall subsystem integration are described elsewhere (Refs. 1 and 2). In this paper we briefly summarize the sensing and electronics components of the JPL/PFMA smart hand and describe in some detail its control capabilities.

II. THE MECHANICAL HAND

II.1 Requirements

The smart hand was designed for and integrated with the PFMA to perform the following specific tasks:

Task 1: Mate and demate a standard fluid coupling.

Task 2: Open and close an access panel by turning a wing nut.

Task 3: Remove and replace a battery module by grasping a square beam.

Task 4: Deploy and retrieve a telescoping vertical antenna.

The gripper's intermeshing claws were designed to grasp square beams (as attached in Orbital Replacement Units) as well as round and oval beams. In addition, a graphics display subsystem provides sensor information to the human operator during task performance. Figure 1 shows the mechanism of the end effector and the overall integrated subsystem including electronics, data handling, display and control input panel.

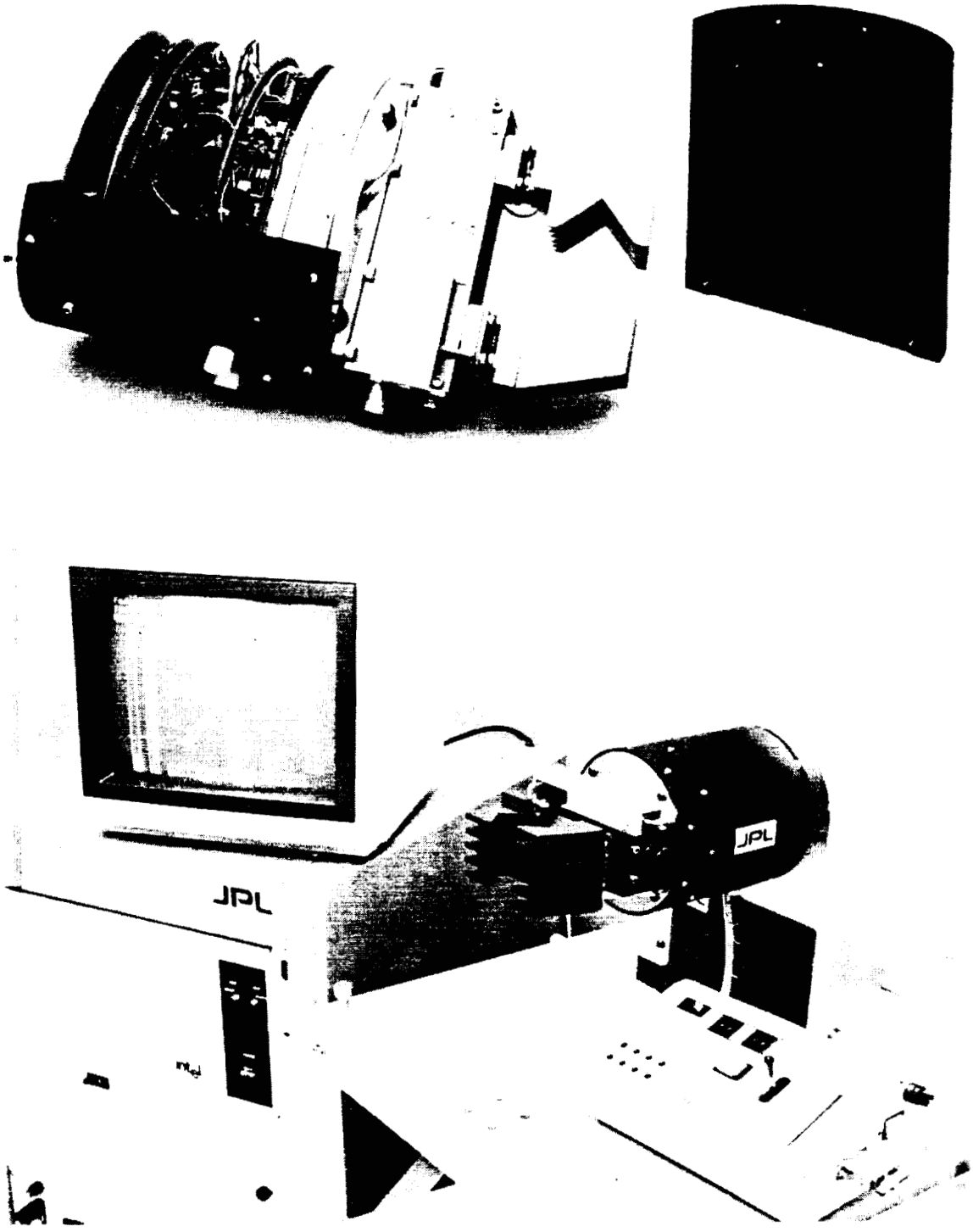


Figure 1. PFMA/JPL Smart End Effector Mechanism, Local Electronics and Overall Subsystem

III. SMART HAND SENSORS

III.1 F/T sensor

During the grasping phase in a zero-g environment, unwanted forces and torques can be detrimental to both the success of the task and the satellite being serviced. To monitor such forces and torques, a force/torque sensor is mounted between the hand and wrist. Semiconductor strain guages are mounted in the Maltese Cross design force/torque sensor with a full-bridge configuration. This configuration results in eight analog guage readings that are read and converted to 12-bit digital values by the Sensor CPU for eventual display for the human operator.

III.2 Clamping Force Sensor

Mounted in the base of each of the two fingers are four semiconductor strain guages in a full bridge configuration. They have been designed to measure up to 120 pounds of clamping force. These readings are converted to 12-bit digital by both the Sensor CPU and Servo CPU. The Servo CPU requires the digitized sensor data in real-time (400 hz) for servoing. The Sensor CPU requires the sensor data to be sent to the Signal Processing CPU for the slower (30 hz) graphics display.

III.3 Tachometer and Potentiometer

Rate and position information is required by the Servo CPU for motor control of the hand closure. Position information is required by the Sensor CPU for eventual display for the human operator.

III.4 Tactile Sensing

For future use, there is a reserve of 32 additional analog channels for force sensing of each of the individual plates of the intermeshing fingers. The force profile along the plates will give misalignment information to reduce torque applied to satellites in a zero-g environment.

III.5 Optical Proximity Sensing

Future plans also include optical proximity sensing. A proximity sensor consists of a photoemitter and a photodector with are focused such that the optic axes of the two converge at a focal point. Distance is determined by the intensity of the light received by the photodetector. This will reduce misalignment before contact, thus reducing unwanted forces and torques during contact.

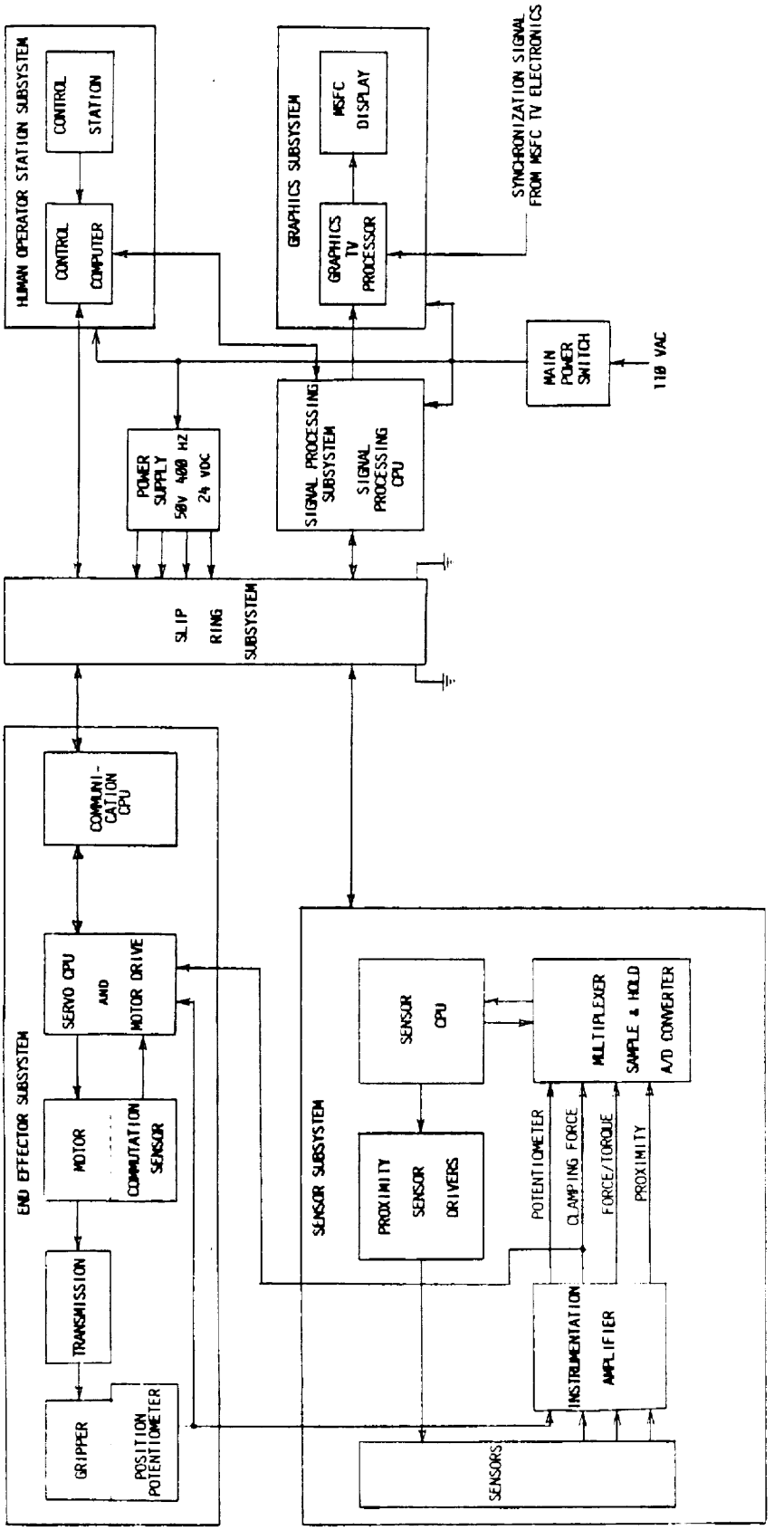


Figure 2. Overall Data Handling for PFMA/JPL Smart Hand

IV. LOCAL ELECTRONICS

The local electronics (electronics in the base of the hand) consists of the Sensor Subsystem and the End Effector Subsystem. See Figure 2.

IV.1 Sensor Subsystem

The Sensor Subsystem consists of the Sensor CPU, Multiplexer, Sample & Hold, A/D converter and instrumentation amplifiers. The Sensor CPU controls the analog multiplexing and the A/D conversion. The multiplexer handles 8 force/torque channels, 2 clamp force channels, and 1 position channel. After the Sensor CPU receives a conversion complete signal, it reads in the data and transmits it on request to the Signal Processing Subsystem via an RS-232 serial link through the Slip Rings.

IV.2 End Effector Subsystem

The End Effector Subsystem constists of two CPUs, the Motorola MC68701 for communications and the Motorola MC68705 for motor servoing.

The Communication CPU receives commands from the Control Computer in the Human Operator Subsystem via an RS-232 serial link through the slip rings. It receives serial data through its on-chip serial port and then checks for transmission errors using a 16-bit checksum. If the command is error-free, it passes the command to the Servo CPU via fast (30 micro seconds) parallel communication.

The Servo CPU executes the command to control the 3-phase D.C. brushless motor. Pulse width modulation and commutation to the motor windings is also done by the Servo CPU.

V. EXTERNAL ELECTRONICS

The External Electronics (electronics not in the hand) consist of the Slip Ring Subsystem, Human Operator Subsystem and the Graphics Subsystem. See figure 2.

V.1 Slip Ring Subsystem

The interface between the smart hand and the PFMA is a rotary slip ring joint. Seven slip ring connections were allocated for power (24 VDC & 20khz 50 VAC), and data communications for both the Sensor Subsystem and End Effector Subsystem.

V.2 Hperator Subsystem

rt of the test system, and also as an independent input ce to the smart hand besides the MSFC Control Compui Control Box was developed. This Control Box has a joys(1-degree of freedom), a slide switch(to command positi 3-way toggle switch (to switch modes), 2 rotary dial hes,(to set rate and force limits), a "hold" buttord status display "LEDs". This box has a Control Computhich is a Motorola MC68705. The joystick, along with ode,clamping force limit and rate limit switches is reathe MC68705. The proper command is generated and sent he Communication CPU via RS-232 across the slip rings e Communication CPU.

V.3 SiProcessing Subsystem

Tignal Processing CPU reads the sensor data from the R serial data stream coming from the Sensor CPU. It thends graphics commands via its Multibus to the Graphiocessor in the Graphics Subsystem.

V.4 Grs Subsystem

Traphics processor receives graphics commands from the l Processing CPU and generates a graphical represion of the force/torque, claw position, and clampirce. In figure 3 the three-axis coordinate system on thehics display shows the resultant forces. The bar graphsng the periphery of the display shows the resultorques due to roll, pitch and yaw. The vertical bars o left indicate gripper opening and clamping force sensedach of the two claws.

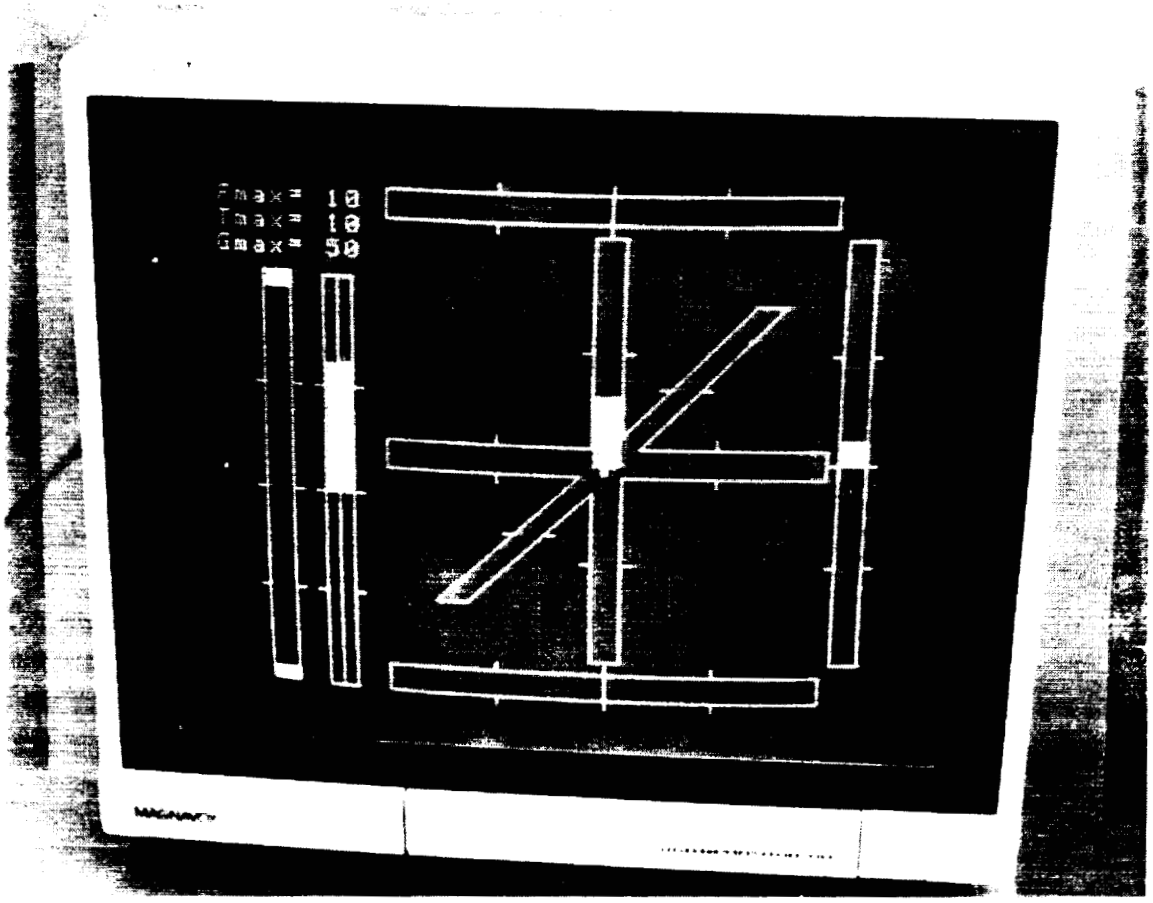


Figure 3. PFMA/JPL Smart End Effector Graphics Display (Left Vertical: e/e Opening-Closing Status; Second Vertical: Grasp Force Sensors Value; Center 2 and a 1/2 Frame: Up-Down, In-Out and Left-Right Forces; Top Horizontal: Roll Torque; Bottom Horizontal: Yaw Torque; Right Vertical: Pitch Torque)

VI. DESCRIPTION OF CONTROL MODES

Three primary control modes are designed into the smart hand. These modes are as follows:

(1) A Position Mode wherein the claws may be positioned to a specified differential opening and which incorporates an autonomous default "backup" mode if a force is detected. The hand reverses direction and stops after 1/8". It then refuses to accept position commands that would cause the same collision. This condition remains until the human operator either changes to rate mode or commands a position to move opposite the direction of the previous collision.

(2) A Rate Mode wherein the claws may be moved at a specified differential rate and which automatically transitions into a Force Mode Control configuration when an object has been encountered which generates a force opposing claw motion.

(3) A Hold Mode wherein a currently existing force which has been generated in Force Mode may be commanded to continuously control claw squeezing force without further command input.

A complete control mode diagram for the smart hand is shown in Figure 4. This diagram details the drive system hardware selected for the final system configuration. In addition, a computer simulation of this system has been generated for use in prediction of control mode performance. After optimizing the smart hand drive system hardware selection for motor torque-speed characteristics, power efficiency and feedback element resolution/dynamic range tradeoffs, consideration of algorithm parameters was undertaken.

The processor utilized as the smart hand servo controller is the Motorola MC68705 operating at 2MHz. It was considered imperative that all control algorithms (with attendent feedback data sampling) operate rapidly compared to the system response. A design target of less than 2.5 msec was considered desireable based upon the inherent need to minimize the buildup of force and energy transfer to the workpiece during conditions of unanticipated contact.

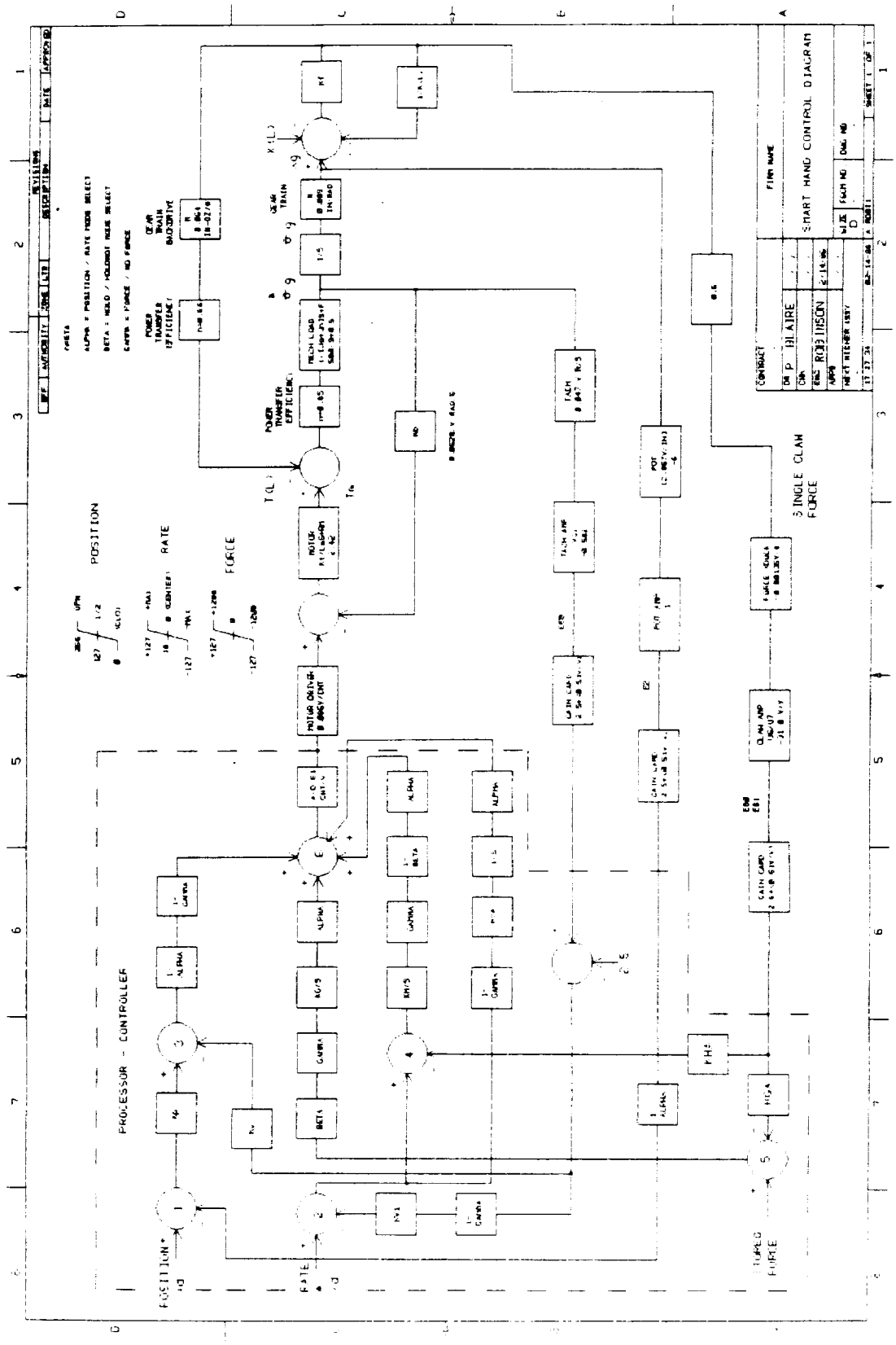


Figure 4. Smart Hand Control Diagram

VI.1 POSITION MODE

As can be determined from Figure 4, the Position loop has a characteristic equation of

$$s^2 + (42 + 42Kv)s + 36.3Kp.$$

Because the MC68705 processor being utilized has no hardware multiply/divide capability, it was desireable to implement algorithm gain factors as powers of 2 so that digital shift techniques could be utilized. Therefore, desiring a critically damped system, a set of appropriate gains would be Kv=0.25 and Kp=38. To incorporate digital shifts gains of Kv=0.25 and Kp=32 were selected. With data sampling at a 400 Hz rate and a control loop natural frequency of approximately 6 Hz, linear analysis should be applicable.

Four included figures document the expected and actual performance of the smart hand in Position mode. Figure 5 is the result of a simulation run using the analytically established gains, and shows slight overshoot. Figure 6 shows the results of a large position step test of the hand using the established gains. Slightly more overshoot is observed, however, this anomaly was traced to excessive mechanical deadband in the test unit which could not be readily remedied. In order to reduce the mechanical wear of the drive mechanism during testing, the gain Kp was reduced to 8. In addition, the rate feedback gain Kv was made position dependent, increasing to 0.5 when the actual position was within .100 inches of the commanded position. The performance with these changes is shown in Figure 7. Even though the position loop static accuracy was compromised by the gain reduction the increased smoothness of operation indicated overall increased benefits. Figure 8 indicates the large step position performance where an object is encountered prior to reaching the commanded position. As can be seen when a force is detected the claws stop trying to finish positioning and "back up" slightly to eliminate any detection of force and the hand awaits a subsequent valid command recognizing that it cannot complete the last one received.

VI.2 RATE MODE and RATE TO FORCE TRANSITION MODE

The Rate Mode configuration from Figure 4 yields a characteristic equation of

$$s^2 + 42s + 42Kva*Kv1.$$

Selecting Kva=8 and Kv1=1 as a proper gain set for the loop constants, provides an adequately damped system response as demonstrated by the simulator run output shown in Figure 9. System tests have validated these gain constants through demonstration of proper rate loop performance over the full dynamic range of operation.

Now to the more difficult section of the design requirements, the Force Mode control loop and the transitioning to it from the Rate mode. Without some predictor capability (i.e. proximity sensing) it is important that the bandwidth of the Force loop be maximized so that unplanned force transients imparted to the workpiece are minimized. From Figure 4 it can be determined that the characteristic equation of the Force mode configuration is

$$s^3 + 42.08s^2 + 2672.4s + 3752Kh*Kha.$$

Selecting Kh=1 and Kha=0.5 yields factors of

$$(s + 0.7)(s^2 + 41.3s + 2642).$$

These factors indicate a slightly underdamped control response. A simulator run output using the selected parameters is shown in Figure 10. The response is considerably less damped than the linearized system equations would indicate, however, this has been determined to be due to a realistic motor power limit included in the simulation. To compensate for the effect of power limiting the damping was increased by raising Kha to 1. Operational testing of the smart hand in the Rate mode with Auto Transitioning to Force indicated that the increased damping was adequate. Photos showing performance in these control regimes are included as Figures 11 thru 13. These tests were run by starting with the claws completely open and giving a full rate close command with obstacles of various compliances set to interfere with the closing. Figure 11 shows the transitioning region when a spring loaded compliance of approximately .001 in/lb was used as the target workpiece. Figure 12 shows the transitioning when a

solid aluminum bar (compliance less than .00001 in/lb) was utilized as the workpiece. The observable stepping in the force profile of relatively non-compliant loads is a phenomena of the software integrator and the incremental nature of the pulse width modulation resolution. Figure 13 is an expansion of the initial transient of Figure 12 demonstrating the transient energy transfer of the claw dynamics to the workpiece under near worst case conditions.

VI.3 HOLD MODE

Figures 14 and 15 show simulated performance of the control system to load disturbances when in the Hold Mode. The gains of the loop were set to Kg=1 and Kga=1 based upon the evaluation of the Force Mode response as previously described.

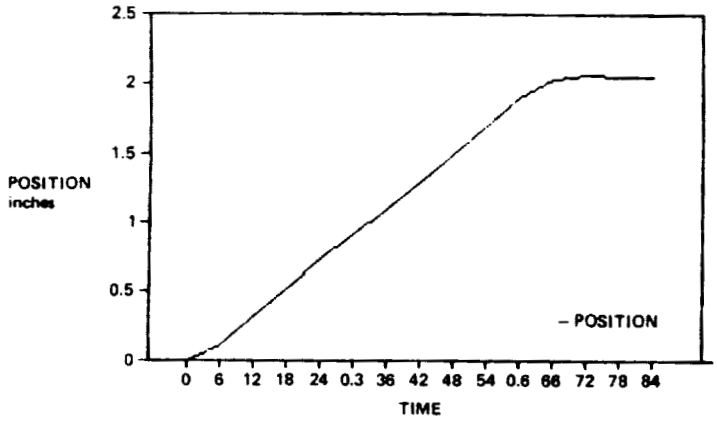


Figure 5. Smart Hand Position Mode

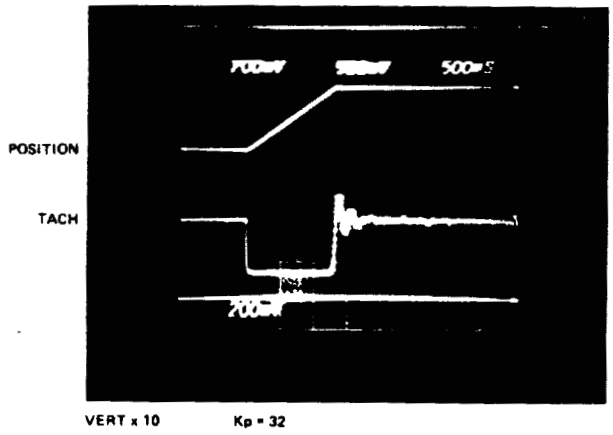


Figure 6. Position Servo Underdamped

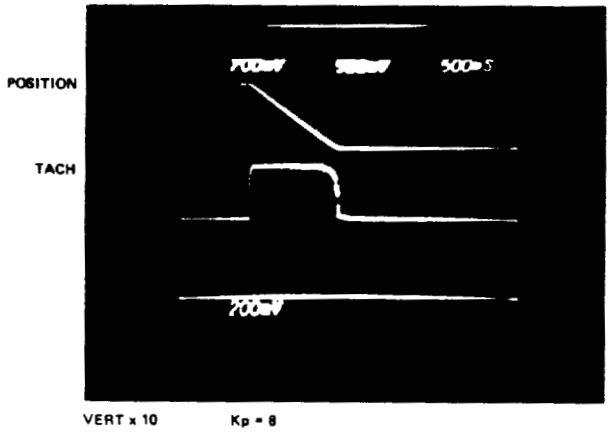


Figure 7. Position Servo Behavior

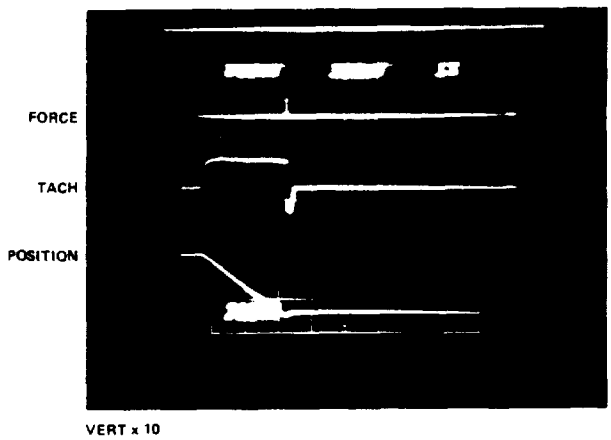


Figure 8. Position Mode With Interfering Force

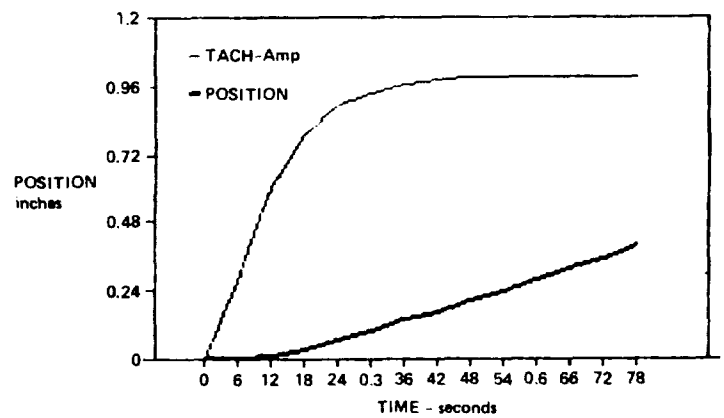


Figure 9. Smart Hand Rate Mode

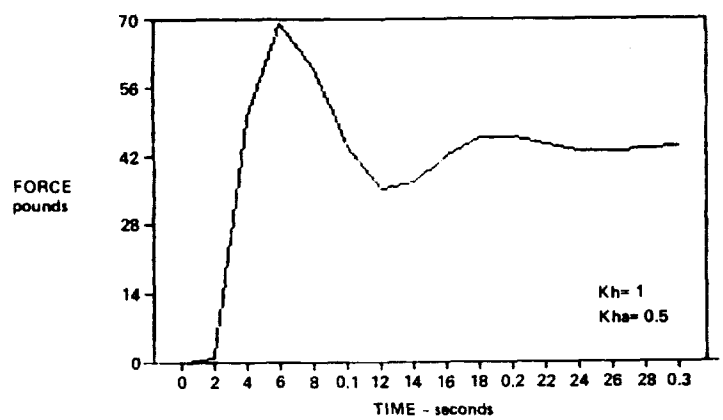


Figure 10. Smart Hand Force Without Hold

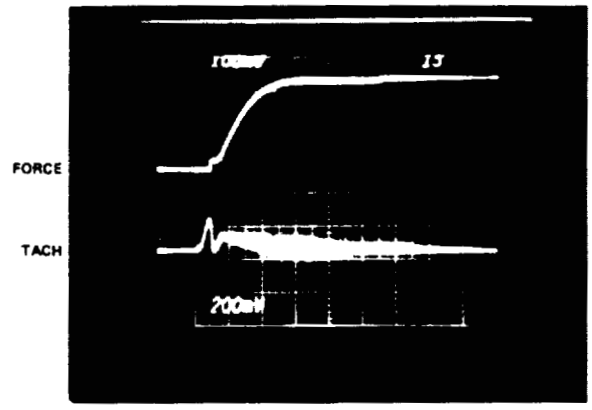


Figure 11. Rate to Force Transition - Full Rate

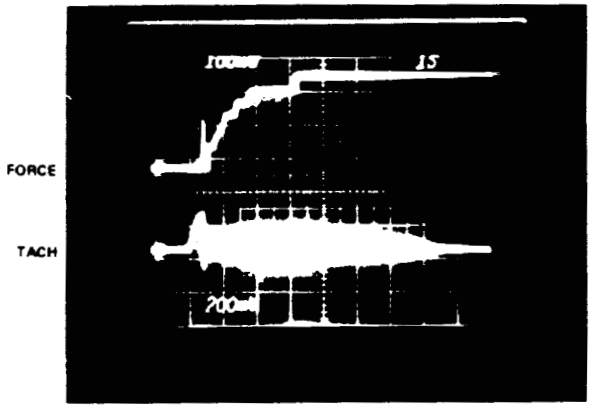


Figure 12. Rate to Force Transition - 1 sec/cm Scale

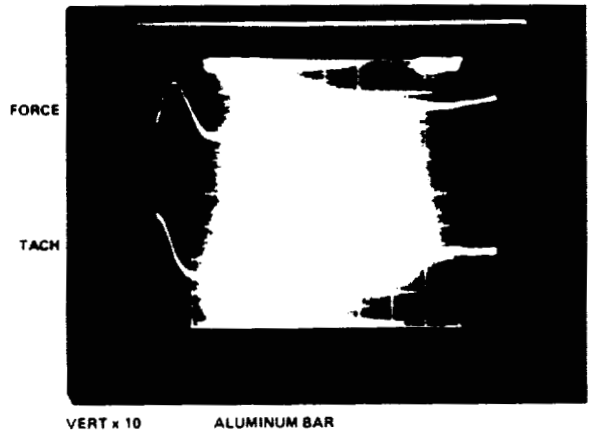


Figure 13. Rate to Force Transition - Full Rate Initial Transient Expanded 50 ms/cm Scale

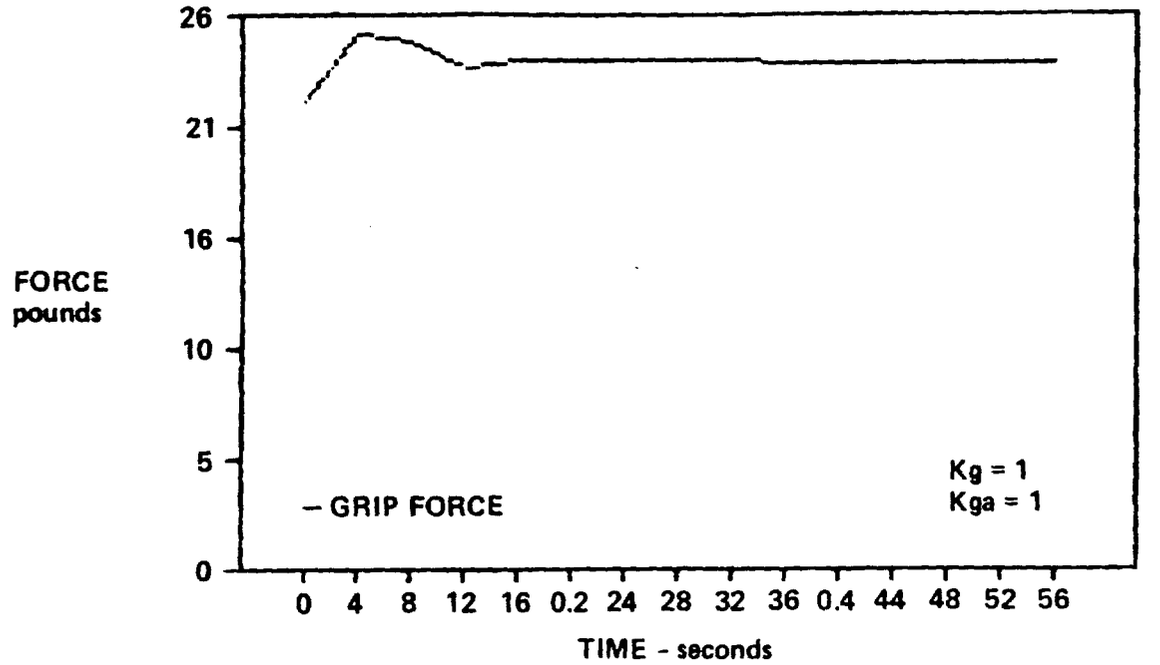


Figure 14. Smart Hand Force With Hold

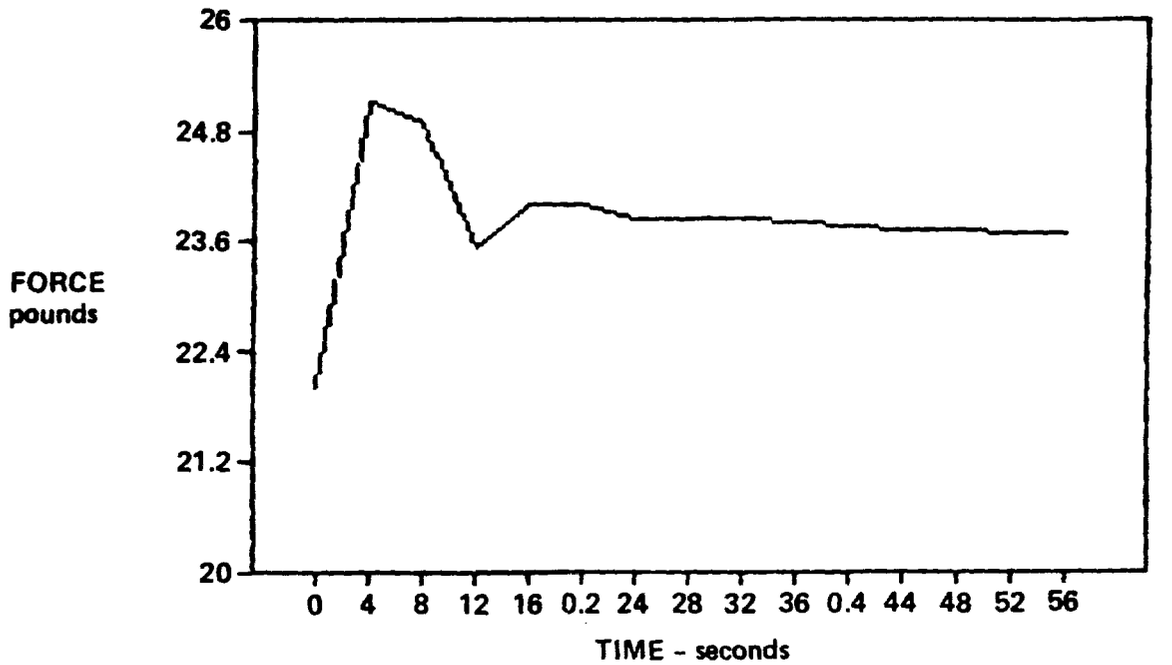


Figure 15. Smart Hand Force With Hold

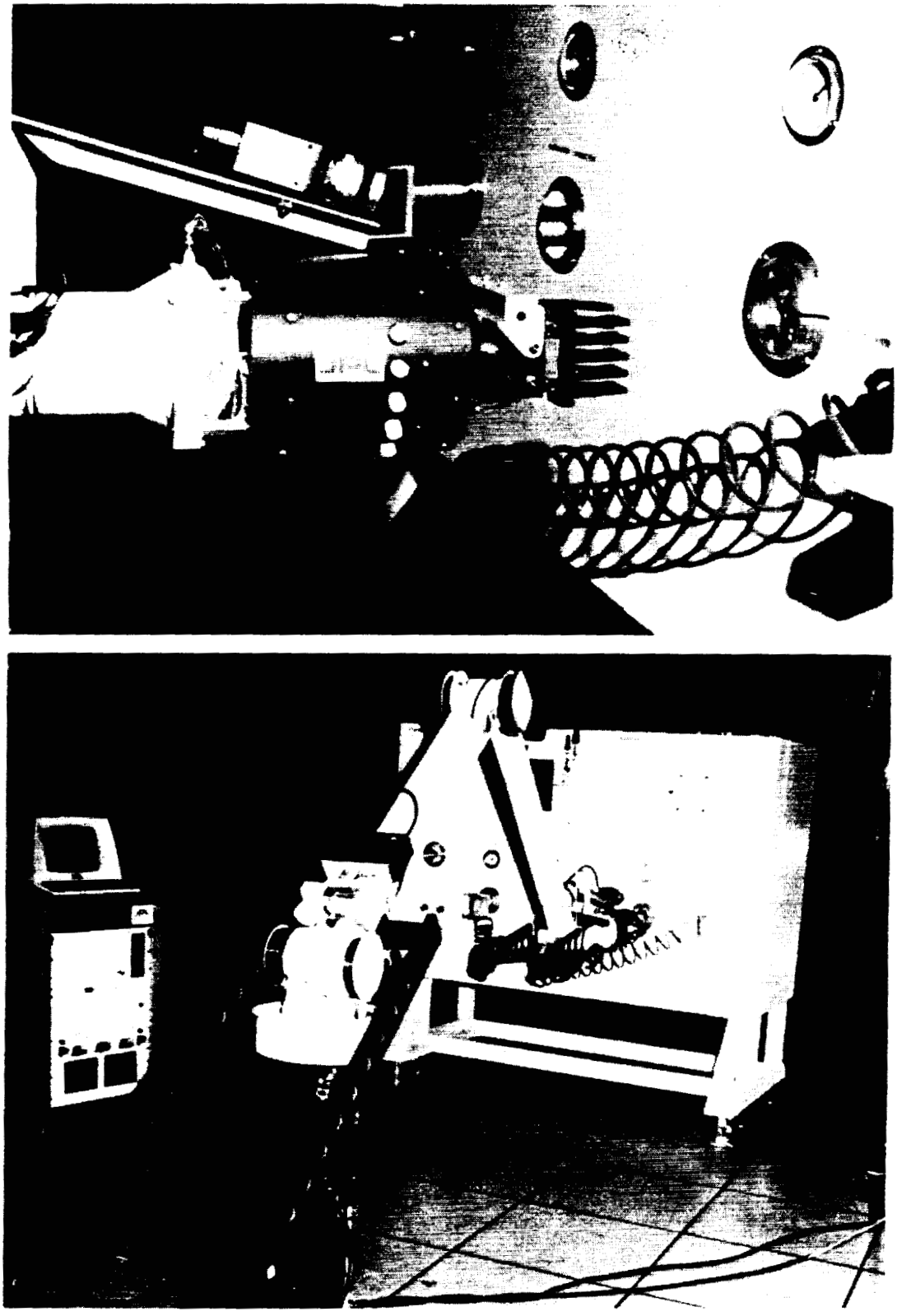


Figure 16. PFMA/JPL Smart End Effector Integrated With PFMA Arm at MSFC (Performing Connector Coupling Operation)

VII. CONCLUSION AND FUTURE PLANS

The smart hand has been mounted to the PFMA at MSFC (see Fig. 16) and tested with human operators performing simulated satellite servicing tasks. The tasks consisted of connecting and disconnecting a fluid coupling device, simulating module changeout, and attempting to exert constant forces and monents on the environment. Operators performed the tasks both with and without the force/torque display. Testing consisted of recording forces and torques from the wrist and jaw mounted sensors while operators performed tasks in manual control mode from a remotely located control station at MSFC. The tests and the results are described and evaluated in detail in Ref. 3.

In general, the experienced operators accomplished the tasks with lower levels of root-mean-square forces than intermediate or naive subjects. However, the test results have shown that improved display and manipulator control modes will be required to take full advantage of the end effector's sensing capabilities. The general conclusion is that sensors, displays, actuators and control modes for teleoperation cannot be designed or fully evaluated in isolation. For improved and optimized performance, the full teleoperation control loop, including the human operator, must be considered as it was pointed out in Ref. 3.

Future development plans include:

(i) Implementing the automatic execution of grasp force control

(ii) Implementing event driven displays

(iii) Designing a proximity ranging device integratable with the existing smart hand system.

In an event driven display, the simultaneous presence (or absence) of several force and torque component levels will be monitored and automatically indicated on the display with a distinct and easily perceivable symbol.

VIII. ACKNOWLEDGEMENT

This work was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract to the National Aeronautics and Space Administration.

IX. REFERENCES

1. A.K. Bejczy and B.M. Jau, Servicing with Smart End Effectors on OMV Manipulator, Proceedings of Satellite Servicing Workshop II, Johnson Space Center, Houston, TX, November 6-8,1985.

2. A.K. Bejczy, E.P. Kan and R.R. Killion, Integrated Multi-Sensory Control of Space Robot Hand, AIAA Guidance, Navigation and Control Conference, Snowmass, CO, August 19-21,1985,paper no.: AIAA-85-1882.

3. B. Hannaford, Task Level Testing of the JPL-OMV Smart End Effector, Proceedings of Space Telerobotics Workshop, JPL-CALTECH, January 20-22,1987

VISION SYSTEMS FOR RECOGNIZING KNOWN OBJECTS

William J. Wolfe*, Gregory Duane*, Mitchell Nathan**

*Martin Marietta Denver Aerospace
Advanced Automation Technology
Mail Stop 0427
P. O. Box 179
Denver, Colorado 80201

**Department of Psychology
Campus Box 345
University of Colorado
Boulder, Colorado 80309

INTRODUCTION

One of the most challenging problems in computer vision is that of representing and subsequently recognizing **known objects**. The core problem is to match image-derived features with existing object models. The **data-driven** aspects of the recognition process, which work at extracting primitives from the image, such as edges, boundaries, regions etc. , must be brought into registration with the **model-driven** processes which attempt to predict the appearance of a known object. (See figure 1).

These data-driven and model-driven processes typically meet in an ill defined middle ground, often called intermediate level vision, where many advanced computer vision concepts, such as intrinsic images and 2 1/2 dimensional sketch, have taken root. Advanced research has demonstrated that progress in these areas requires the use of many interacting models, from geometric models of the object to detailed models of the physics of the sensors, and the illumination and reflective properties of surfaces. Horn's work on **shape from shading** shows how such models can be used to derive shape parameters [1]. Further, work on **optical flow** and **structure from motion** represent attempts to incorporate models of motion as well [2].

These advanced techniques have grown over the years into areas of specific expertise within computer vision research. Our interest in this paper, however, centers around the more basic geometrical and topological **modeling** of objects and how such models can be used in a vision system, an area whose criticality was recognized at the onset of computer vision research, as demonstrated in the work by Roberts [3], but where progress has been amazingly slow.

THE REPRESENTATIONAL PROBLEM

Outdoor scenes present such a wide variety of form and motion that it is not hard to understand the **representational** problems that might arise, but it is a little surprising that even in highly structured environments the problem remains extremely difficult. Certainly, many problems can arise with terms like "**chair**" , since precise definitions are hard to come by, invoking quasi-philosophical issues. But even when we take a more precise example, such as a particular manufactured item, we discover that it can be described in many ways and at various levels, such as geometric resolutions, topological shape descriptors, functional, materials, etc. , all of which can have an impact on the recognition process. Even relatively simple objects, such as scissors or pliers, can be difficult to represent because of the moveable parts (open, closed, etc.). Also, many manufactured objects consist of wires, cables and other flexible structures and the particular condition of these structures at the time of viewing are impossible to predict. Thus, a complex object, such as a satellite with thermal blankets and solar panels, can defy representational attempts despite the fact that the object is in some sense well known.

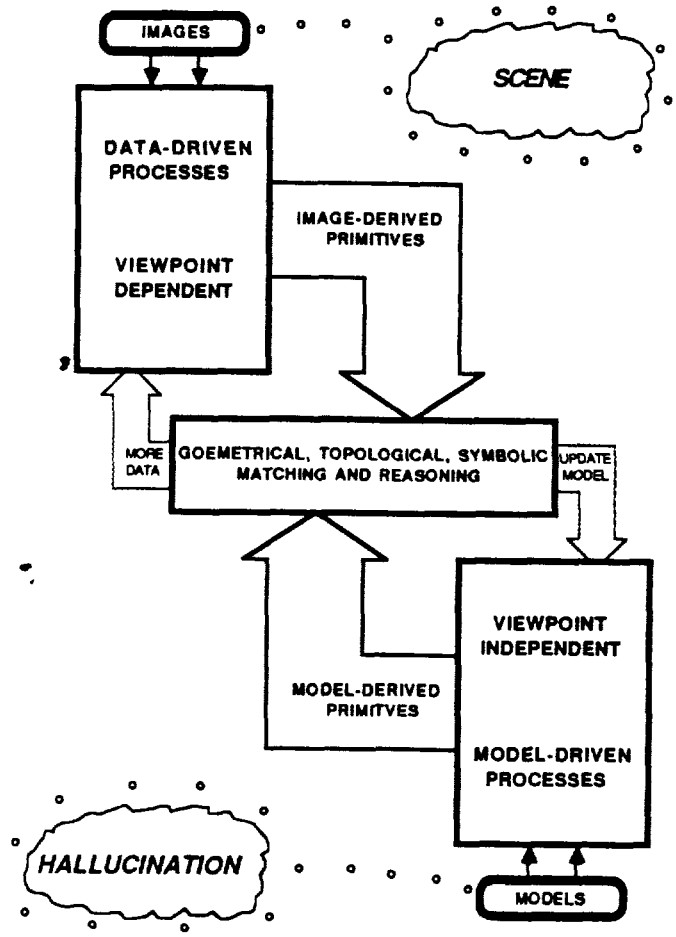


FIGURE 1 : DATA-DRIVEN PROCESSES MEET THE MODEL-DRIVEN PROCESSES.

OBJECT-CENTERED MODELS

There are many ways to describe, or model, an object. Here we have chosen to break object-centered modelling into a three level hierarchy : symbolic, topological and geometric. We refer to the geometric level as the "lowest" and the symbolic level as the "highest". This hierarchy provides a context within which we can describe and compare three different vision systems that were developed in our computer vision laboratory over the past few years. (See figure 2).

Geometric Level

At the **lowest** levels of a **hierarchy** of object representations are purely geometric descriptions consisting of precise mathematical parameters in object-centered coordinates that build the object surface from such primitives as vertices, curves and surface patches. It is important to note that completely within this "lowest" level is a separate hierarchy of spatial resolutions, progressing from coarse to fine surface geometry. The different levels of resolution can be very useful; for example, **collision avoidance** requires rough estimates of object shape whereas **manipulation** requires fine detail. Although various methods of representing surface geometry exist, from the brute force cataloguing of many vertices and connectives to the more elegant generalized cylinders ([4])and intrinsic geometric features ([5], [6]), no single, best approach has yet been identified.

MODELLING HIERARCHY	SAMPLE SYSTEMS
SYMBOLIC	RULE-BASED
TOPOLOGICAL	SPATIAL REASONING
GEOMETRICAL	LOCATING SYSTEM

FIGURE 2 : OBJECT MODELLING HIERARCHY AND SYSTEM EXAMPLES

It is somewhat surprising that straightforward modeling techniques have not proven sufficient for computer vision, especially in view of the long history of geometry. The central difficulty, however, is that object-centered **primitives** are different from the "primitives" that are typically derived from an image. For example, not only is there the problem of **image-derived** edge elements arising from purely spurious artifacts of the imaging process, but even when an edge element does correspond to a distinct geometric feature of the object it is not always true that this geometric feature has any meaning in the **object-centered** model, as is typically the case with occluding contours formed by curved surfaces. In fact, initial computer vision experiments used polyhedral objects mainly because there is a relatively clear relationship between image-derived edge elements and simple wire frame models. In general, however, it is very difficult to match image-derived primitives with model-defined primitives.

One of the reasons for this mismatch in primitives is the fact that one set of primitives is **viewpoint dependent** and the other **viewpoint independent**. Further, purely geometric, object-centered modeling does not use knowledge of illumination, surface reflectivity or sensor characteristics. Thus, a straightforward way to remove this weakness would be to incorporate such models, providing there is the potential to convert viewpoint independent models into predictions of the viewpoint dependent appearance of objects (using methods derived from computer graphics), and correlate them with the image-derived features. This is a powerful approach and represents much of the current research in computer vision; unfortunately, this is not generally practical.

This discussion has tacitly assumed normal **video** sensing and to some extent **color**, but in view of the problems described above one begins to consider alternative sensing methods, particularly, a **range imaging** sensor, such as a scanning laser range finder. Even though a range image suffers from some of the same viewpoint dependency problems as video, it has several advantages. Aside from being immune to shadows and complex shading effects, it provides the kind of direct **metric** information that can be more easily correlated with existing geometric models. Unfortunately, the physics of laser range sensing introduces technical problems of its own, such as processing time, dynamic range, specularity, etc. [7] , but the importance of sensing direct metric information cannot be underestimated.

Sample Geometric Approach : Locating System

This example is based on the problem of determining the location of a known object from a single perspective view. The **Locating System** assumes that low level image processing can extract the image plane location of the vertices of a rectangular shape that is known to be on the object. There are many variations on this theme, using three or more points, and here we have chosen a rectangle for simplicity. Using inverse perspective operations the image plane coordinates can be converted into the three dimensional position of the object relative to the sensor in all six degrees of freedom, [8].

Of course, this method does not recognize the object but it provides a means of getting a quick estimate of the viewpoint from a minimum of object knowledge. Object recognition techniques have a greater chance of working when the viewpoint is approximately known. Estimates of the viewpoint can also be used to derive an improved viewpoint before any object recognition is attempted.

The alternating use of knowledge of viewpoint and then object geometry, beginning with rough estimates of each, is a powerful approach that helps refine the required processing at each stage in a hierarchical manner.

The Topological Level

Even if reasonably effective low level primitives could be derived from video and/or range images, there would still be a **combinatorial** problem. Matching hundreds of low level primitives in an exhaustive manner is prohibitive. Looking for a way out of this difficulty leads to a process of **abstraction** similar to that involved in solving complex planning problems, [9]. The strategy is to postpone the matching process until higher level shape descriptions are derived. That is, certain global features such as "**flat**" or "**curved**" are easier to work with. In fact, these global features tend to have some degree of **invariance** with respect to the viewing conditions. For example, "**has a concave edge**" is a property that could be derived from a variety of viewpoints and helps prune major branches in a tree of possible object matches.

Thus, a slightly higher level of modeling might describe an object as being composed of several surface parts, each classified by a few parameters such as average gaussian curvature and total area, connected together at edges that are also simply classified (straight, curved, convex, concave etc). This level retains some of the detail of the lower, geometric level but loosens it up a bit in favor of more global shape descriptors. For example, **concave** and **convex** edges would be distinguished without the use of actual angular measures. (Of course, if registration is maintained with the lower level geometric description, the actual angular measures could be recovered, if needed). Further, the individual surface parts can be treated as nodes in a **Region Adjacency Graph** which expresses connectivity with a minimum of shape parameters. (Because of viewpoint dependency, any region adjacency graph derived from the image will correspond to a subgraph of the object-centered adjacency graph, thus creating a **subgraph isomorphism** problem that can be combinatorial prohibitive but is still better than matching the lower level primitives).

Within this level, precise geometric information is gradually dropped in favor of **global**, almost symbolic, descriptions. For example, shape descriptors such as "**hole**", "**sharp**", "**round**" and "**smooth**" might be introduced without direct reference to the precise underlying geometry. Of course, such representations as "**cylindrical**", "**spherical**", "**straight**" and "**has parallel sides**" would be exploited for both geometric and symbolic purposes.

Proceding further along the lines of dropping precise geometric parameters leads to topological descriptions of objects in terms of the number of **holes** they have. At this level, precise coordinates are insignificant; instead, objects are considered as members of **homotopy** equivalence classes defined by **homeomorphisms** that mathematically express the continuous deformation of one object into another, such as donut into coffee cup. For example, whereas a sphere and a cube would be equivalent as would a washer and a nut, a sphere would be distinguished from a washer by counting the number of holes and not by any detailed comparison of coordinates.

Sample Topological Approach : Spatial Reasoning System

The **Spatial Reasoning System** described in [10] is a good example of how topological features can be used to recognize known objects. Using the higher level, more abstract, topological and spatial properties provides this system with many interesting side effects, such as the ability to hypothesize the existence of occluded features as well as features that may have been missed by the low level image processing. Further, the system can propose new viewpoints that would disambiguate a conflict between objects that look alike from a particular viewpoint.

The important steps in this spatial reasoning process consist of converting image-derived features, such as surface, edge and point topological relationships, into logical assertions that are further examined to remove viewpoint dependent artifacts, such as occluding edges resulting from curved surfaces. (The system assumes that the low level image processing has done a reasonable, although not perfect, job of measuring three dimensional surface characteristics, such as flat or cylindrical. Theoretically, this information could be easily derived from high resolution range data).

Thus, the data-derived viewpoint dependent assertions are transformed into viewpoint independent assertions and are subsequently expressed in predicate logical terms as a mechanism for implementing various consistency checks. The ultimate goal is to achieve a subgraph isomorphism between the data-derived object description and the stored object model. To minimize the combinatorics, a hierarchical description of critical object features is incorporated so that certain object hypotheses can be rejected at a high level.

The Symbolic Level

The two levels described above were based primarily on descriptions of shape and spatial relationships. The symbolic level brings in "everyday" names and descriptions, emphasizing the functional aspect of an object, such as how and what it is used for, its relationship to other objects, etc.

It is a classic result of research in artificial intelligence that the symbolic level is easy to discuss but astonishingly difficult to imbed into a machine. The basic difficulty is that these simple sounding symbols float atop a sea of detail, with levels of complexity that become apparent only after many attempts at implementation. It seems that human intelligence hinges on a natural ability to simplify complex matters, such as the abundance of sensory information, to the point where the true complexity of the process is well hidden and subconscious and must be laboriously dug out by researchers. Thus, the symbolic level allows us to blithely make assertions such as " chairs are usually near tables" but leaves us struggling to make these relationships explicit.

More demonstrable results in this area have been in expert systems, hence our example in the symbolic category is rule-based.

Sample Symbolic Approach : Rule-based

This particular rule-based approach, as described in [11], attempts to bypass the problems of intermediate level vision by incorporating explicit scene interpretation rules, both domain independent and domain dependent. The method was originally developed for outdoor applications such as guiding an autonomous vehicle down a road but the basic principle applies to object recognition as well.

The basic idea is to achieve a high level, symbolic description of the scene, in terms of labeled regions, via a sequence of split and merge operations in conjunction with the application of rules that encode knowledge of the behavior of the low level algortihms together with knowledge of the domain. The number of rules can become prohibitive, but the number can be kept reasonable by sufficiently restricting the scene domain.

For example, merging criteria such as " average gradient between regions " are augmented with interpretation rules such as :

RULE 22 IF	(REGION IS ADJACENT TO THE TOP IMAGE BORDER)	AND
	(REGION IS BRIGHT)	AND
	(REGION IS LARGE)	AND
	(BOUNDARY OF REGION WITH TOP BORDER IS LONG)	AND
THEN	(REGION IS "SKY")	

The chief advantage to this approach is that the critical interpretation criteria is made explicit, allowing an easy way to add new knowledge. This method blends the low level operations that extract of primitives (regions) with high level domain knowledge.

CONCLUSION

The future of computer vision research will see the refinement of the various approaches described above. Particularly, however, there will be great progress in those scenarios dealing with known objects. Incremental methods that merge new information with stored models, (see [12] and [13]), will have a critical impact on automation. For example, a teleoperated robotic system that relies on a human for initial alignment could then resort to an automatic vision system that exploits specific knowledge of the objects and viewpoint. Such a system, however, would have to rely on several levels of object modelling, from the geometric to the symbolic.

ACKNOWLEGEDMENTS

We would like to thank John Bradstreet, Ben Bloom and Bob Terrell for several helpful comments during the preparation of this paper.

REFERENCES

[1] Horn, B., Robot Vision, MIT Press, 1986.

[2] Ulman, S., The Interpretation of Visual Motion, MIT Press, 1979.

[3] Roberts, L., "Machine Perception of Three Dimensional Solids", Optical and Electro-optical Information Processing, MIT Press, 1965.

[4] Binford, T. O., "Visual Perception by Computer", IEEE Conference on Systems and Control, Miami, Dec, 1971.

[5] Barrow, H. G. and J. M. Tenenbaum, "Computational Vision." Proceedings of the IEEE, V 69, N 5, (May 1981), 572-595.

[6] Besel, P. J., Jain, R. C., "Invariant Surface Characteristics For 3D Object Recognition in Range Images", Computer Vision, Graphics and Image Processing, 33, 33-80,1986.

[7] Nitzan, D., Brain, A. E., Duda, R. O., " The Measurement And Use of Registered Reflectance And Range Data In Scene Analysis", Proceeding of the IEEE 65:2, 1977.

[8] Wolfe, W. J., Arensdorf, G., White, G., Pinson, L. J., "Robotic Locating System", SPIE, Vol 726, Intelligent Robots And Computer Vision, Fifth In A Series, 1986.

[9] Sacerdoti, E. D., "Planning In A Hierarchy Of Abstraction Spaces", Artificial Intelligence, 5(2) (1974) 115-135.

[10] Magee, M., Nathan, M., "A Viewpoint Independent Modelling Approach To Object Recognition", Journal Of Robotics And Automation, Feb, 1987.

[11] Duane, G. S., "Knowledge-based Segmentation Of Road Scenes", SPIE, Vol 697, Applications Of Digital Image Processing IX, 1986.

[12] Herman, M., Kanade, T., "Incremental Reconstruction of 3D Scenes from Multiple, Complex Images", Artificial Intelligence, 30 (1986) 289 - 341.

[13] Herman, M., "Merging 3-D Symbolic Descriptions Obtained From Multiple Views Of A Scene", SPIE, Vol 726, Intelligent Robots And Computer Vision, Fifth In A Series, 1986.

An Optimal Resolved Rate Law for Kinematically Redundant Manipulators

B. J. Bourgeois

McDonnell Douglas Astronautics Co.
- Engineering Services

May 1987

An Optimal Resolved Rate Law for Kinematically Redundant Manipulators

B. J. Bourgeois
McDonnell Douglas Astronautics Co. - Engineering Services

ABSTRACT - The resolved rate law for a manipulator provides the instantaneous joint rates required to satisfy a given instantaneous hand motion. When the number of degrees of freedom in the task space is the same as the number of degrees of freedom in the joint space the Jacobian matrix is square and the resolved rate law is easily determined for non-singular configurations. When the joint space has more degrees of freedom than the task space the manipulator is kinematically redundant and the kinematic rate equations are under-determined. In this paper an objective function is optimized with respect to the n kinematically redundant rate equations to provide an optimal resolved rate law that can be tuned to control the joint motion in a variety of ways. This law is used in an iterative algorithm to find joint angle solutions to the inverse nonlinear kinematic equations. The behaviour of the optimal resolved rate law is demonstrated and investigated in a 4 degree of freedom kinematically redundant planar arm model. A weighting matrix is used in the resolved rate law to avoid reach limits during the trajectory to a desired hand state. The treatment is applicable to manipulators with any number of revolute joints.

1.0 INTRODUCTION

The resolved rate law for a manipulator converts the instantaneous hand rates into instantaneous joint rates [1]. This allows the joints to be simultaneously commanded to move the hand with a desired instantaneous translational and rotational velocity. The space that the hand moves in is called the task space [2], which is usually composed of 6 degrees of freedom. The space mapped out by the joint angles is called the joint space. The mathematical relationship between the task space and the joint space defines the resolved rate law for the manipulator.

The kinematic equations express the hand state in terms of the manipulator joint angles and are usually very nonlinear. It is usually straight-forward, but tedious to write down the kinematic equations that express the hand state in terms of the joint angles. It is easy to differentiate the kinematic equations to arrive at an expression for the hand translational and rotational rates in terms of the joint rates. The matrix that results is sometimes called the Jacobian matrix for the manipulator. The kinematic equations for most manipulators are very nonlinear and generally cannot be inverted to solve for the joint angles in terms of the hand parameters [2].

The Shuttle Remote Manipulator System (SRMS) has six joints and the end-effector operates in a six degree of freedom space (three spacial and three angular). This is a convenient design because the number of degrees of freedom in the joint space and in the task space are the same; the Jacobian matrix is square and can be easily inverted, except in specific configurations where the Jacobian matrix is singular. When these singularities are avoided, the inverted Jacobian is a valid resolved rate law for the SRMS because it transforms a desired end-

effector translational and rotational velocity into joint rate commands. The joint rate commands can be issued periodically to the six servo motors to accomplish an end-effector motion as desired.

The simple resolved rate law described above is used in the SRMS flight software to drive the manual and the automatic modes. For these modes there is a requirement to move the hand coordinate system (or some coordinate system rigidly associated with the hand system) from one state to another with translation along a relatively straight path and rotation about a constant vector. Much effort has been spent finding such paths that are free from encounters with joint reach limits.

When a manipulator has more joints than the number of degrees of freedom in the task space it is said to be kinematically redundant [3]. The Jacobian matrix for a kinematically redundant manipulator is not square and cannot be directly inverted to arrive at an easy resolved rate law. There are more joint variables to solve for than there are kinematic equations. There is not enough information to solve for the joint rates needed to move the hand. In general, there may be an infinite number ways to move the joints in unison to provide the desired hand motion for the kinematically redundant manipulator [4],[5].

It is essential to arrive at some sort of a resolved rate law in order to control or simulate a manipulator. Several methods have been introduced to arrive at adequate resolved rate laws for the kinematically redundant manipulator. One approach is to add specific constraints on the manipulator so that the kinematic equations can be solved. A more general approach is to minimize or maximize an objective function subject to the kinematic constraint equations. These methods have been investigated in several papers to study iterative solutions to the kinematically redundant constraint equations [1],[3-7].

In this paper an optimal control law with a weighting matrix is derived using the Moore-Penrose pseudo-inverse for general manipulators with various dimensions in task space and joint space. The behavior of the control law is demonstrated and investigated using a kinematically redundant planar arm simulation. Several algorithms are introduced and evaluated for dynamically adjusting the weighting matrix during the trajectory for the purpose of avoiding joint reach limits.

2. PROBLEM FORMULATION

The resolved rate law for a manipulator is derived from the kinematic equations. For a manipulator with n joints and a hand operating in a task space of m dimensions, the m kinematic equations are of the form:

$$x = \{x_k\} = \{f_k(\theta_1, \theta_2, \ldots \theta_n)\} \quad k=1,m \tag{2.1}$$

where x is the vector containing the task space coordinates and θ is the vector of joint angles. If each joint is moved by a small amount, $\Delta\theta$, then the movement of the hand in the task coordinates, Δx, is found in the differential of the kinematic equations:

$$\Delta x = [J]\, \Delta\theta \tag{2.2}$$

where J is the Jacobian matrix [8], composed of the partial derivatives of the functions f with respect to each of the joint angles. Similarly, the kinematic rate equations are found by differentiating the kinematic equations with respect to time.

$$v = dx/dt = [J]\, w \tag{2.3}$$

where v is the hand velocity vector expressed in the task coordinates and w is the vector of joint rates. The resolved rate law is found by solving the kinematic rate equations (2.3) for the joint rates (w) in terms of the hand velocity (v). In the case where the task space and the joint space have the same number of dimensions (m=n) the Jacobian matrix is square and the resolved rate law is easily found.

$$w = [J]^{-1} v \tag{2.4}$$

When the determinant of the Jacobian is zero, the manipulator is in a physical singularity and cannot supply motion in all of the dimensions of the task space. In mathematical terms the joint space does not span the task space when the arm is in a singularity.

When the manipulator has fewer joints than dimensions in the task space (n<m) the system is overdetermined and the resolved rate law may be found by using the pseudo-Jacobian method. For a 5 jointed manipulator acting in a task space of 6 dimensions, there will be 6 kinematic equations but only 5 joint variables.

$$\underset{6x1}{v} = \underset{6x5}{J}\ \underset{5x1}{w} \tag{2.5}$$

The brackets around the Jacobian have been dropped to simplify the notation. The resolved rate law for such a manipulator can be derived by pre-multiplying by the Jacobian as follows:

$$J^T v = J^T J\, w \tag{2.6}$$

$$w = (J^T J)^{-1} J^T v \tag{2.7}$$

$$\underset{5x1}{w} = [\ (\underset{5x6}{J^T}\ \underset{6x5}{J})^{-1}\ \underset{5x6}{J^T}\ ]\ \underset{6x1}{v} \tag{2.8}$$

The expression within the brackets is the Moore-Penrose pseudo-inverse of the full rank rectangular (6x5) Jacobian for the overdetermined system of equations 2.5 [9].

For a kinematically redundant manipulator (n>m) the Jacobian matrix is not square and the system of equations is underdetermined. For a 7 jointed manipulator operating in a task space of 6 dimensions there are 6 kinematic equations and 7 joint variables. The Jacobian matrix is a 6 by 7 matrix. There are several approaches which have been used to solve this underdetermined set of equations. The approach taken here is to

introduce an objective function to be minimized subject to the constraint equations:

$$\underset{6x1}{v} = \underset{6x7}{J}\ \underset{7x1}{w} \tag{2.9}$$

An obvious objective function to consider is:

$$Z = \frac{1}{2}\left(w_1^2 + w_2^2 + w_3^2 \ldots + w_n^2 \right) \tag{2.10}$$

When this function is minimized, the solution results in the least amount of instantaneous motion in all joints. Using the method of Lagrangian multipliers (L) the following n+m equations result [11].

$$w = J^T L \tag{2.11}$$

$$J w = v \tag{2.12}$$

Solve equation 2.11 for the Lagrangian multipliers and use equation 2.12 to substitute v for Jw.

$$L = \left(J J^T \right)^{-1} J w = \left(J J^T \right)^{-1} v \tag{2.13}$$

Substituting this expression for the Lagrangian multipliers gives the resolved rate law.

$$w = \left[J^T \left(J J^T \right)^{-1} \right] v \tag{2.14}$$

It is interesting to compare this result with that of the pseudo-inverse result of equation 2.7. The expression in brackets in equation 2.14 is the Moore-Penrose pseudo-inverse of the Jacobian for the underdetermined system of equations (2.9) [4],[9].

A more interesting objective function [1] has the form:

$$Z = \frac{1}{2}\left(a_{11} w_1^2 + a_{22} w_2^2 + \ldots a_{nn} w_n^2 \right) \tag{2.15}$$

or in matrix notation:

$$Z = \frac{1}{2} w^T A w \tag{2.16}$$

The constraint on the weighting matrix A is such that the function Z is non-negative for all values of w [11]. This condition will be satisfied by considering only diagonal weighting matrices with positive values. The resolved rate law that results from optimizing this objective func-

tion can be found by following the procedure used in equations 2.11 through 2.14.

$$w = A^{-1} J^{T} (J A^{-1} J^{T})^{-1} v \qquad (2.17)$$

This result also appears in [1]. For the case of the seven jointed manipulator described above this resolved rate law is

$$\underset{7x1}{w} = \underset{7x7}{A^{-1}} \underset{7x6}{J^{T}} (\underset{6x7}{J} \underset{7x7}{A^{-1}} \underset{7x6}{J^{T}})^{-1} \underset{6x1}{v} \qquad (2.18)$$

where the dimensions of each matrix and vector have been indicated for clarity. The weighting matrix A is of special interest. It can be used to control the motion of the joints by dynamically changing the values of the diagonal components during the trajectory.

3.0 IMPLEMENTATION

The control law expressed by equation 2.17 was implemented into a kinematically redundant planar manipulator model. The planar arm model was developed for the purpose of studying the motion of the redundant manipulator. This model is easy to work with because the task space is confined to a flat plane and is adequate for studying the behavior of the control law.

3.1 The Kinematically Redundant Planar Manipulator (KRPM)

The KRPM model has a task space composed of 3 degrees of freedom as shown in figure 1. The task space is composed of X, Z and P (pitch) directions for the hand, and all joints are pitch joints. The number of pitch joints (n) can be specified from 3 to 10. For this study, n was set to 4 so that there is only 1 redundant joint except where noted otherwise. This was done to form a direct analogy with the 7 jointed manipulator operating in a task space of 6 degrees of freedom. The kinematic equations for the 4 jointed planar arm are shown below. The usual abbreviated notation is used to simplify the algebra.

$c1 = \cos(\theta_1)$
$c12 = \cos(\theta_1 + \theta_2)$
$c123 = \cos(\theta_1 + \theta_2 + \theta_3)$
$c1234 = \cos(\theta_1 + \theta_2 + \theta_3 + \theta_4)$
similarly for s1, s12, s123, and s1234 using sines
L_i = Length of boom i (3.1)

The kinematic equations are now easily written.

$$X = L_1c1 + L_2c12 + L_3c123 + L_4c1234 \tag{3.2}$$

$$Z = -L_1s1 - L_2s12 - L_3s123 - L_4s1234 \tag{3.3}$$

$$P = \theta_1 + \theta_2 + \theta_3 + \theta_4 \tag{3.4}$$

The Jacobian for this arm can be found analytically by differentiating the above equations.

$$\begin{vmatrix} -L1s1 - L2s12 & -L2s12 - L3s123 & -L3s123 & -L4s1234 \\ -L3s123 - L4s1234 & -L4s1234 & -L4s1234 & \\ -L1c1 - L2c12 & -L2c12 - L3c123 & -L3c123 & -L4c1234 \\ -L3c123 - L4c1234 & -L4c1234 & -L4c1234 & \\ 1 & 1 & 1 & 1 \end{vmatrix} \tag{3.5}$$

The Jacobian for the planar arm can be greatly simplified by considering the special case where all of the boom lengths are set to unity. The following abbreviations are also convenient.

$$\begin{aligned} c14 &= c1 + c12 + c123 + c1234 \\ c24 &= c12 + c123 + c1234 \\ c34 &= c123 + c1234 \\ &\text{similarly for s14, s24, and s34} \end{aligned} \tag{3.6}$$

The Jacobian for the special case can now be conveniently expressed as follows.

$$J = \begin{vmatrix} -s14 & -s24 & -s34 & -s1234 \\ -c14 & -c24 & -c34 & -c1234 \\ 1 & 1 & 1 & 1 \end{vmatrix} \tag{3.7}$$

The KRPM model was simulated in FORTRAN on an HP9000 desktop 32 bit super micro computer. The model simulates the kinematics of any planar arm with 3 task dimensions (m=3) and any number of pitch joints (n) and booms of independent lengths. The Jacobian is computed numerically using a recursive vector form [1] to allow the simulation of various arms types. The attributes of the manipulator are described in a data file. Various manipulators may be represented by changing the data file.

3.2 Implementation of the Control Law

The optimal RRL (equation 2.17) was implemented into the KRPM arm model by first adapting the dimensions to those of the planar arm. The task space contains 3 dimensions, and the number of joints (n) may be 4 or greater. The RRL for the 4 jointed KRPM is defined as follows with dimensions shown.

$$[RRL]_{4x4} = A_{4x4}^{-1} J_{4x3}^{T} \left(J_{3x4} A_{4x4}^{-1} J_{4x3}^{T} \right)^{-1} \quad (3.8)$$

The simulation iterates through the RRL to drive the hand to a desired state. The flow of the calculation is as follows. The arm is positioned at a valid set of joint angles ($\theta_{initial}$) and through the forward kinematics the resulting hand state ($x_{initial}$) is computed. Then an input is made to indicate the desired final hand state for the arm (x_{final}). The difference between the two hand states (Δx) is found.

$$\Delta x = x_{final} - x_{initial} \quad (3.9)$$

The number of steps to take during the trajectory (s) can be selected. The vector Δx is divided into s steps. The RRL is computed using equation 3.8 and then the desired joint angle step $\Delta\theta$ is computed.

$$\Delta\theta = [RRL]\, \Delta x/s \quad (3.10)$$

The joint angles are then updated by adding the changes in joint angles. This procedure is repeated for steps 2 through s, maintaining the same step length in distance (X and Z) and in rotation but with an adjustment in the vector direction ($|\Delta x|$). After the last step is taken (step number s), a Newton-Raphson (NR) iteration is automatically invoked to trim up the final hand state to within a tolerance of the desired hand state. This method does not consider the joint rates of the manipulator. This simulation progresses by taking steps.

3.3 Step Size

A study was performed to determine the step size needed to provide hand motion along a straight line. Good results are measured by inspecting the path that the end-effector describes. The ideal trajectory should be a straight line. Several trajectories were tested while varying the number of steps between 1 and 80.

In figure 2 the arm is commanded from the joint angle state at A to the hand state at B. When a Newton-Raphson (NR) iteration is used (s=1) the first step actually goes the wrong way, to point 1 in figure 2a. When the NR iteration is completed, the arm ends up at the joint angle state shown at B. In figure 2b a two step iteration (s=2) is shown. The first step is a NR half-step and the second step is the remaining half step with an adjustment in direction. A full step NR iteration occurs between 2 and B resulting in a different final joint state than in 2a. A ten step iteration (2e) results in a reasonably straight hand trajectory, but eighty steps are required for very good results (2f).

In figure 3a the arm was commanded from the joint state of (90,0, -90,0) at A to the hand state of (3,0,0) at B with a small step size arriving at the joint configuration at B. This trajectory was then reversed by commanding from the joint angles at point B to the original hand state at A (2,-2,0) for various step sizes. Notice that for large

step sizes (figures 3b and 3c), the arm does not return to the original joint configuration of (90,0,-90,0) in figure 3a. As the step size decreases (a larger number of steps), the reverse trajectory converges upon the perfect joint state of (90,0,-90,0). This is expected because the trajectory that minimizes the motion of the joints in one direction should also minimize the motion in the reverse direction also. This procedure is considered a validation of the implementation of the RRL. These results illustrate the importance of taking small step sizes while seeking a practical joint angle solution for a redundant arm using the iterative inverse method.

3.4 Iterative Inverse Solutions

Several investigators have used modified NR algorithms to find inverse kinematic solutions for manipulators [3],[5],[6],[10]. All of these approaches are aimed at finding a quick joint angle solution with large step sizes in joint space, causing the hand of the manipulator to take an unpredictable and unrealistic path. This poses two problems when dealing with a physical manipulator. When the robot is actually commanded from the initial joint angles to the hand state through a resolved rate law, a different set of joint angles is likely to result than the one found by the iterative inverse. This effect is best illustrated in figure 2 where the joint angles at the final configuration in 2a are very different from the joint angles at the end of figure 2e. The joint angle solution is not very useful if the manipulator cannot be commanded to it. Secondly, when large steps in joint space are made there is more chance of violating the joint reach limits.

If the iterative inverse for the redundant arm constrains the hand to follow a straight path, rotate about a constant vector and checks for joint reach limits during the iteration then a solution arrived at will be a feasible one. This requires having a knowledge of the RRL that will drive the manipulator and taking small steps in the iteration.

4.0 BEHAVIOUR OF THE OPTIMAL RESOLVED RATE LAW

The behaviour of the optimal resolved rate law in the KRPM model is demonstrated in this section. Several trajectories were run to illustrate the ability of the control law to handle the redundancy of the kinematics and to study the effects of the weighting matrix on the joint motion.

The ability of the control law to handle the redundancy of the arm was demonstrated by driving the arm to the same hand state from various starting configurations. It also serves as an inverse solution to the kinematics, by providing several possible joint sets that satisfy the requested hand state. In figure 4 the end-effector was commanded to the state X=3, Z=0, and Pitch= 0 (3,0,0) from six different initial joint configurations. In each case the end-effector ends up at the final state of (3,0,0), but with different final joint angles. This simple test demonstrates the ability of the control law to drive the KRPM to different final joint states for a given end-effector state. The final joint angles are dependent on the initial joint angles.

The above maneuvers were performed with the weighting matrix A in

equation 3.8 equal to identity. This is the equivalent of having no weighting matrix (equation 2.14). The effect of the weighting matrix on the motion was demonstrated by running the same trajectory with various values of one of the components of the A matrix and observing the effects on the motion of the corresponding joint.

In figure 5 the arm was commanded to the end-effector state of (2,0,0) at B from the joint angle state (90,-90,0,0) at A. When the weighting matrix is not used (A is identity), the final value of the second joint is undesireable (fig. 5a). When a value of 2.0 is used for A(2,2) and 1.0 for all other diagonal components of A, the final position of joint 2 is noticably better (fig. 5b). Joint 2 moved less from start to finish than in figure 5a. The joint moves progressively less from start to finish as A(2,2) is increased. The remaining pitch joints have moved more to make up for the loss of mobility in joint 2, thus resulting in a more desirable overall final arm configuration.

In figure 5a the final condition of the arm is not disireable because joint 2 could be very near a reach limit, thus restricting any future movement after arrival at the desired end-effector state. In figure 5e, the final situation is much more desireable, because joint 2 has more freedom to move around in the neighborhood of the final end-effector state.

In this section it has been demonstrated that the weighting matrix can be used to discourage the motion of particular joints. It seems reasonable to use this information to maintain the joints away from their respective joint reach limits. This is a very desireable goal in robotic control, but is limited to redundant manipulators.

5.0 REACH AVOIDANCE ALGORITHMS

The behaviour of the pseudo-inverse of equation 2.14 has been reported to be peculiar in some cases [4]. The peculiarity has been associated with joint reach limit violations during certain tasks such as a closed path or cyclic motion. If reach avoidance logic is incorporated into the RRL, these problems may be resolved. One method of incorporating reach avoidance into the RRL is to include the upper and lower joint limits as a constraint in the optimization algorithm [6], which may become a complicated treatment. A simpler approach is taken here which makes use of the weighting matrix A in the RRL of equation 3.8.

In the previous section the effect of the weighting matrix on the arm motion was illustrated. It was shown that the redundant arm can be controlled to arrive at different final joint angles, some more desireable than others, and yet satisfy the same hand state. With these two findings, it is evident that the arm can be driven to arrive at various final joint angles as desired by controlling the weighting matrix during the trajectory.

The components of the weighting matrix (diagonal) must be computed from pass to pass according to some driving requirements. Examples of driving requirements are obstacle avoidance, joint reach limit avoidance, or some mechanical or electrical criteria. For this study, the goal is reach limit avoidance.

Three algorithms were implemented for evaluation. The first algo-

rithm is the simplest: when any joint is within a tolerance of a reach limit then the component of the weighting matrix for that joint is set to a large value (ABIG), otherwise the component is set to unity. ABIG will be a design constant that may be varied to provide the desired performance.

The second algorithm is similar to the first, but has the following requirement. If a joint is moving away from its reach limit then the weighting matrix component for that joint is set back to unity. This encourages the joint to move away from the tolerance zone.

The third algorithm does not use the tolerance test. The value of the weighting matrix component for each joint is scaled from 1 at its midrange value to ABIG at either of its joint reach limits. Also, as in algorithm number 2, if the joint is moving toward its midrange value then the value of the weighting matrix component is set to unity. This algorithm is designed to encourage each joint to stay near the midrange value.

Each of the above three algorithms were implemented into the KRPM model described previously and tested until validated. The algorithms were then used to study a single joint encountering a reach limit and two joints encountering reach limits simultaineously.

To test the ability to avoid a single joint reach limit, take the case where the start and end of a trajectory are known to be valid end-effector states, but a reach limit is encountered without reach avoidance. The trajectory shown in figure 4d was used for this test where the third joint begins at -90 degrees, reaches -106 degrees, and ends at -87 degrees. Suppose that the limit for this joint is at -100 degrees. Figure 6 shows the trace of the third joint with no reach avoidance and for each of the three reach limit avoidance algorithms using a value of 100 for Abig. Each of the algorithms successfully avoided the imposed reach limit. In method 1 the joint angle does not move back out of the tolerance zone of 10 degrees. In methods 2 and 3, the joint moved back out of the reach zone of 10 degrees from the reach limit.

The trajectories with and without reach avoidance are shown in figure 7. Notice that with reach avoidance the first joint moves faster during the first few interations (figures 7b, c, and d) than without reach avoidance (figure 7a).

In figure 8a the arm was commanded from the joint state of (90,0, -135,90) to the hand state of (-.1,-2,90) causing two joints, joint 3 and 4, to approach reach limits. With reach avoidance both joint positions are improved in the final configuration (figure 8b).

In the case shown in figure 9a joint 3 exceeds a -160 degree limit and then goes past -180. With reach avoidance (figure 9b) joint 2 swings out dramatically to allow joint 3 to avoid its reach limit.

6.0 SUMMARY

The pseudo-inverse Jacobian with a weighting matrix has been derived as a resolved rate law for the kinematically redundant manipulator. The resolved rate law has been demonstrated with a kinematically redundant planar manipulator model. Reach avoidance has been mostly successfull with this model by dynamically adjusting the components of

the ng matrix during a maneuver. In some extreme cases the reach limit not avoidable. The locally optimized resolved rate law has been ved by incorporating joint reach avoidance. Reach avoidance has used in the inverse seeking solution to arrive at feasible solut The need has been demonstrated for using small steps sizes in trative inverse seeking algorithm for the purpose of arriving at usoint angle solutions for trajectory planning.

work could be aimed at studying reach avoidance techniques duririal tasks such as cyclic motion. A similar exercise should be ped with a manipulator operating in 6 degrees of freedom as the behawould be different than for the planar arm which operates in a limitne with all pitch joints.

ACKNCENTS

The wishes to express appreciation to Barry Rogers and Susan Rogerthe development of the KRPM model and to Richard Theobald for his ging support and final review of the paper.

7.0 RCES

[1] . Whitney, "The Mathematics of Coordinated Control of stic Arms and Manipulators," *J. of Dynamic Systems, ement and Control,"* Trans. ASME, Vol. 92, pp. 303-309, Dec.

[2] aig, *Introduction to Robotics*, Addison-Wesley, 1986.

[3] nhabib, A.A. Goldenberg, and R.G. Fenton, "A Solution to the Kinematics of Redundant Manipulators," *J. of Robotic*, 2(4), 373-385 (1985).

[4] Klein and C. Huang, "Review of Pseudoinverse Control for Use inematically Redundant Manipulators," *IEEE Trans. Systems, d Cybernetics*, Vol SMC-13, No. 3, March/Apr 1983.

[5] D. Orin, and M. Bach, "An Inverse Kinematic Solution for lically Redundant Robot Manipulators," *J. of Robotic*, 1(3), 235-249 (1984).

[6] Goldenberg, B. Benhabib, and R.G. Fenton, "A Complete lized Solution to the Inverse Kinematics of Robots", *IEEE of Robotics and Automation*, Vol RA-1, No. 1, pp14-20, 1985.

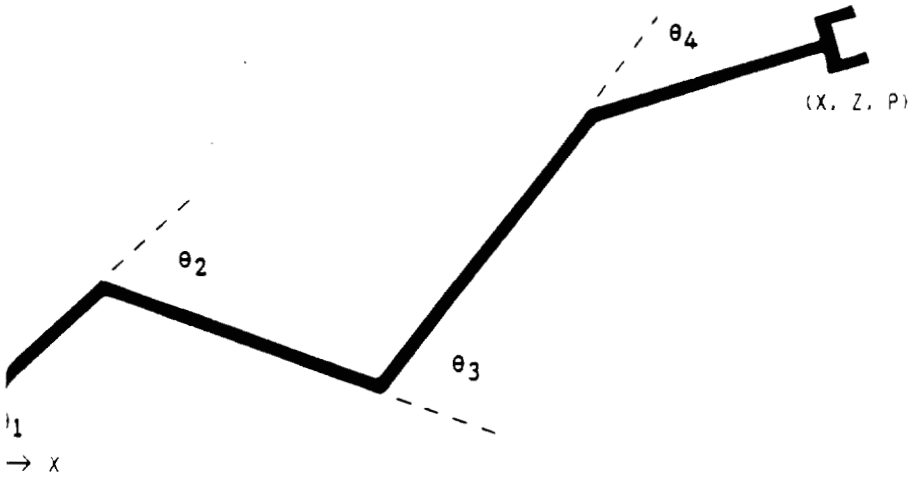
[7] Mati, P. Morasso, and V. Tagliasco, "The Inverse Kinematic for Anthropomorphic Manipulator Arms," *J. of Dynamic, Measurement, and Control*, Vol. 104, pp. 110-113, March

[8] Paul, *Robot Manipulators: Mathematics, Programming and*, MIT Press., Cambridge, MA, 1981.

[9] Fe, Malcom, and Moler, *Computer Methods for Mathematical tion*, Prentice-Hall, NJ, 1977.

[10] , D.E. "Optimum Stepsize Control for Newton-Raphson Solution linear Vector Equations," *IEEE Transactions on Automation*, Vol. AC-14, No. 5, pp.572-74, Oct. 1969.

[11] lik, Raymond L., *Theory and Techniques of Optimization For ing Engineers*, Barnes & Nobles, NY, 1971.



: 1. A kinematically redundant planar manipulator (KRPM) four joints and a task space of X, Z, and Pitch (P).

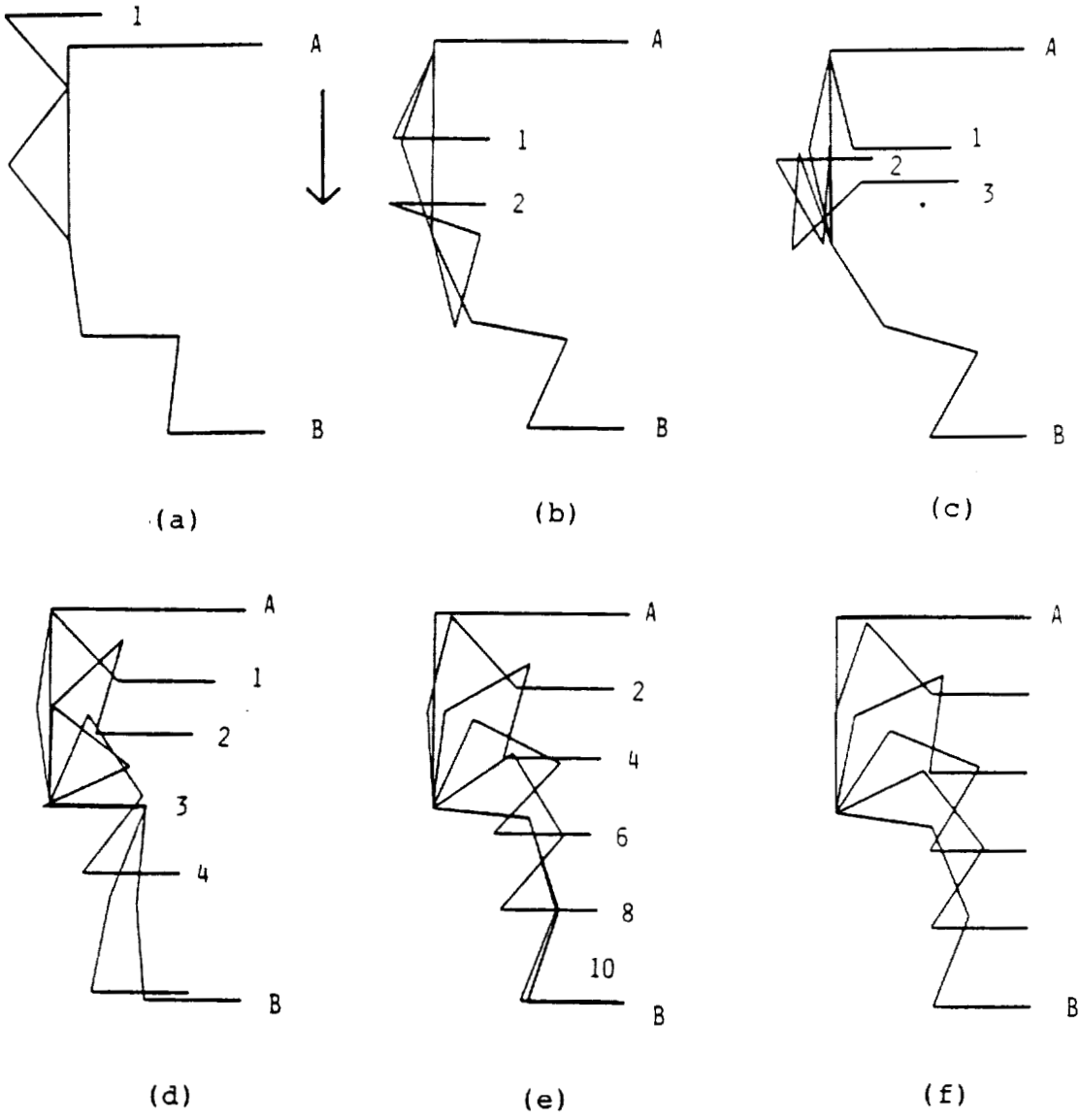


Figure 2. When a full step is taken from A to B, the first step (indicated by a "1") goes the wrong way. As smaller step are taken from A to B by dividing the path into 2 (b), 3(c), 5(d), 10(e), and 80 (f) equal steps before taking full steps to get to B, the trajectory becomes straight and each step goes in the correct direction.

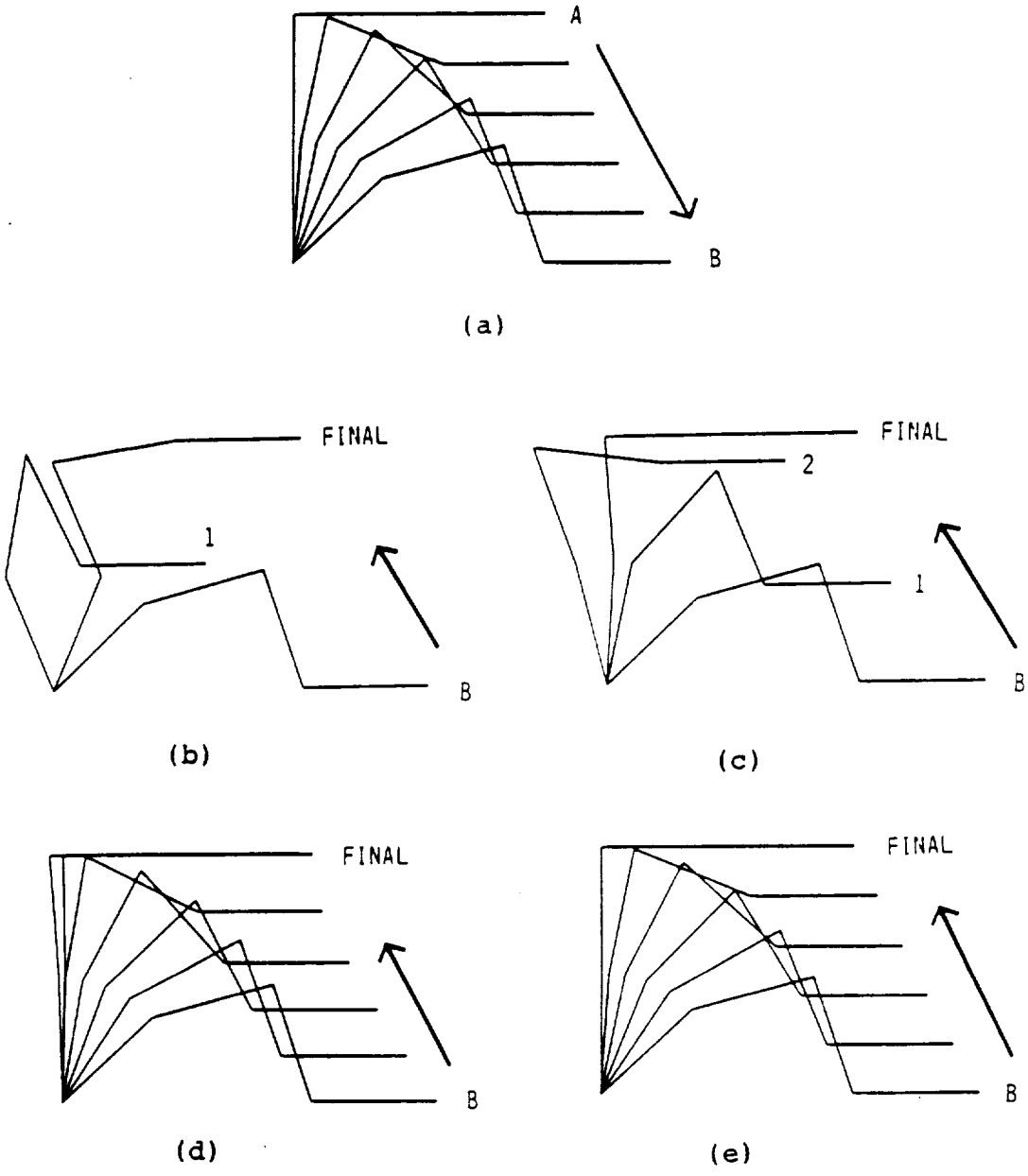


Figure 3. The arm was moved from A to B taking small steps (a). Stepping back to A with a full step iteration does not achieve the same joint angles as A. As smaller steps are taken as shown in the 2 step (b), 5 step (c), and 40 step (d) iterations the original configuration of A in (a) is approached.

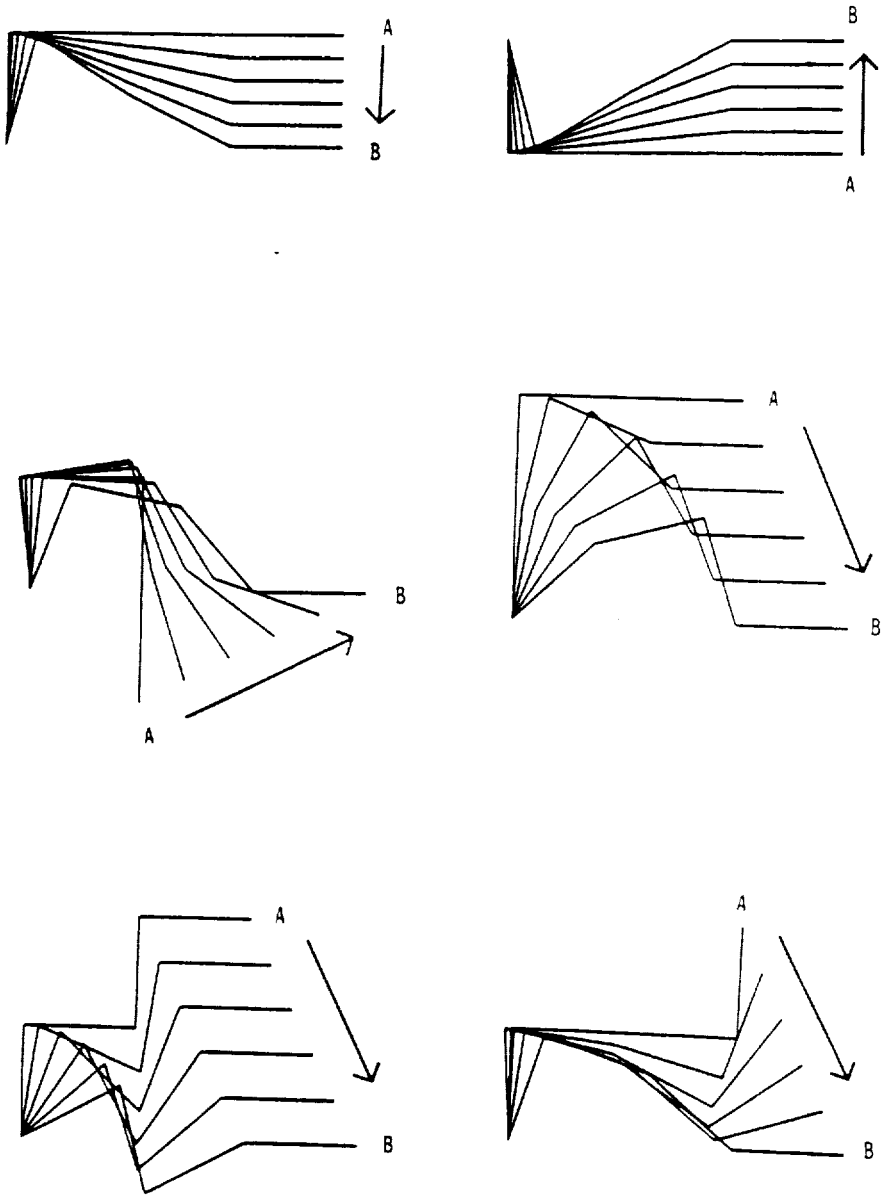
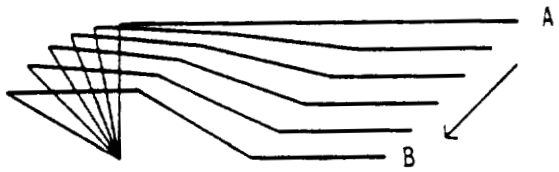
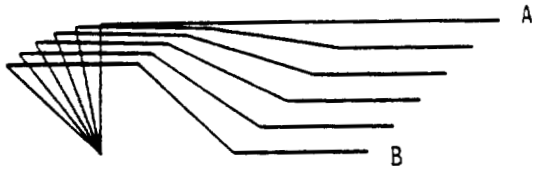


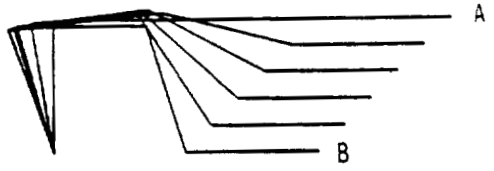
Figure 4. The KRPM is commanded to the same hand state of 3,0,0 at B from six different initial joint angle states at A to demonstrate the behaviour of the RRL.



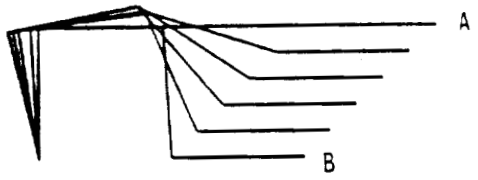
(a)



(b)



(c)



(d)

Figure 5. Effects of the weighting matrix on the trajectory are shown for values of A(2,2) set at 1 (a), 2 (b), 10 (c), and 100 (d).

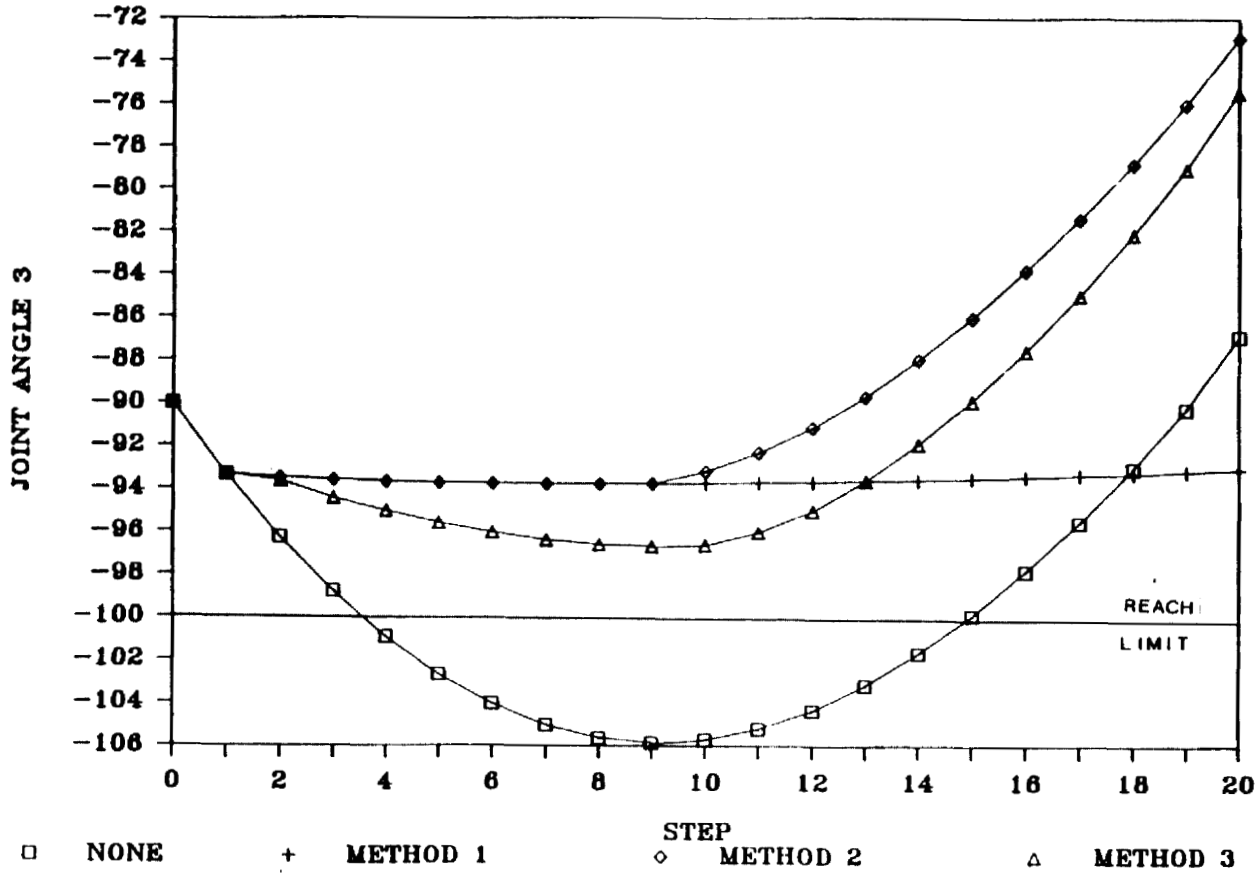


Figure 6. The effects of reach avoidance on joint 3 during a command from a joint state of 90,0,-90,0 to a hand state of 3,0,0. The reach limit is successfully avoided for each of the 3 reach avoidance algorithms.

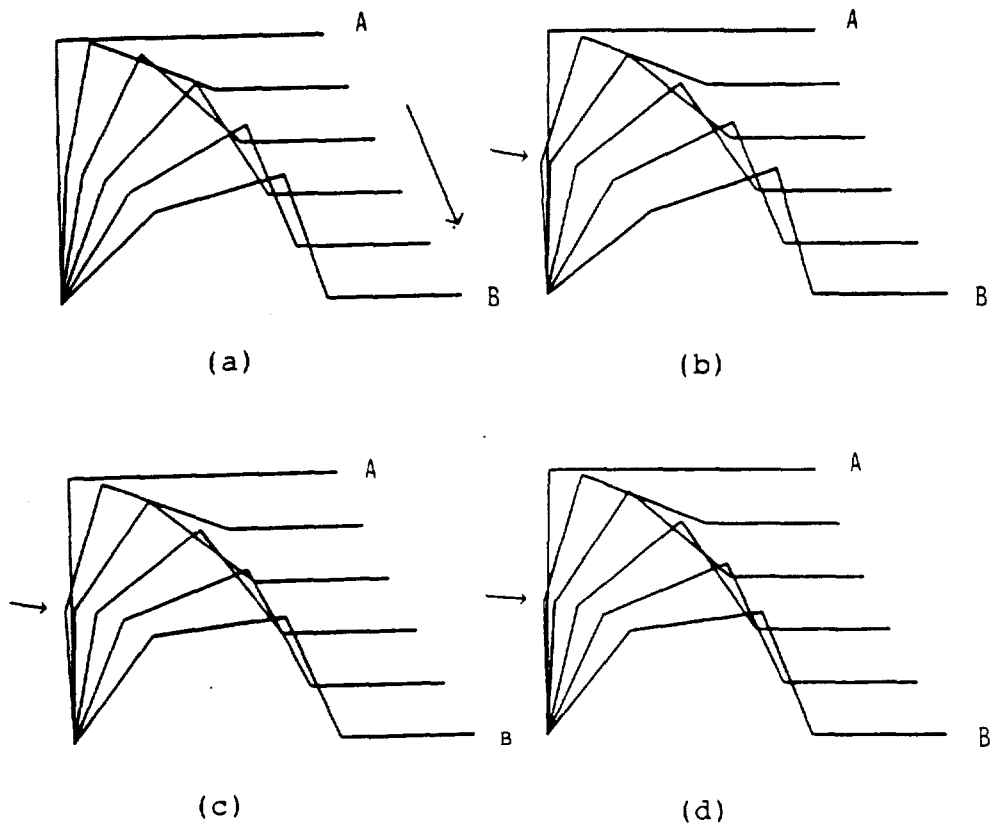


Figure 7. The trajectories are shown for the case of Figure 6 for no reach avoidance (a) where joint 3 violates a limit of -100 degrees, and for each of the reach avoidance algorithms: method 1 (b), method 2 (c), and method 3 (d) where the reach limit is avoided.

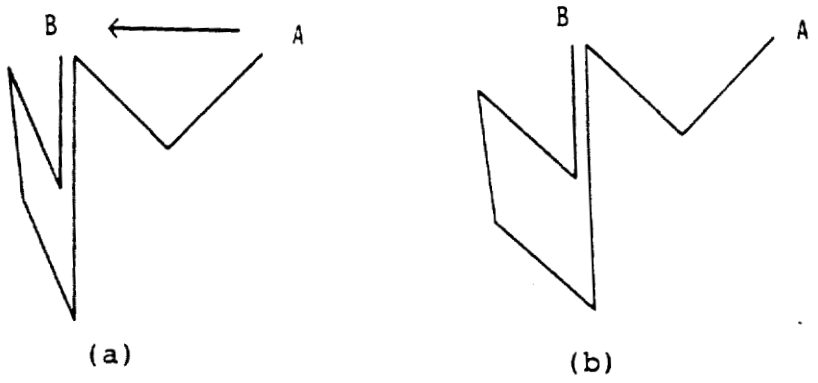


Figure 8. Reach limit avoidance demonstration for the case of two joints violating reach limits. In (a) joints 3 and 4 violate reach limits of -160 and 160. With reach avoidance (b) both reach limits are avoided simultaneously.

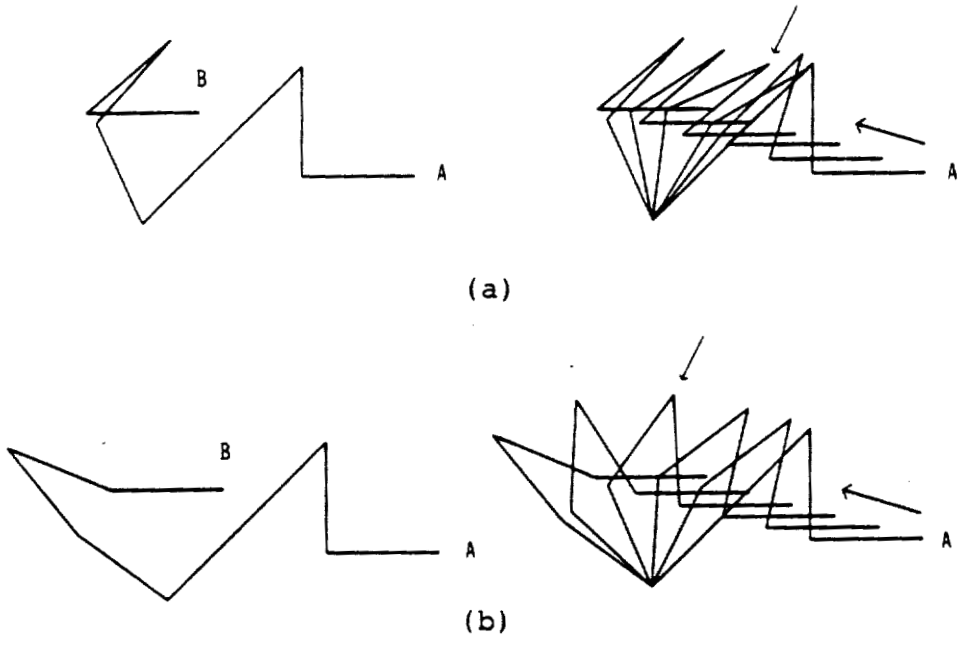


Figure 9. In (a) joint 3 exceeds -180 degrees. This problem is avoided in (b) when reach avoidance is used. The intermediate arm positions are shown to the right.

AN ADAPTIVE CONTROL SCHEME FOR A FLEXIBLE MANIPULATOR

T. C. Yang*and J. C. S. Yang†
Robotics Laboratory
Department of Mechanical Engineering
University of Maryland
College Park, Maryland 20742

and

P. Kudva‡
Advanced Technology and Research, Inc.
3933 Sandy Spring Road
Burtonsville, Maryland 20866

April 13, 1987

Abstract

The problem of controlling a single link flexible manipulator is considered. A self-tuning adaptive control scheme is proposed which consists of a least squares on-line parameter identification of an equivalent linear model followed by a tuning of the gains of a pole placement controller using the parameter estimates. Since the initial parameter values for this model are assumed unknown, the use of arbitrarily chosen initial parameter estimates in the adaptive controller would result in undesirable transient effects. Hence, the initial stage control is carried out with a PID controller. Once the identified parameters have converged, control is transferred to the adaptive controller. Naturally, the relevant issues in this scheme are tests for parameter convergence and minimization of overshoots during control switch-over. To demonstrate the effectiveness of the proposed scheme, simulation results are presented with an analytical nonlinear dynamic model of a single link flexible manipulator.

*Research assistant
†Director, Robotics Laboratory
‡Research Engineer

1 INTRODUCTION

Automated manipulation is finding increasing use in production, military and space industries for performing routine, monotonous and hazardous tasks. The present day manipulators can perform with sufficiently adequate accuracy at the expense, however, of payload capacity and operating speed. One possible cost-effective solution is to build manipulators with lighter links. The lightweight links reduce the moment of inertia at each joint, permit the use of direct drive motors and have the advantages of manufacturing simplicity and lower cost. The next generation of manipulators would naturally have to be flexible. Mechanical flexiblity, however, generates a fairly severe problem of control of the manipulator end effector motion in its work space. This is due to the inevitable excitation of structural vibrations and the resulting interactions between these vibrations and the control action which would effect the accuracy required of the manipulator.

The successful implementation of flexible robots is contingent upon achieving acceptably uniform performance with regard to variations in load , task specification, reasonable speeds and the ability to compensate for any environmental disturbances. In contrast to robots consisting of rigid links, the dynamic behavior of flexible manipulators is not easy to characterize, especially under conditions of high speed and large amplitude motion. It is not only the fact that this behavior is described by highly nonlinear differential equations but also the lack of a precise knowledge of this description that makes the design of an acceptable control system, over the total range of its operation, a formidable task. The dynamic effects due to changes in configuration, load and moments of inertia, higher speed and unpredictable disturbances tend to degrade the performance of the flexible manipulator arm. The control scheme that is to be developed, therefore, has to overcome these dynamic effects.

This paper attempts at a resolution of difficulties posed by this problem by employing a self-tuning control approach. The strategy here briefly consists of (i) a least squares on-line parameter identification of an equivalent linear model, followed by (ii) a tuning of the controller gains by an adaptive control algorithm throughout

the range of the manipulator operation. Thus any changes occuring in the manipulator's dynamic description will automatically be reflected in the parameter estimates and would, therefore, be counteracted by updating the controller gains.

An important step prior to parameter identification is to obtain a valid model structure of the manipulator dynamics. This is derived by analytical modelling based on Lagrange's equation and assumed mode shape functions from the finite element method. This nonlinear analytical model is used to generate the input/output data which, in turn, is employed in the least squares parameter estimation. Since, initially, the parameters are assumed to be unknown, the parameter estimates obtained during this initial stage would be unsuitable for updating the controller gains. Hence, during this initial stage, a simple PID stabilizing controller is used with the manipulator model and the parameter identification process is initiated. On convergence of the parameters, the control action is switched over to the adaptive controller. A salient feature of the present work involves the implementation of a convergence test to minimize any undesirable transient effects following the switch-over.

2 THE ADAPTIVE POLE PLACEMENT CONTROLLER

The control scheme considered here is based on adaptive pole placement. While a variety of configutations can be found in the literature ([1], [3], [4]) for pole placement, the one involving a Luenberger observer structure (Fig. 1) as suggested by Elliot and Wolovich [1] is used here. This choice is based on the fact that it results in a closed loop system of the same order as the open-loop system (due to pole-zero cancellations). Also it does not add any undesirable zeros to the plant as might happen with the structure suggested in [3].

The adaptive pole-placement concept is briefly presented below in a discrete-time framework:

Let the plant to be controlled have the transfer function

$$\frac{B(q^{-1})}{A(q^{-1})}. \tag{1}$$

where q^{-1} is the backward shift operator, and

$$A(q^{-1}) = 1 + \sum_{i=1}^{n} a_i q^{-i} \tag{2}$$

$$B(q^{-1}) = \sum_{i=1}^{n} b_i q^{-i} \tag{3}$$

so that it has the description

$$A(q^{-1})y(t) = B(q^{-1})u(t). \tag{4}$$

where u(t) and y(t) are the input and output respectively.

The Adaptive Pole Placement Algorithm :

From the structure of Figure 1, one can formulate the following equations.

$$Q(q^{-1})g(t) = \hat{K}(t, q^{-1})u(t) + \hat{H}(t, q^{-1})y(t) \tag{5}$$

$$u(t) = g(t) + v(t) \tag{6}$$

where

$$\hat{K}(t, q^{-1}) = \sum_{i=1}^{n} \hat{k}_i(t) q^{-i} \tag{7}$$

$$\hat{H}(t, q^{-1}) = \sum_{i=1}^{n} \hat{h}_i(t) q^{-i} \tag{8}$$

and

$$Q(q^{-1}) = 1 + \sum_{i=1}^{n} q_i q^{-i} \tag{9}$$

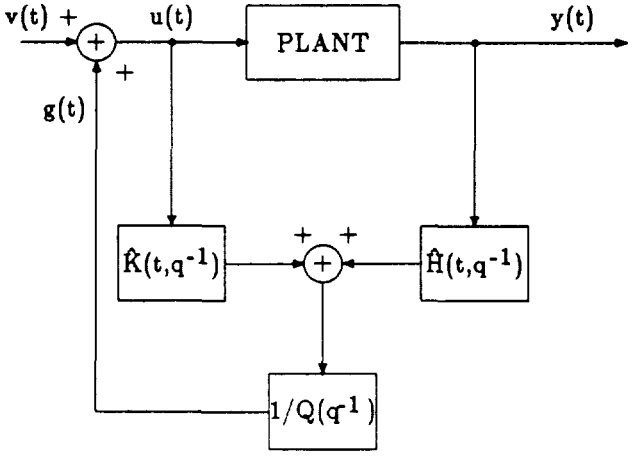


Figure 1: Luenberger observer structure for pole placement control

Let

$$\hat{A}(t,q^{-1}) = 1 + \sum_{i=1}^{n} \hat{a}_i(t) q^{-i} \tag{10}$$

$$\hat{B}(t,q^{-1}) = \sum_{i=1}^{n} \hat{b}_i(t) q^{-i} \tag{11}$$

where $\hat{a}_i(t)$ and $\hat{b}_i(t)$ are the estimates of a_i and b_i.

If $\hat{K}(t,q^{-1})$ and $\hat{H}(t,q^{-1})$ are made to satisfy the following relation

$$\begin{aligned} & \hat{H}(t,q^{-1})\hat{B}(t,q^{-1}) + \hat{K}(t,q^{-1})\hat{A}(t,q^{-1}) \\ = \; & Q(q^{-1})[\hat{A}(t,q^{-1}) - A_d(t,q^{-1})] \end{aligned} \tag{12}$$

then the resulting closed-loop transfer function becomes

$$\frac{B(q^{-1})}{A_d(t,q^{-1})} \tag{13}$$

when the identified parameters converge to the plant parameters, where

$$A_d(t,q^{-1}) = 1 + \sum_{i=1}^{n} A_{di}(t)q^{-i} \tag{14}$$

With this structure, however, the plant cannot be made to track a step input signal. In order to equip this structure with such a tracking facility, unity feedback is applied and an integrator is inserted in the forward path. This can be formulated as

$$w(t) = s(t) \cdot c + w(t-1) \tag{15}$$

$$s(t) = v(t) - y(t) \tag{16}$$

Also equation (6) should be modified to :

$$u(t) = g(t) + w(t) \tag{17}$$

The desired denominator $A_d(t,q^{-1})$ and the scalar gain c can be determined from the desired closed-loop denominator $D(q^{-1})$ by the following equations.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & \hat{b}_1 \\ -1 & 1 & 0 & 0 & \cdots & \hat{b}_2 \\ 0 & -1 & 1 & 0 & \cdots & \hat{b}_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & -1 & 1 & \hat{b}_n \\ 0 & 0 & 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} A_{d1} \\ A_{d2} \\ A_{d3} \\ \vdots \\ A_{dn} \\ c \end{pmatrix} = \begin{pmatrix} d_1 + 1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \\ d_{n+1} \end{pmatrix} \tag{18}$$

where

$$D(q^{-1}) = 1 + \sum_{i=1}^{n+1} d_i q^{-i} \tag{19}$$

Since we can obtain $A_d(t,q^{-1})$ from equation (18) , the $\hat{H}(t,q^{-1})$ and $\hat{K}(t,q^{-1})$ can be obtained from equation (12).

The block diagram of this scheme is shown in figure 2.

Note that the step input tracking facility is achieved by increasing the order of the overall system to only n+1.

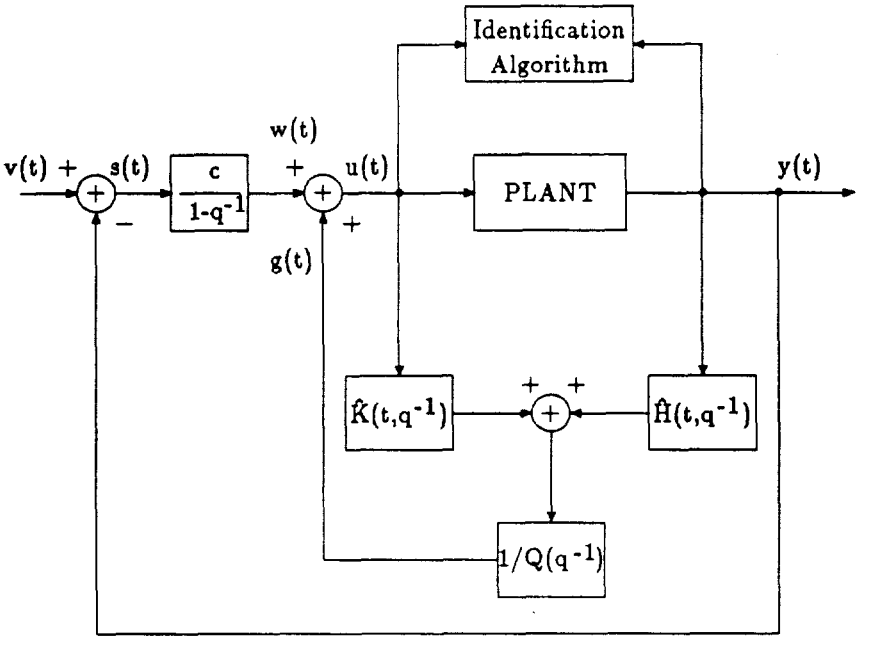


Figure 2: Adaptive Pole Placement Control Scheme Using Luenberger Observer Structure with Integral Action

The Least Squares Identification Algorithm:

The estimates $\hat{a}_i$ and $\hat{b}_i$ used in the control scheme are obtained by a least squares parameter identification algorithm [4] as follows:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \frac{P(t-2)\Phi(t-1)[y(t) - \Phi(t-1)^T\hat{\theta}(t-1)]}{1 + \Phi(t-1)^T P(t-2))\Phi(t-1)}; \quad (20)$$

$$t \geq 1$$

$$P(t-1) = P(t-2) - \frac{P(t-2)\Phi(t-1)\Phi(t-1)^T P(t-2)}{1 + \Phi(t-1)^T P(t-2)\Phi(t-1)} \quad (21)$$

with $\hat{\theta}(0)$ given and $P(-1)$ is any positive definite matrix, where

$$\hat{\theta}(t) = [-\hat{a}_1(t), -\hat{a}_2(t), \cdots, -\hat{a}_n(t), \hat{b}_1(t), \hat{b}_2(t), \cdots, \hat{b}_n(t)]^T$$

is the current parameter estimate vector, and

$$\Phi(t-1) = [y(t-1), y(t-2), \cdots, y(t-n), u(t-1), u(t-2), \cdots, u(t-n)]^T$$

3 SWITCH-OVER FROM PID TO ADAPTIVE CONTROLLER

A critical question in the present control scheme is to determine an appropriate time to switch from the initial PID controller to the adaptive controller. The simplest way is to wait till the parameter estimates resulting from the identification algorithm have converged to their true values. The following criterion provides a check on such a convergence.

The Convergence Criterion:

Assume $\| \tilde{\theta}(0) \|^2 \leq M$.
If

$$\lambda_{max}[P(t-1)] \leq \frac{\epsilon \lambda_{min}[P(-1)]}{M} \tag{22}$$

then

$$\| \tilde{\theta}(t) \|^2 \leq \epsilon, \tag{23}$$

where

$\tilde{\theta}(t) = \hat{\theta}(t) - \theta$
$\hat{\theta}(t)$: identified parameter vector at time t
θ : actual parameter vector = $[-a_1, -a_2 \cdots, -a_n, b_1, b_2, \cdots, b_n]^T$
ϵ : the error tolerence for the convergence test of the identified parameters.
$\lambda_{max}[P(t-1)]$: maximum eigen value of $P(t-1)$.

Proof:

From [4] (p. 61), one can get the following inequality

$$\lambda_{min}[P(t-1)^{-1}] \| \tilde{\theta}(t) \|^2 \leq \lambda_{max}[P(-1)^{-1}] \| \tilde{\theta}(0) \|^2 \quad (24)$$

which implies

$$\| \tilde{\theta}(t) \|^2 \leq \frac{\lambda_{max}[P(t-1)]}{\lambda_{min}[P(-1)]} \| \tilde{\theta}(0) \|^2 \quad (25)$$

Using (22) and (25), it follows that

$$\| \tilde{\theta}(t) \|^2 \leq \frac{\epsilon \| \tilde{\theta}(0) \|^2}{M} \leq \epsilon \quad (26)$$

Thus by computing $\lambda_{max}[P(t-1)]$, one can test the convergence of the parameter estimates.

Switching Logic:

Once the identified parameters have converged to their true values and the system step response has reached steady state, control action is switched over from the PID to the adaptive controller. This is probably the simplest manner to implement the switch-over without causing any undesirable transients.

An alternative switching logic is proposed here which does not require the step response to reach steady state. However, this logic is limited only to those systems that satisfy the conditions for one-step-ahead control [4].

Assume

(i) the plant to be linear time invariant,

(ii) the switching instant to correspond to $t = 0$, and

(iii) the desired output trajectory after switching to be the same as the one that would have been obtained, had the adaptive controller been applied to the plant starting at rest from that position(y_o), where

$$y_o = [y(-1), y(-2), \cdots, y(-n), u(-1), u(-2), \cdots, u(-n)]^T \theta \quad (27)$$

is the output at the switching instant.

In order to satisfy the last assumption, a correction input $u_c(t)$ is needed to compensate for the terminal conditions resulting from the PID controller. Thus the plant input would be

$$u(t) = u_d(t) + u_c(t) \tag{28}$$

where $u_d(t)$ is the input generated by the pole placement algorithm.

The plant output can be expressed as

$$y(t) = y_d(t) + y_c(t) \tag{29}$$

where

$$y_d(t) = \Phi_d{}^T(t)\theta \quad ; y_d(0) = 0 \tag{30}$$

$$y_c(t) = \Phi_c{}^T(t)\theta \quad ; y_c(0) = y_o \tag{31}$$

where the subscripts denote the correspondence of the two components.

From assumption (iii), $y_c(t) = y_o$ for $t \geq 0$, and the compensating input $u_c(t)$ is obtained using (31) as:

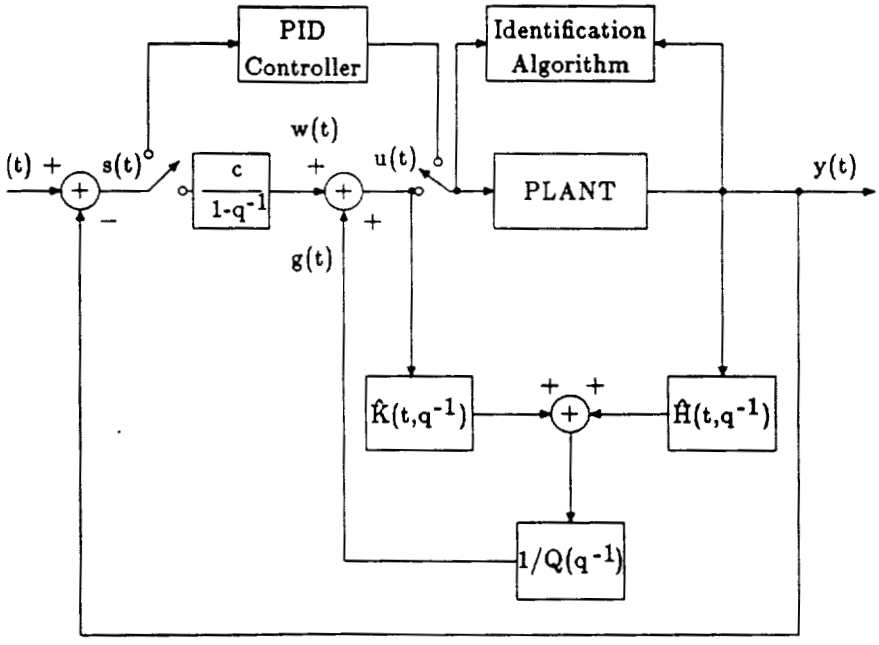
$$u_c(t) = \frac{1}{b_1}\{y_o - [y_c(t), y_c(t-1), \cdots, y_c(t-n+1), 0, u_c(t-1), \cdots, u_c(t-n+1)]^T\theta\} \tag{32}$$

With a proper choice of the sampling interval, the flexible manipulator discrete model is found to meet the requirements of one-step-ahead control. However, this approach is found to be suitable only in those situations where the deflections are small, and is not used in the simulation here.

The complete control block diagram is shown in Fig. 3 .

4 SIMULATION RESULTS

The dynamic analytical model of the single link flexible manipulator is described by [5]:



ure 3: Complete Block Diagram of PID-Adaptive Pole Placement Controller

$$\begin{pmatrix} (J+\sum_{i=1}^{N} m_{ii}r_i^2) & m_{1\beta} & m_{2\beta} & \cdots & m_{N\beta} \\ m_{1\beta} & m_{11} & 0 & \cdots & 0 \\ m_{2\beta} & 0 & m_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & 0 \\ m_{N\beta} & 0 & 0 & \ldots & m_{NN} \end{pmatrix} \begin{pmatrix} \ddot{\beta} \\ \ddot{r_1} \\ \ddot{r_2} \\ \vdots \\ \ddot{r_N} \end{pmatrix} = \begin{pmatrix} u - 2\sum_{i=1}^{N} m_{ii}r_i\dot{r}_i \\ (\dot{\beta}^2 - \omega_1^2)r_1 \\ (\dot{\beta}^2 - \omega_2^2)r_2 \\ \vdots \\ (\dot{\beta}^2 - \omega_N^2)r_N \end{pmatrix} \tag{33}$$

$$y = tan^{-1}\frac{w(l,t)}{l} + \beta \tag{34}$$

$$w(x,t) = \sum_{i=1}^{N} \phi_i(x) r_i(t) \tag{35}$$

where

u : input torque to the beam
y : tip position
l : the length of the beam
β : the hub angle
$\phi_i(x)$: the mode shape functions of the beam
$r_i(t)$: the generalized coordinates

The desired closed-loop denominator for the adaptive pole placement controller is chosen as

$$D(q^{-1}) = 1 + d_1 q^{-1} + d_2 q^{-2} \tag{36}$$

where

$$d_1 = -2e^{-\xi\omega T} cos\ \omega T\sqrt{1-\xi^2} \tag{37}$$

$$d_2 = e^{-2\xi\omega T} \tag{38}$$

where T is the sampling period in seconds.

For computer simulation, the following numerical values are used: $n = 4$, $N = 2$, $\omega = 5$, $\xi = 1$, $T = 0.1$, $P(-1) = 10^8 I_{8\times 8}$, $M = 10$ and $\epsilon = 0.7$. The switching from PID to the adaptive controller occurs at $t = 4$ secs.

The results are shown in Fig. 4.

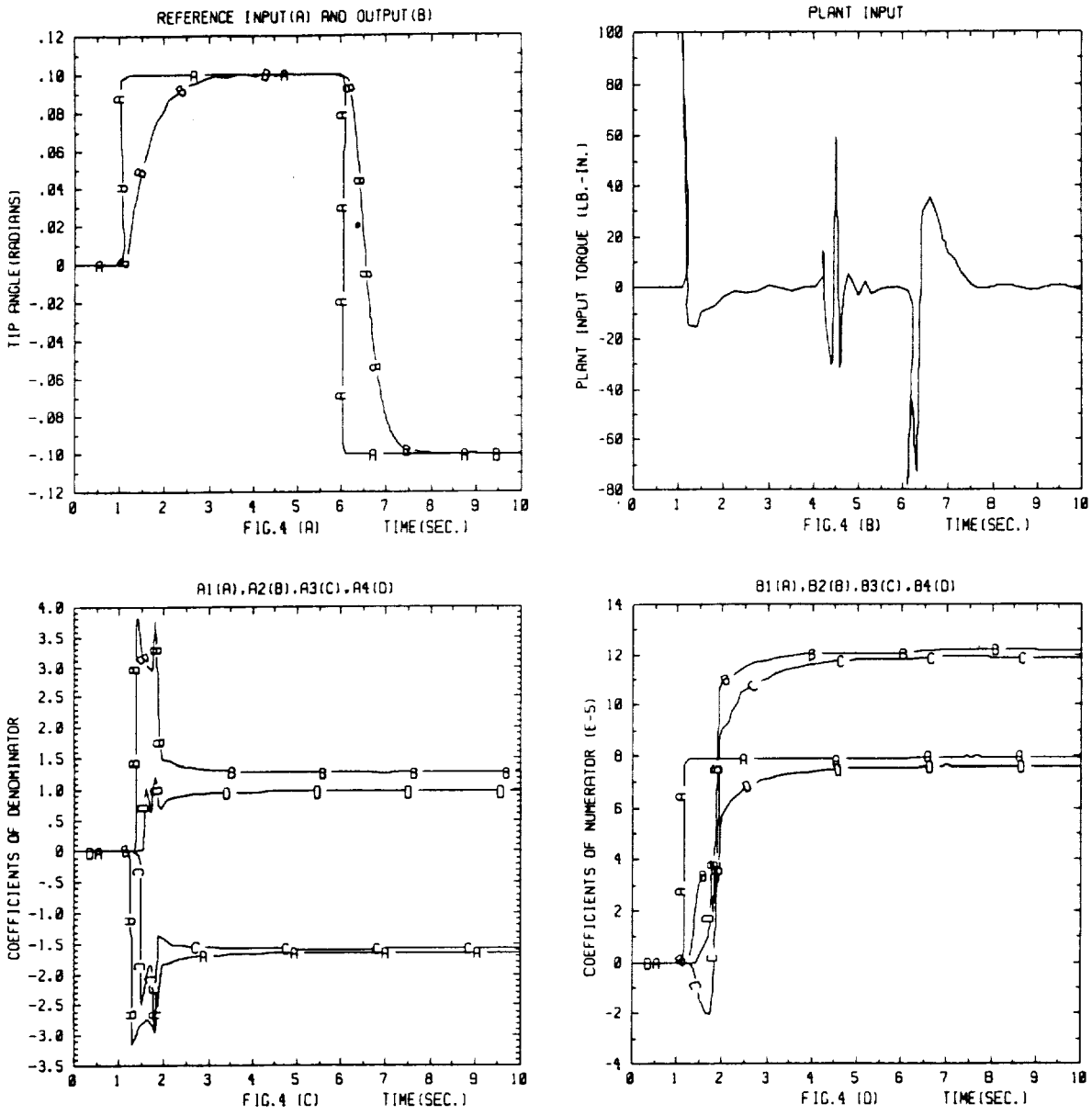


Figure 4: Simulation of Combined PID and Adaptive Pole Placement Control:
(A) reference input v(t) and plant output y(t),
(B) plant input u(t),
(C) convergence of identified coefficients of denominator,
(D) convergence of identified coefficients of numerator.

5 CONCLUSION

A simulation based study for the adaptive control of a single link flexible manipulator has been considered. Such a control approach is of practical importance since the dynamic characteristics of the manipulator change considerably especially while picking up or releasing payloads. In such cases unless the control gains are suitably updated, the performance would be poor.

Since the adaptive control scheme depends on the parameter estimates from an on-line identification algorithm, the initial control action is carried out with a PID controller during which the identification process is initiated. On convergence of the parameter estimates, control is smoothly transferred to the adaptive controller. A criterion for testing the convergence has been presented. The simulation results amply demonstrate the effectiveness of the proposed scheme.

Experimental verification of the control scheme on a laboratory test set-up is presently in progress.

References

[1] Elliot, H. and W. A. Wolovich, " *Parameter Adaptive Identification and Control* " IEEE Trans. Auto. Control, vol. AC-24, no. 4, pp. 592-599, Aug. 1979.

[2] Elliott, H., R. Cristi and M. Das, " *Global Stability of Adaptive Pole Placement Algorithms* " , IEEE Trans. Auto. Control, vol. AC-30, no. 4, pp. 348-356, April 1985.

[3] Goodwin, G. C. and K. S. Sin, " *Adaptive Control of Nonminimum Phase Systems* ", IEEE Trans. Auto Control, vol. AC-26, no. 2, pp. 478-483, April 1981.

[4] Goodwin, G. C. and K. S. Sin,"*Adaptive Filtering Prediction and Control* ", Prentice Hall, 1984.

[5] K-H. Sung and J. C. S. Yang,"*A Dynamic Study of A Flexible One-Link Manipulator: Analytical and Experimental Approach* " , to be presented at 2nd Intl. conf. on Robotics and Factories of the Future, San Diego, CA. July 1987.

Robot Sensor Language

Stephen Leake
National Bureau of Standards

Abstract

RSL (Robot Sensor Language) is a data-driven, semi-interpreted, hierarchical, user extensible, robot task description language. It provides four levels of task decomposition, with structures and syntax specialized for each level. The user can add commands for new sensors appropriate to the task at hand. The language is highly interactive, easing debugging and algorithm development. It may also be used as an interface to a task planning system.

Introduction

RSL is a response to the need for a robot language that supports user-designed sensors, along with hierarchical task decomposition and real-time execution. It is written in RCS, the NBS - developed Real-time Control System[1], and runs on 8086 based Multibus hardware.

RSL is a high-level language, specialized for sensor-interactive robot tasks. It is data-driven in the sense that all data relating to a particular task is separated from the control process that executes the task. This makes programming different tasks a matter of changing data files, rather than changing control code, leading to a much more reliable control system. Data describing a robot task is of two types: environmental data, such as object sizes and positions; and algorithms, which give information about what sequence of steps are needed to complete the task. RSL supports representation of both types of data in high-level source code. In order to speed execution, this high-level source code is first compiled into a linked list representation stored in common memory, which is then interpreted by control levels. Any piece of RSL source code can be edited and re-compiled at any time; the linked lists are updated, with garbage collection, and the result can be executed immediately. It is not necessary to re-compile an entire application to make small changes.

RSL supports hierarchical task decomposition. The high-level language is explicitly hierarchical, decomposing tasks into paths, path-points, and trajectories. The compiler and control levels also follow this hierarchy, and are highly modular. This structure makes it easy to add new sensors, or new functional capabilities at any level.

The rest of the paper is organized as follows: section 1 gives an introduction to RSL structures, and introduces a short example task used to illustrate the language. Section 2 discusses the implementation of the compiler and control levels. Section 3 gives a brief overview of some applications that have used RSL. Finally, section 4 concludes with a discussion of future work.

Commercial equipment is identified in this paper in order to adequately describe the systems under development. In no case does such identification imply recommendation by the National Bureau of Standards, nor does it imply that this equipment was necessarily the best for the purpose.

Author's current address: National Bureau of Standards, Bldg. 220, rm B127, Gaithersburg, MD 20899

Section 1 : RSL structures

Task decomposition

It is helpful to use a simple example to illustrate the task decomposition. Consider moving a box from a truck to a conveyor, using a forklift end-effector equipped with sonar range sensors and proximity sensors. The first level of task decomposition (called the task level), yields the following sequence of steps:

1) Move the fork to the vicinity of the truck.
2) Using long range sensors, find the approximate position of the box on the truck, then use short range sensors to align the fork tines with the box, and insert the tines under the box.
3) Lift the box clear of the truck.
4) Move to near the conveyor
5) Gently set the box on the conveyor. (The position of the conveyor is known to the robot controller, so no sensors are needed for this step.)
6) Remove the tines from under the box, and move clear.

The next level of decomposition is the path level. Each of the steps in the task level descriptions is, in fact, a path. Continuing the example, the path for step 2 decomposes into:

1) Scan across the truck, reading the long range sonars.
2) Goto 20 inches (as measured by sonar) in front of the closest point seen in the scan.
3) Move to the pickup side of the box.
4) Use several sonars to align with the floor of the truck, and the side of the box.
5) Insert the tines under the box, using proximity sensors to avoid hard collisions.

Each step in a path is called a path-point. The names path and path-point are derived from the notion of thru-points along a path used to go around obstacles, but the meaning here is generalized to include the use of sensors in various ways. A pre-planned path is a simple case, where no sensors are used.

The last level of decomposition is the trajectory level. Each path-point decomposes into one or more trajectories. For example, step 1 in the path above decomposes simply into a Cartesian straight-line trajectory, while step 4 involves several sensor-servoed rotations.

Each level of decomposition is now discussed in more detail, starting from the bottom.

Trajectory

A trajectory is an algorithm for commanding the position of the end-effector as a function of time. It may be calculated solely on the basis of a priori information (as in " goto the truck "), or incorporate sensor feedback (as in " align to the box "). The trajectory algorithms typically take parameters describing the goal pose and limits on the velocity and acceleration. Sensor-based algorithms will have more parameters describing how to use the sensor data. RSL provides four trajectory commands: Cartesian straight-line and joint interpolated for point-to-point motion, and two others for real-time Cartesian sensor servo[2]. The user may add other trajectory commands as needed.

Path-points

A path-point is an algorithm for a single motion of the end-effector, usually involving a sensor. For example, path-point number 2 above commands a motion towards a point, and uses the sonar to measure the distance remaining. When the distance drops to 20 inches, the motion is halted. This is a motion terminated by a sensor condition. On the other hand, the path-point that aligns the fork with the floor is sensor controlled: the sonars give the distance to the floor, and a rotation is computed that

moves towards alignment. The orientation of the fork is servoed to the floor orientation.

There is typically a group of path-point commands for each type of sensor. For example, sonars are used in scan, range, and align path-points. Each path-point command takes parameters which identify the individual sensor to use, and give information on how to use the sensor data. For example, the align path-point command has parameters identifying two sonars, a rotation axis, and a goal orientation.

The base RSL system provides only the **goto** path-point command, which moves the end-effector to a given location, using no sensors. Users must add other path-points to use their sensors. Since the language was designed with user extensions in mind, this is easy to do.

Paths

A path is an algorithm for a simple task, such as moving between locations or grasping objects. The algorithm may be simply a path in space that guarantees no collisions, or it may involve sensors to help locate the object when its position is not accurately known. In the above example, the first path uses no sensors - it simply moves to the truck. The second path uses sensors to find the box. For simplicity, paths consist of a linear sequence of path-points. Any branching or looping must be done within a path-point, or at the task level.

Individual paths are identified by a path type and parameters. The parameters typically consist of named locations, objects, and tools that are involved in the task. The path type gives the intended purpose of the path, and is used in the task level decomposition. There are six path types provided by RSL; ***move-to, approach-pickup, depart-pickup, approach-release, depart-release,*** and ***named.*** The first five path types are used in the **TRANSFER** task, discussed below. They correspond to the paths in the example: step 1 is a ***move-to*** path, step 2 is ***approach-pickup,*** step 3 is ***depart-pickup,*** step 4 is ***move-to,*** step 5 is ***approach-release,*** and step 6 is ***depart-release.***

The sixth path type, ***named,*** is provided mainly for debugging; it provides a simple way to test small pieces of more complex paths. It also provides a way to program simple tasks that do not need a task level of decomposition.

The path parameters are handled in a way that allows a single path definition to be used for several related tasks. For example, the ***approach-pickup*** path that finds the box on the truck could also be used to find the box anywhere else, or it could find different sized boxes, since the location of the truck and the size of the box are path level parameters.

Tasks

For the task level, the term "task" is used in a specific way; RSL tasks are algorithms for high-level functions such as transferring pallets from a truck to a conveyor, or deburring machined parts.

All tasks are decomposed into a sequence of path types; the specific path to be executed is identified by the path type and the parameters of the current task. The user provides paths for each path type/parameter combination required.

The base RSL system provides two tasks; **MOVE-TO** and **TRANSFER. MOVE-TO** simply moves the robot to a goal location, by executing the ***move-to*** path that connects the current location to the goal. **TRANSFER** transfers objects from a source location to a goal location. This is the task used in the example. The sequence of paths for the **TRANSFER** task is;

TRANSFER *object, source location, goal location*
1) ***move-to*** *object , current location, source location*
2) ***approach-pickup*** *object , source location*
3) ***depart-pickup*** *object , source location*
4) ***move-to*** *object , source location, goal location*
5) ***approach-release*** *object ,goal location*
6) ***depart-release*** *object ,goal location*

This sequence decomposes the transfer task into six steps, each of which is programmed by the user as a path. The sequence is repeated if either the source or the destination is an array. Note that the user may treat each object / location combination differently, using different sensor based strategies, while maintaining the high-level **TRANSFER** task definition. For example, transferring a large box from a table would involve using sonar sensors to find the box, while transferring a small machined part from the same table would involve a vision sensor. The user would provide two different sets of paths, identified by the object type. The **TRANSFER** command would automatically select the appropriate path, based on the object type in the task parameters.

The user can add other tasks to RSL, to fit the user's application.

Environmental data

RSL provides ways of representing poses (position and orientation) of locations and objects. Locations can be defined in absolute world coordinates, or relative to a base location, using movetables. Movetables provide a convenient user syntax for specifying relative transforms; the transform is built up out of simple steps, consisting of a vector translation, or a rotation about a single axis. Some information about sizes of objects is also represented, for use by a gripping end-effector. Locations can be grouped into arrays, for use in palletizing operations.

Section 2 : Implementation

RSL is implemented using the NBS Real-time Control System (RCS). RCS is a micro-processor based system for real-time control applications. It runs on Multibus based 8086 / 8087 hardware. The operating system is based on FORTH, but has been significantly extended to support multiple processors, a mid-level high-speed compiled language, inter-processor communications, and common memory. It inherits from FORTH the user-friendly features of interactive execution and incremental compilation, making debugging a simple and easy process.

RCS is designed to support a hierarchical control structure, with all levels of the hierarchy executing in parallel, in a cyclic manner[6]. A system clock defines a cycle time, and each level executes its control process once each cycle. This cyclic execution means that each level in the hierarchy is executing the appropriate control algorithm at all times. This is contrasted with sequential execution in which a higher level routine calls a lower-level routine, and the higher level waits for the lower level to complete before it resumes executing. One advantage of cyclic execution is reaction time; since each level samples all inputs each cycle, the system can react to an external event, at any level, in one cycle. This could be done with interrupts in a sequential system, but it is hard to terminate the interrupt routine in a way that aborts the current routine cleanly, and even harder to predict all of the possible interactions.

For example, consider step 2 in the example task above. The Path-point level is monitoring the sonar sensors, and updating the command to the Trajectory level as often as possible. Meanwhile, the Trajectory level is controlling the fork's acceleration and velocity, to maintain smooth motion. The two levels execute simultaneously. In a sequential system, the Path-point level would have to read the

sensor, issue a command to the Trajectory level, and then wait until the command was completed before reading the sensor again.

Another advantage of hierarchical design and cyclic execution is modularity; each level has well defined interfaces to sensors and other levels. As long as the interface design is met, any level can be modified independently of the others. Also, any or all levels can be single stepped while the others are running, to help in debugging.
RSL consists of a compiler and an interpreter (see figure 1). The interpreter consists of four control levels running on three processors; a fourth processor runs the compiler and acts as a system supervisor and user interface. No external development system is needed; all programming is done on the final application system. The RSL compiler compiles RSL source code describing locations, movetables, objects and paths into a linked list representation, which is stored in common memory. The control levels then access the common memory to retrieve the data as needed.

The four control levels in the interpreter correspond to the four levels of task decomposition. TASK, PATH, and PATH-POINT execute on one processor. The trajectory level is split into two modules, CARTESIAN and JOINT, which execute on separate processors, to achieve faster cycle rates. The Joint Servo level in figure 1 is assumed to be provided by the robot manufacturer.

The input to TASK is a user command. This command is decomposed according to the task definition, which results in a sequence of paths. One path at a time is commanded to PATH; as each path is completed, a new one is commanded.

PATH accepts path commands, retrieves all the path parameters from common memory, and decomposes the path into path-points, following the path definition compiled into common memory. It commands one path-point at a time to PATH-POINT, waiting for each to complete.

PATH-POINT accepts path-point commands from PATH, retrieves the path-point parameters from common memory, and executes the path-point algorithm. The path-point algorithm typically involves reading a sensor, calculating an object pose based on the sensor data, and updating the parameters in the current trajectory command.

CARTESIAN accepts trajectory commands from PATH-POINT, and retrieves the trajectory parameters from common memory. It executes the trajectory algorithm, and outputs a pose for the robot wrist, once every control cycle. JOINT accepts the wrist pose, converts it to joint coordinates, and commands it to the joint servos.

The cycle time is 28 milli-seconds for a Unimation PUMA 760 or 4000, and 24 milli-seconds for an American MERLIN. These times are determined by the servo rates in the two robots. Note that only the trajectory level must generate new output every cycle; the upper levels are free to take as much time as necessary to process sensor data, or decide on the next task decomposition step.

RSL is user-extensible in many ways. It is designed to allow addition of new trajectories, path-points, paths, and tasks. The user adds routines to the appropriate level of the interpreter to execute the control algorithm, and also adds routines to the compiler that compile the parameters for the algorithms into common memory. Note that the compiler is very simple compared to a typical computer language compiler; the syntax and structures used in RSL are very simple, so adding to the RSL compiler is straight-forward.

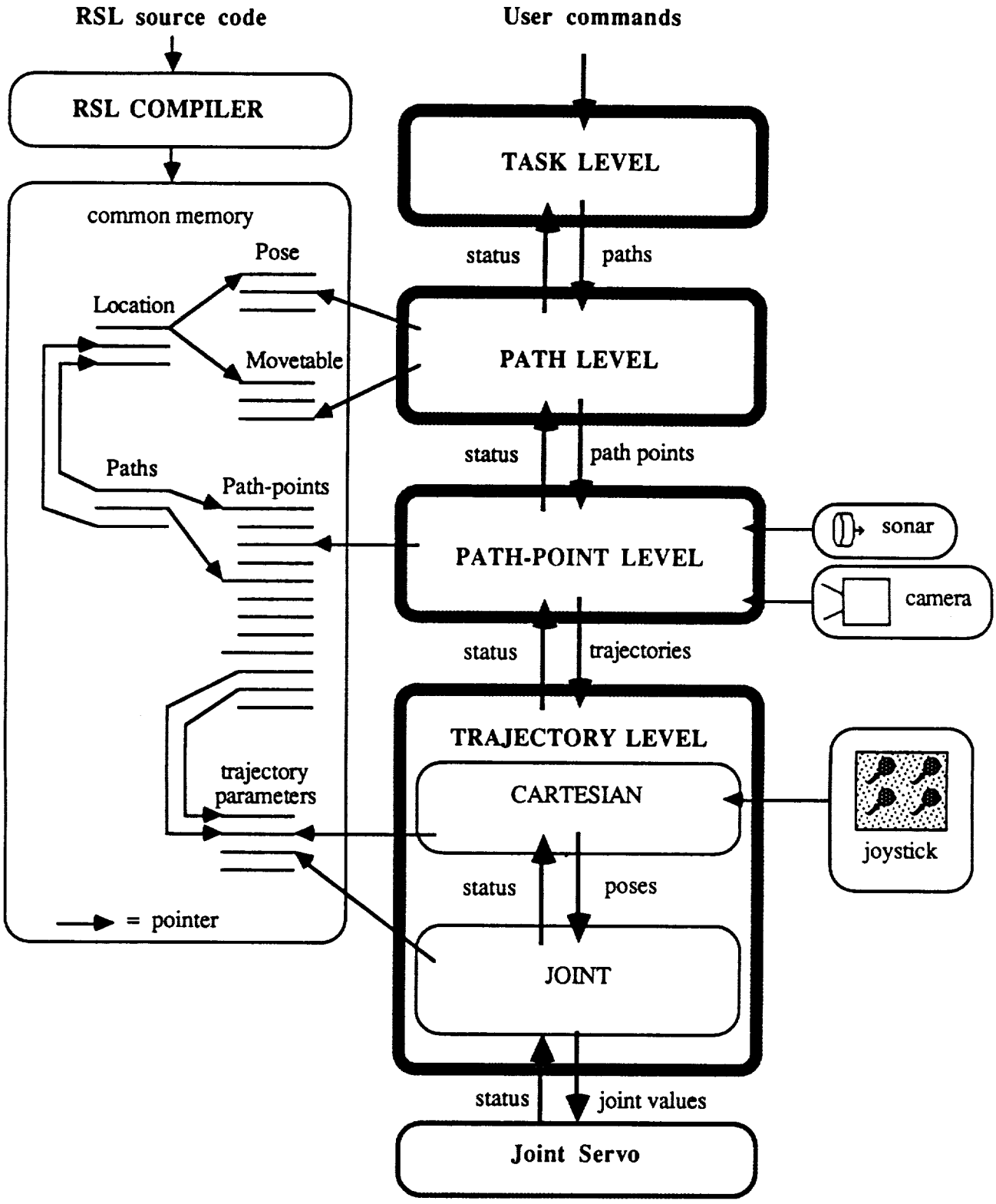


Figure 1. RSL control levels and common memory structures.

Section 3: current applications

RSL has been used successfully in three applications to date. The first is the Field Materiel Handling Robot system, which uses a fork with sensors (very much like the example above) to off-load boxes of ammunition (and other materiel) from trucks[3]. RSL was first used to program a mockup of this task on a PUMA 760, then the mockup was transferred to a UNIMATION 4000 robot. Many path-points were added to the base system, to handle all the sensors.

A second application is a cleaning and deburring workstation in the Automated Manufacturing Research Facility at NBS[4]. RSL is used to program a PUMA 760 to use air-powered deburring tools to deburr machined part edges. A separate workstation level (running on a SUN), encodes the part geometry into a path, with parameters indicating the tool to use, feed rates, etc. The path is down-loaded to the RSL system, where it is compiled and run. Path-points were added to control a vise, various deburring tools, and a tool quick-change. A deburring task and associated paths were also added. The task level accepts commands from the workstation level, and commands paths that deburr the parts, changing tools as required.

The third application is a satellite docking mockup[2]. The RSL system reads a solid-state camera, determines the position of a satellite (as indicated by four LEDs on the satellite), and drives the robot end-effector to dock with the satellite.

Section 4: future work

RSL does not support an explicit world model; all knowledge about how the world works, and in particular how sensors and end-effectors interact with objects, is implicit in the tasks and paths provided by the programmer. This makes it difficult to use more than one sensor at a time. A world model will make it possible to combine data from several sensors, by comparing the sensor readings with predictions, and servoing the model to the sensors[5].

A second area for future work is task representation. Currently, RSL uses SMACRO code to express the task level, and linear sequences of path-points to express the path level. (SMACRO is the computer programming language provided by RCS: it is similar to C, but less powerful). The path level needs to be more flexible, in particular to allow for error conditions. On the other hand, the task level should be more restrictive. The power provided by SMACRO (or C) code is deceptive; it is too easy to write code that works for a particular instance, but is not generic enough to be used for several tasks, or robust enough to work reliably. There should be a task description language that allows adequate flexibility, while guiding the programmer into writing code that works, and is generic and maintainable. The path structure provided by RSL is a first attempt at such a language.

Another reason for developing a good task description language is that it would make an excellent interface between a planning system and the control system. The deburring workstation application mentioned above has shown that this approach is worth pursuing. Having a more powerful task description language available will make it easier to incorporate task level planning for more complex tasks.

Acknowledgments

The author would like to thank Sandor Szabo and Karl Murphy for many useful ideas, and for using RSL in the applications. Tony Barbera and ML Fitzgerald designed and developed the first versions of RCS, and provided much support during the development of RSL.

References

1. The RCS Users Reference Manual. to be published.

2. Stephen Leake, " Cartesian Trajectory Algorithms for Real-time Sensor Servo ", to be published.

3. Harry G. McCain, Roger D. Kilmer, Sandor Szabo, Azizollah Abrishamian, " A Hierarchically Controlled Autonomous Robot For Heavy Payload Military Field Applications ". Intelligent Autonomous Systems, An International Conference, Amsterdam, The Netherlands, 8-11 December, 1986.

4. Harry G. McCain, Roger D. Kilmer, Karl N. Murphy, " Development of a Cleaning and Deburring Workstation for the AMRF", Proceedings of the Deburring & Surface Conditioning Conference, Sept 23-26, 1985, Chicago, Illinois.

5. Ernest W. Kent, James S. Albus, "Servoed world models as interfaces between robot control systems and sensory data", Robotica (1984) volume 2, pp 17-25.

6. A. J. Barbera, M.L. Fitzgerald, J.S. Albus, L.S. Haynes, " RCS: The NBS Real-Time Control System " , Proceedings of the Robots 8 Conference and Exposition, vol 2, pp 19-1 through 19-33, Detroit, Michigan, June, 1984.

Advanced Data Management Design For Autonomous Telerobotic Systems in Space Using Spaceborne Symbolic Processors.

Andre Goforth
Information Sciences Office
Intelligent Systems Technology Branch, Code RII
NASA Ames Research Center
Moffett Field, California 94035

ABSTRACT:
The use of computers in autonomous telerobots is reaching the point where advanced distributed processing concepts and techniques are needed to support the functioning of Space Station era telerobotic systems. This paper covers three major issues that have impact on the design of data management functions in a telerobot. It also presents a design concept that incorporates an intelligent systems manager (ISM) running on a spaceborne symbolic processor, (SSP), to address these issues.

The first issue is the support of a system-wide control architecture or control philosophy. Salient features of two candidates are presented that impose constraints on data management design. The second issue is the role of data managements in terms of system integration. This refers to providing shared or coordinated data processing and storage resources to a variety of telerobotic components such as vision, mechanical sensing, real-time coordinated multiple limb and end effector control, and planning and reasoning. The third issue is hardware that supports symbolic processing in conjunction with standard data I/O and numeric processing. A spaceborne symbolic processor, (SSP), that currently is seen to be technologically feasible and is being developed is described and used as a baseline in the design concept.

INTRODUCTION

The objective of this paper is to introduce, informally and largely by examples and comparison, an advanced design concept to data management in autonomous telerobots. The motivation for introducing advanced data management techniques in such systems is to address the system-wide complexity problem in general and the system level issues of evolvability and modularity in particular.

Data Management is a broad term. Nonetheless, it is fair to say that it is at the core of most system integration efforts. For NASA, one possible and logical approach to developing telerobotic systems is to view building them as another satellite or space craft. In this scenario, the contractor is largely responsible for system integration. As a consequence, Data Management is tucked away in the last development activity preceding the operations phase. Fortunately, there are several major efforts within NASA to elevate systems architecture and integration to an anticipatory design process[1,2].

---

[1] Albus, James A., McCain, H.G., Lumia, R., NASA/NBS Standard Reference Model For Telerobot Control System Architecture(NASREM), December 4, 1986, National Bureau Of Standards, Robot Systems Division.

CONCEPT DESIGN

In this paper, we are interested in Data Management as an integral collection of support services that satisfy the system-wide information(data) processing needs of a telerobotic system. Therefore, we recommend that these services be viewed as a subsystem that is on an equal footing with all the other subsystems in a telerobotic system.

For example, a popular view of computers is that they "think" or are the "brains" in a feature-laden appliance. In a like manner, data management in telerobots may be viewed as the nervous system as well as the brains. Given this idea, the possibility of incorporating higher levels of intelligence in a primitive information broker such as a Data Management Subsystem allows for an evolutionary approach to increasing autonomy and intelligence of telerobots.

To further this objective, we recommend that data management have its own powerful organization conventions or architectural model. An Intelligent Systems Manager, (ISM), would embody the principles of the architecture in a particular design. Among the many possible goals for an ISM, as an intelligent information (data) broker, two important ones are to reduce the complexity of interaction among multiple intelligent subsystems, and to oversee top level safety of a telerobotic system with respect to these subsystems.

Currently, there are several telerobotic system architectures and models that partition the high level telerobotic functions in different ways. We now discuss how the ISM concept fits in with these and how it aids in inserting advanced distributed processing concepts into a telerobotic control design.

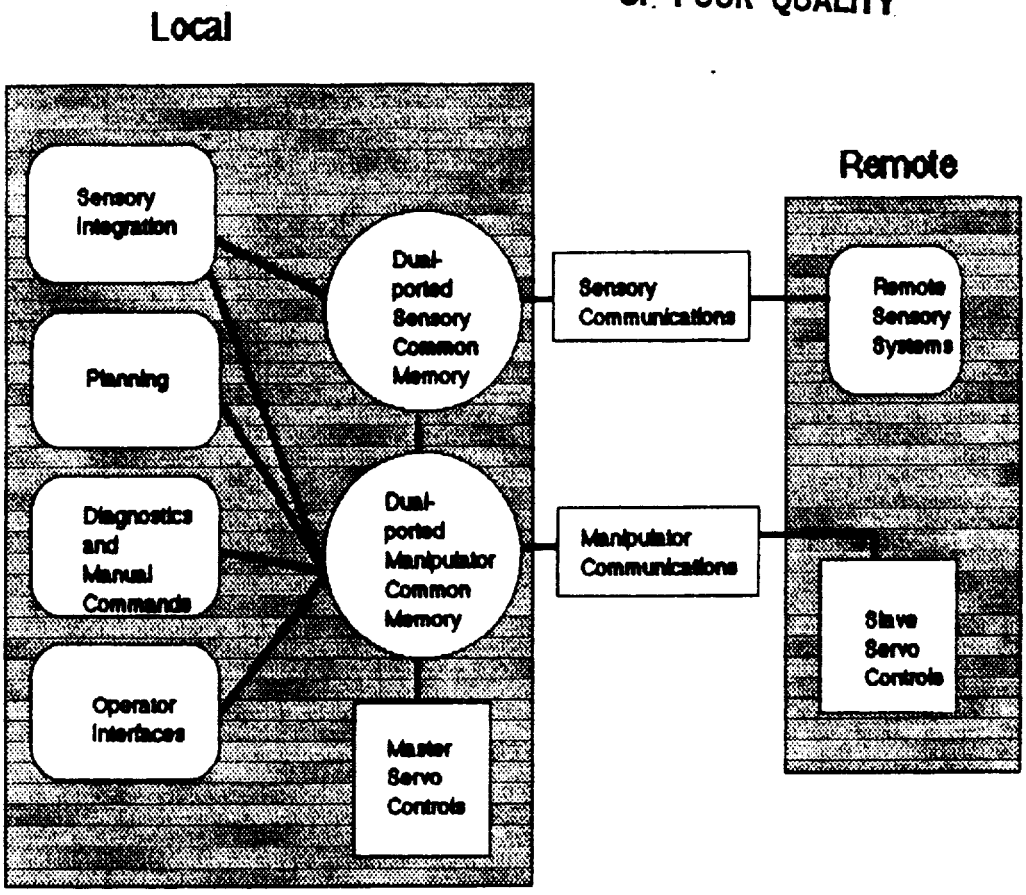
A common starting point for architecture definition of a telerobot is the specification of a controller(s). At one end of the scale we have a 'point design'. For example, requirements are immediately mapped to a specific collection of "off the shelf" components that are hardwired together to function as a controller. Of course, what is a 'point design' is a matter of degree and depends on one's systems engineering criteria.

A more general purpose approach involves defining a set of generic activities that require services of a controller(s). This is particularly difficult because the field of autonomous telerobotics is so new. As a consequence, the subject of design criteria for partitioning a telerobotic system into subsystems is evolving. One partitioning, given by Martin, et al[3], is illustrated in the conceptual layout shown below. Martin, et al, propose the use of powerful microprocessors and outline an architecture to interconnect and interface them to support teleoperator control of mechanical manipulators. In this example and in general, separating out what is integral to a Data Management Subsystem from data processing elements indigeous to other subsystems in a telerobot is no easy task.. For example, in Martin, et al's, design there are data management functions in all activity areas.

---

[2]Functional Requirements For The 1988 Telerobotic Testbed, JPL D-3693, October 1986.

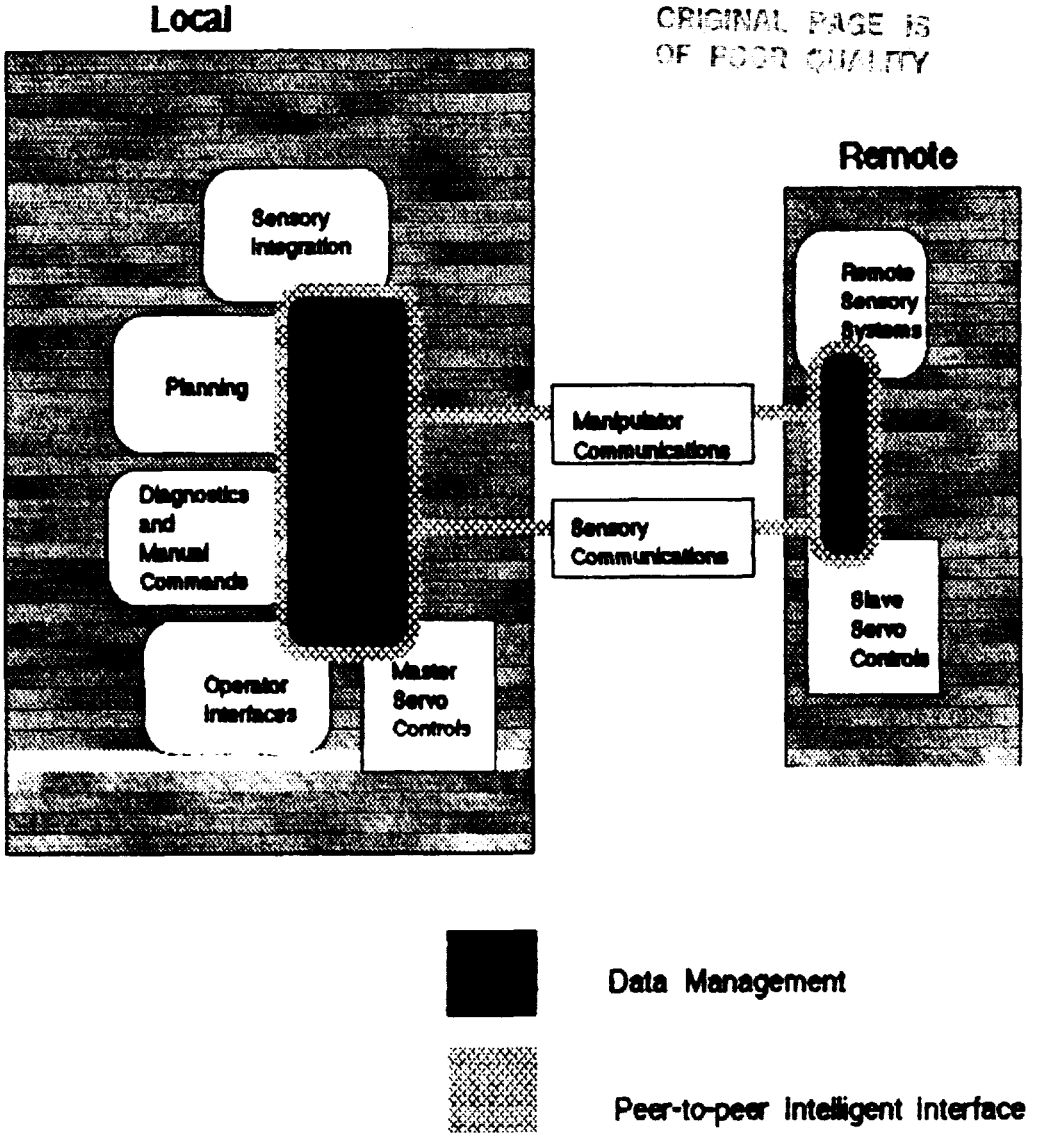
[3]Martin, Lee H., Paul E.S. Salterlee Jr., and Richard F. Spille, "Distributed Control Architecture For Real-Time Telerobotic Operation", JPL Space Telerobotics Workshop, January 20-22, 1987 (in publication).



Conceptual layout of the principal centers for telerobotic control(Martin et al)

Figure 1

What we propose in this example is that another activity area - data management - be added to the top level list of major control activities as shown in the following figure:



Example of Data Management As A Principal Center For Control.

Figure 2

Since, we do not know the implementation cost, the above conceptual layout is not yet recommended as a better way to solve the specific class of telerobotic functions that Martin, et al, are addressing. We use it to illustrate an important shift in design. Each subsystem has its own information processing needs and, therefore, will have internal data management facilities. The crux of the matter is figuring out what the interfaces are between the Data Management subsystem and the other subsystems.

INTELLIGENT INTERFACE

On one hand, the layout proposed above may seem to be equivalent to the systems integration design already derivable from the existing architecture as proposed by Martin, et al. On the other hand, suppose a uniform and common interface is available

that any subsystem must use in order to coordinate end-to-end activities with other subsystems in the telerobot. This interface is between a subsystem and the Data Management subsystem. This is represented in the above figure by the overlap of the Data Management subsystem over the others and the stripped band.

Let's assume such an arrangement for the moment. In order to add another subsystem, this subsystem must be built to work with that interface. In the case that the new subsystem can obtain all nonindigenous resources and end-to-end services only from the data management subsystem, we then say that system evolvability or scalability in linear in the architecture with respect to this interface.

The issue now facing us is whether such an interface is definable in telerobotic systems and, if so, what will it take for it to be effective? Another way to look at the question is to ask what is the best way to organize the Data Management subsystem so that its impact on subsystem dependencies is minimized? The more flexibility the Data Subsystem has the better.

Existing interfaces occurring in different levels of computer technology reflect a form of architectural linearity or scalability for a variety of modules. For example, at the hardware level, standard buses such as the VME and Multibus support a number of controllers up to a standard configuration limit. At the local area network level, Ethernet supports the linear insertion of nodes up to a configuration-dependent maximum.

At the operating systems level, there are distributed operating systems running on multiprocessor configurations that allow, up to a limit, processes or jobs which have no interdependencies to be assigned to available processors at a fixed per process/job overhead. A commercial operating system exhibiting such a capability is Dynix[1]. It is a proprietary design and runs on a multiprocessor system, the Balance 8000, which supports a configuration of up to 24 32-bit microprocessors. In fact, performance measurements show an efficient linear performance curve for jobs with certain types of dependencies.

The point we want to emphasize is that was only a few years ago it seemed impractical to even try to organize[2] large scale systems such as computers to operate in parallel on a collection of jobs and processes that were reasonably independent and maintain a linear performance curve across a useful work range. Today this is becoming routine. Likewise, with something as complex as a telerobot , we can make headway in achieving efficient linearity for a practical set of parallel and independent tasks. Therefore, we wee instances of linearity in performance and scalability in a variety of computer technology levels.

OBJECTS

The success of achieving this sort of linearity and modularity of performance from the Data Management subsystem in a telerobotic system depends initially on our ability to

---

[1]The Balance 8000 Technical Reference Manual, Sequent Computer Systems, Inc., Beaverton Oregon.

[2]Amdahl's Law circa 1970: For the same amount of money one big computer will provide more throughput that a collection of smaller ones.

represent the concept of an intelligent interface in software. The representation approach we recommend is through the use of abstract objects.

The use of abstract objects is one of several key concepts in advanced distributed processing[1]. Many programming languages support objects either directly by syntactic conventions or indirectly by the programmer's use of an object-oriented methodology. For example, work is being done in object-oriented design using Ada[2,3]. Strongly related to the object oriented design approach is the concept of layered design. The layers of an onion are often used as an analogy for the layering of objects in the whole design.

Is is beyond the scope and purpose of this paper to do more than introduce the salient aspects of this subject. We are interested in provoking thought along these lines in development people working in a variety of telerobotic disciplines. In particular, those R & D engineers who must deal with the system engineering problems of data management or are greatly affected by it s presence or absence.

The sort of telerobotic systems we are anticipating will have requirements that are at least as computationally complex and challenging as those envisioned in DARPA's research[4] in autonomous vehicles. NASA's Flight Telerobotic Servicer(FTS) will easily have the same complexity in data management and computational requirements as these systems are envisioned to have if it is to support a major autonomous mode of operation for space stations maintainence.

Object oriented design methodology and programming at the most general or abstract level spans at least two different data/information management disciplines. Whereas, computer science is often associated with procedural languages such as Ada and with numerical algorithms, Artificial Intelligence(AI), is associated with functional languages such as Lisp and with symbolic processing. These differences are mentioned to illustrate that, in fact, this concept and methodology is an integral part of each of these two disciplines(or styles) and is a strong common point between the two. This commonality in using objects is often lost because of the pigeon-holing of professionals as being either in AI or in computer science. Therefore, we prefer to keep the discussion of objects independent of any language (which is a form of implementation of these concepts) or discipline.

---

[1]Lampson, B.W., Paul, M., and Siegert, H.J., Distributed Systems-Architecture and Implementation, An Advanced Course, 1981, Springer-Verlag, 15-16.

[2]Firesmith, Donald G., Object-Oriented Development, Proceeding: First International Conference on Ada Programming Language Application For the NASA Space Station, June 2-5, 1986, High Technologies Laboratories, University of Houston-Clear Lake, Texas, D.4.1.1., D..4.1.11.

[3]Booch, Grady, Software Engineering With Ada., The Benjamin/Cummings Publishing Company, Inc. 1983.

[4]Torrero, Edward A., Military R&D, The DARPA Program['Strategic Computing', DARPA October 1983], Next-Generation Computers, IEEE Press, 1985, 153-154.

Another reason to keep the discussion at a generic level is that, except for small subsystem controllers in telerobots or scientific payload instrument control applications, current implementation of programming languages supporting these constructs put major constraints on object representation, manipulation, and performance in a distributed environment. However, significant strides are already being made. Martin, et al, note that the performance capability of the Novix[1] microprocessor is largely due to its direct support of the Forth language.. A relatively new language called Neon[2], which borrows heavily features from Forth[3] and Smalltalk[4] supports objects in its syntax. Consequently, it is an example of a product that straddles the fence between AI and computer science. Such innovations are likely to continue and even accelerate in the future. Oak Ridge National Laboratory, in a report on the Man-Equivalent TeleRobot, mentions that its experience in using Forth as a language for real-time control has been positive[5]. Therefore, it is plausible that an object design environment that meets significant telerobotic requirements in data management could be developed in the near future. Whether it is a 'new' language, such as Neon, or a sophisticated development environment built on top of existing software technology, such as Ada or one of the varieties of Lisp, remains an open question. The key issue in either case is the availability of an appropriate computer architecture(s) and technology that support such a design approach and, at the same time, will embed successfully in telerobots.

We are encouraged to pursue this approach to data management given that there are more and more programming implementations of these concepts available on powerful microprocessors.

Abstract objects(or resources), both active and passive, are such things as files, directories, processes, tasks, virtual I/O devices, databases, and any other item that is useful for the designer to identify as part of the system at a certain level. In contrast, real objects are such things as processors, secondary storage, controllers and any physical item that must be taken into account at a certain level (layer) of detail to perform a function. (The irony is that the use of abstract objects in designing data management architecture is making abstract things appear as "hardware" entities to be software/systems engineer and real things such as scientific instruments and sensors appear as "software" entities.) Each object is specified by its representation and a set of operations or functions and associated parameters that can be performed on the representation. The implementation details of an object representation are contained in an object manager. (Note that an object manager may itself be a object in the system.) In a distributed environment, message passing is needed to exchange information

---

[1]See footnote 3.

[2]The Neon Manual, Kriya System, Inc. 505 N. Lakeshore Drive, Suite 5510 Chicago, Ill. 60611. Runs on the Macintosh computer.

[3]Winfield, A. The Complete Forth, A New Way To Program Microcomputers, Wiley and Sons, 1983.

[4]Goldberg, A., Smalltalk-80: The Language and Its Implementation, Addison-Wesley, 1983.

[5]Recommendation For The Next-Generation Space Telerobot System, Oak Ridge National Laboratory, TM-9951, March 1986.

between object managers and to carry out operations on objects. The mechanism of message passing may be implemented using shared memory and procedure calls or may be implemented by send/receive operations that require a protocol to insure reliability and fault tolerance.

We now describe the conceptual elements of what we mean for an interface to be intelligent, and, correspondingly, refer to an intelligent Data Management subsystem as one that uses such an interface. We use the terminology of objects discussed up to this point in order to establish specifically the properties of this interface.

An interface is defined as a set of conventions for the exchange of information between two object managers. The three components of an interface are:

- A set of visible data objects or modules and the allowed operations and parameters associated with each visible data object or module.

- A set of rules governing the logical or legal sequences of these operations.

- The encoding and formatting conventions required for operations and data.

We say a peer-to-peer intelligent interface exists between the Data Management subsystem and the other subsystems in a telerobot when the following properties hold: First, it is possible for the Data Management subsystem to actually pass or export copies of code that meet the specification of the three components of the interface specification to a subsystem. A extreme example of control is that the Data Management subsystem would have to pass or export all of the actual code needed in a subsystem for that subsystem to be able to use Data Management subsystem services. As an example of negotiated control, a subsystem could likewise pass or export select components of the interface back to the Data Management subsystem for purposes of adaptive configuration of services. Second, the interface itself must be symmetric. By this we mean that two peers use the same specification (with differences limited to addresses and local house-keeping functions) in order to interact. In some computation settings, such as data communications, such an interface would be viewed as a protocol. In this sense, what we see being passed two subsystems in an autonomous telerobot is the protocol itself tailored to allow for special "hand-eye" coordination or the control of dynamic chaining of control loops. However, we see even more complex information being passed in such a fashion.

For example, in DARPA's autonomous vehicles program it is envisioned that these vehicles are characterized by their ability to accept high-level task description[10]. In a like manner, an intelligent data management subsystem would have to be able to take a template of information given to it by a higher level task synthesizer.. Consequently, it would pass templates of information, (in AI parlance, knowledge base facts and rules) to the subsystems in order to set up the coordination of data processing functions within the telerobot.

The Data Management subsystem could support an even more adaptive mode of interaction among subsystems if the overall telerobotic control design allowed a top level task synthesizer to pass information templates directly to subsystems, (A low level Data

Management subsystem service of pass information template to X, Y, and Z would be used and available on a reflex basis to a high level task synthesizer.). The subsystems, in turn, would synthesize their information needs based on what was requested of them and would then pass their information templates up to the Data Management subsystem.

The object oriented, peer-to-peer, intelligent interface envisioned as the Data Management subsystem boundary discussed in this paper, has a built-in conceptual adaptability to integrate the following two autonomous telerobotic data processing requirements : to support what appears to be the top down flow of data, i.e., the cognitive and more offline type of activities of planning and reasoning; and, to support the bottom up or reactive and more real-time activities such as run-time control of physical processes and processing of sensory information.

For example, some telerobotic operator controlled operations may require on demand a large portion of the data management subsystem's resources to handle real-time interrupts and to process a large quantity of data, for example, integrating multi-sensor data). For a given cost/performance profile a fixed or nonadaptive data management subsystem design may be easily overloaded by real-time operations. Similarly, a static design may be overwhelmed by large amounts of planning and reasoning due to critical and abrupt changes in task objectives.

INTELLIGENT AGENTS

From the point of view of a subsystem, what are the bare minimum or necessary and sufficient conditions to support the peer-to-peer intelligent interface concept? In order to answer this question, we introduce the concept of the Intelligent Agent that, by our definition, resides in each subsystem of an autonomous telerobot. The Intelligent Agent has potentially several roles within the context of of a subsystem. These roles may be determined by an external knowledge source. In a paper by Sztipanovits, the Multigraph Architecture (MA) is a four layer architecture for intelligent systems that provides "knowledge-level" information for Autonomous Communication Objects (ACO) in its Knowledge Base Layer[1]. For the purposes this paper we focus our concept of an Intelligent Agent as a small compact information broker[2] that is responsible for managing the interface of its host subsystem with respect to the rest of the system. It has to insure the correct use by its host subsystem of the intelligent interface.

All Intelligent Agents in the system adhere to the same intelligent interface in a fair manner. Each IA has the capability to actually pass or export an object from its subsystem to another one and have the Intelligent Agent in the receiving subsystem accept it upon demand, i.e. within a 'reasonable' time frame. The capability for an IA to accept any object upon demand may be impractical. What is more practical and

---

[1] Sztipanovits, J., Execution Environment For Intelligent Real-Time Control Systems, JPL Space Telerobotics Workshop, January 20-22, 1987 (in publication).

[2] An anonymously authored NASA SAIS document uses the term Intelligent Agent as the "eyes and ears" in remote space platforms that take commands and sends information to a master controller located in the Space Station or in a larger system. All instruments and platforms are designed in such a way as to be able to host an Intelligent Agent.

addresses the intent of the conceptual design is that each IA can be unilaterally signaled and required to, at the very minimum, take a "command " object.

The last capability that an Intelligent Agent must have in its role as interface manager is the ability to reset(replace) another Intelligent Agent. Equivalently, any Intelligent Agent automatically accepts any "command" object, and that "command" object may be "replace yourself with me".

INTELLIGENT SYSTEMS MANAGER

Obviously some higher level management functions is required in order not to have Intelligent Agents resetting one another in a hazardous manner. The Intelligent Systems Manager (ISM) is , by our definition, the designated Intelligent Agent that has the authority to give and revoke all other IA's capability to reset peers. Furthermore, it has authority to give and take other resource privileges of IAs. It is now readily possible to design such an organizational scheme to be logical secure through the use of object capabilities [1]. The actual design of incorporating these into a Data Management subsystem still has to be done. What makes the designer's job much easier is that, if properly used, capabilities can insure the logical soundness of an executive resource controller such as an IA/ISM in real time.

The proper use of object capabilities assumes that a logically sound theory and specification of access and control between the ISM and IAs has been developed. As a simple example, only legal sequences of reset capability are ever granted.. What are the rules used by the ISM and conditions maintained by it among all the IAs so as to enforce correctness? The development of a logically sound cooperation mechanism in which only legal sequences are possible and illegal ones created by external corruption of data are contained, is a major area of research in advanced distributed processing. In elementary and not so elementary cases mature theoretical results are available. What yet needs to be done is to investigate the technology and implementation aspects. An autonomous telerobotic system hosting Intelligent Agents is an ideal testbed for capability-based architecture design.

The interface manager function is only one of several for an IA/ISM in a telerobotic system. It happens to be a minimum and the cornerstone of the design concept. The generic role for IA/ISMs is to serve as accretion points. These points are viewed being within the Data Management subsystem and allow for the insertion of more and more "smarts" or intelligence in the whole design of an autonomous telerobot. Note, this is specifically directed to information flow between subsystems. A telerobotic system will gain 'smarts' from advances in sensor and reasoning technologies. In addition, the IA/ISM will allow a telerobotics system to get "smart" from integration of subsystems. The IA is, itself, a place to insert improved reasoning and learning technology. However, for an initial implementation, only the interface manager portion may be done.

Later on, as experience is gained with this approach, more functionality and robustness can be added. The result is that a sequence of IAs and ISMs may be built, each one more advanced that its predecessor and serving more and more autonomous

---

[1] Lampson, B.W., Paul, M., and Siegert, H.J., Distributed Systems-Architecture and Implementation, An Advanced Course, 1981, Springer-Verlag, 202, 235-245.

telerobots.. A demonstration goal of the intelligent interface could be that different generations of IAs may coexist in real-time in a system. Furthermore, insertion of a next generation IA could be made in real-time without having to disable the telerobotic system for any major length of time.

The Intelligent Systems Manager design concept is not meant to be an alternative to a system-wide control architecture or control philosophy. Specifically, we see it providing a conceptual bridge for mapping between the data processing resource space and the overall functional space as described in a model such as the NASA/NBS NASREM model[1] and the target subsystems in a telerobotic system. This approach of introducing another concept design model such as the IA/ISM is an attempt to bridge the concept hierarchy problem discussed by Wolf, et al[2]. The problem is what are the appropriate levels in which to decompose a problem such as building an intelligent supervisory control system. The four given are the functional, resource, knowledge and computer architecture. Each of these may have their own model.

The NASREM model gives an all inclusive functional system model for a telerobot. It is a six layer hierarchical model from top to bottom and has three horizontal partitions for sensory processing, world modeling and task decomposition. In our view, one of the purposes of this model is to be the framework for developing a system effectiveness criteria to be used to evaluate proposed designs. For a specific telerobotic system, this is accomplished through iterations of tradeoff analysis of mission objectives (requirements) and constraints. The logical distribution of functions in the NASREM model is to provide a gauge for a particular design's effectiveness. On the other hand, the distribution of a logical function in an implementation is subject to another effectiveness model that incorporates constraints of the resource function.

For example, Hawker, et al, of Lehigh University, in a paper[3] on multiple robotic manipulators, limited their interpretation of the NBS approach to dual arm control as requiring a triad of controllers. One controller for each arm and a third to control these two. With the ISM design approach to data management, a design goal would be to have the ISM dynamically hand off to two Intelligent Agents(assuming each arm is in a different subsystem and hosts an IA) so that each one could directly communicate in the cooperation of the two arms without the ISM in the loop. For example, ISM send to IA right arm a reset object Q and then tells IA left arm to accept from IA right arm an object that will cause reset of itself.

The reason for all of this is that the dynamic control algorithm is likely to be considerably different from single arm control. Therefore, we have to replace both single arm IAs (or those portions critically related to run-time control) with a version of an IA that effectively does the dual arm operations. The ISM has to fashion out of a higher order information template (calling for dual arm control), all the contextual information about the task.

---

[1]See footnote 1

[2]Wolfe, William J., Raney, Steven D., Distributed Intelligence For Supervisory Control, JPL Space Telerobotics Workshop, January 20-22, 1987,(in publication).

[3]Hawker, Scott J., Nagel, R.N., Robers, Richard, and Odrey, Nicholas G, Multiple Robotic Manipulators, Byte Magazine, January 1986, 203-219.

Note, this most likely will not be doable with only dynamic swapping of to an alternate program due to configuration and linkages overhead. The object Q may contain several other objects in it that pertain to, for example, real-time collision avoidance and other high level world model information condensed down to be appropriate at this level. Consequently, the logical hierarchy of function of the NASREM model is preserved but the real-time flow of information would go according to the criteria of a different model, a system effectiveness data resource model for autonomous telerobots.

An even more interesting problem than that of dual arm control is the change out of an end effector by a robot. It is highly likely that the data management in the telerobot would have to dynamically reconfigure itself to accommodate such changes. Consider, the more extreme case, where a robot has to take itself apart to fit through an aperture or repair itself by swap out.. The run-time requirements on Data Management adaptability will indeed be challenging.

Realizations of the flexible data processing example have not been tried for robotics yet because of the lack of a suitable computer architecture and technology that support the rea-time object-oriented processing described here for telerobots.. However, this is rapidly changing and may already be attainable in some ways. Therefore, with respect to Hawker's conclusion, we feel that the NASREM model is indeed relevant to dual arm control, but that it should not be used as the final system effectiveness model for data flow and processing in a telerobot.

Anyone developing a large scale telerobotic system may wish to partition the data management subsystem in a one-to-one fashion according to the logical hierarchy in a model such as the NASREM as a first cut to understanding functional and logical relationships. But, as these are understood and clearly identified, another model that is responsible for data management resources should be used to refine the design and the ultimate realization of the Data Management subsystem.

For the reasons present above, we recommend that system engineers working in telerobotics carefully look at how the models are used. Some of the debate of the applicability of using a global framework such as the NASREM is due, in our opinion, to trying to use one model to solve a problem that actually needs four separate ones.

Currently, there is a need to develop a resource model for data management in the context of autonomous telerobots. This model could then be used to gauge the effectiveness of proposed designs for data management. The IA/ISM would be one of them when it is sufficiently developed. For now, our criteria for effectiveness is limited to nominal data management functionality, adaptability, and dependability.

SPACE BORNE SYMBOLIC PROCESSOR

At NASA Ames Research Center, the purpose of the Space Borne Symbolic Processor project is to advance the application of revolutionary computer architectures that combine both numeric and symbolic processing for space and aeronautical flight.

Currently in the AI research community, a great deal of experimentation and prototyping of architectures and technology is underway which is specifically aimed at improving the performance of AI-based systems. For a recent survey, see the January 1987 issue of

Computer. The goal of all this research within roughly the next five years is to improve the performance of symbolic processing applications by at least two to three orders of magnitude.over what can be done today. One of the application areas to benefit from this performance improvement is expert systems and expert system building tools.

Significant advances in computer technology and architecture are needed to support the IA/ISM design concept. in space borne autonomous telerobots. This is true particularly in large scale distributed environments where objects must persist for long periods of time, span physically over disjoint memories, and have real-time interaction with other subsystems composed of sensors, dynamic control loops, human operators, and planning databases.

Two critical requirements for supporting an Intelligent Agent are the efficient representation of objects and the high performance run-time support of object handling. We see the dynamic research activity in the AI discipline as being the most promising long term source of technology to support Intelligent Agents. Our two critical requirements may be stated in the following manner: What software and hardware elements must exist in a subsystem in order to host an Intelligent Agent?

In software, the ability to represent object abstraction is required. This is greatly

In hardware, there must be an efficient architecture to support the movement and run-time behavior of active objects. Note, the implicit condition that this be supported transparently in either a loosely or tightly coupled environment of multiple processors This is a vast subject area and, since the sixties, has often been referred to as the semantic gap between the software and hardware. Research is underway wherein new hardware units are being proposed and tried out to support more directly the movement of variably structured objects in a distributed environment.

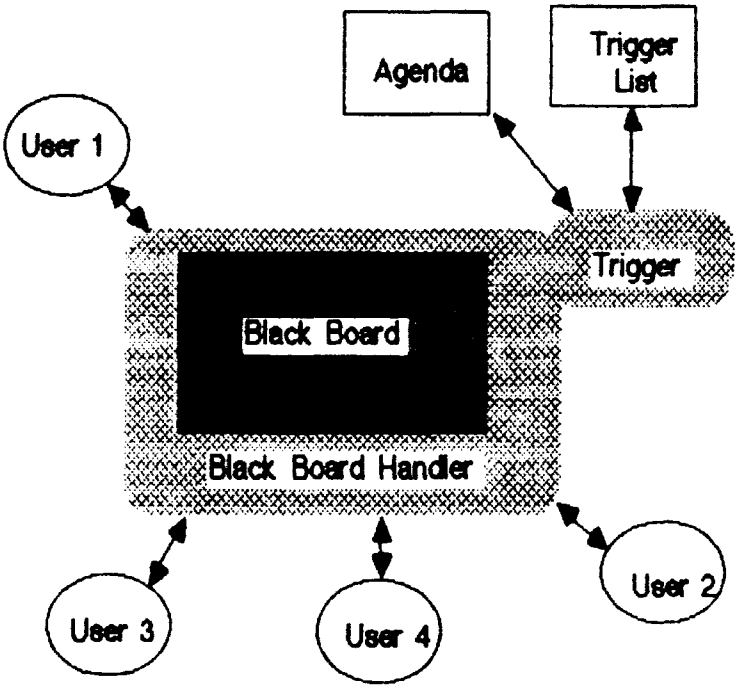
Another crucial hardware element is the specific ability of one subsystem to fork an object into another subsystem. We see rudimentary parallels to this in some programmable interfaces in today's microprocessor-based controllers and in large mainframe computer systems. The IBM 370 series mainframe software would assemble a channel program, send it down the I/O channel to the channel controller, and then hand over control to the controller by a command sequence that said: "execute this program".

The details of realizing this crucial capability of object forking may be implemented in a variety of ways. In terms of a telerobotic system, for example, do we design and build a special hardware backplane that runs through all the subsystems, or glue together existing hardware components, or use a local area network? The most relevant approach to the specifics of doing this is to include this as a systems engineering requirement for subsystems in an autonomous telerobot. Only in the context of a specific set of telerobotic missions requirements can such trade-offs be usefully done. One candidate is the technology being proposed for a Space Borne Symbolic Processor, (SSP).

Integral to the ability to fork an object onto another subsystem is the hardware that allows the resetting of that part of a subsystem.(internal to the Intelligent Agent) that is running a

forked object. As an example, a mainframe would be able to unilaterally reset a controller regardless of what it is doing.

The induced run-time design requirement to support IAs is nontrivial. A sophisticated trigger mechanism that uses a dynamically prioritizable vectored interrupt scheme is one possibility. Work is being proposed for designing and building off-the-shelf hardware components to support the triggering mechanism needed at the hardware level to support the Blackboard model. Simply put, the model is an AI paradigm of individuals communicating by free association by writing on a blackboard their knowledge of a problem(or task) and reading from it as they please. At the nitty gritty hardware level, requirements are much more constrained and strict if serious real-time applications are to be supported. The following figure is one illustration of an architecture of a blackboard at the subsystem/component level.



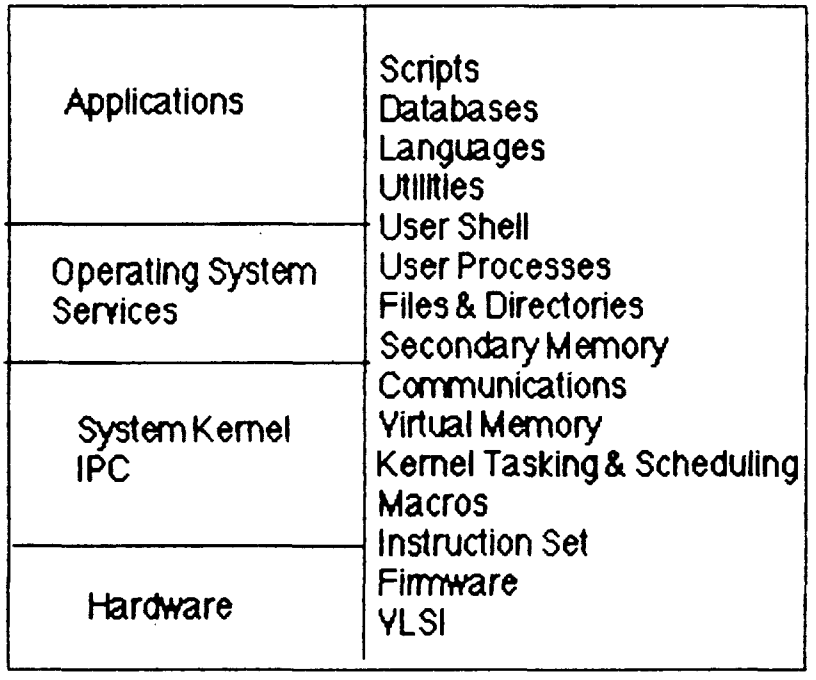
Block Diagram of Black Board Model

Figure 3

The hardware trigger mechanism for real-time black boards may be used in a design to support the Intelligent Agents embedded in a telerobot within the next five years. However, without such devices available it is a crucial tradeoff, depending on certain

telerobotic constraints, whether to even design the Data Management subsystem as hosting an Intelligent Agent. There is a compromise case that the Data Management can afford one and another subsystem cannot. In this instance, this subsystem could multiplex its access to the Data Management subsystem through a shared IA hosted on another subsystem to remain true to the design concept and meet cost constraints.

The current research thrust in architectures and technology for AI applications questions the very basic tenets of computer system design. For example, the boundary between what is software and hardware and the usual conventions of layering an architecture are being re-explored. The following figure is an example of logical layers (or levels) that are used to organize and understand various functions in an computer design.



Computer Architecture Hierarchy Model
Figure 4

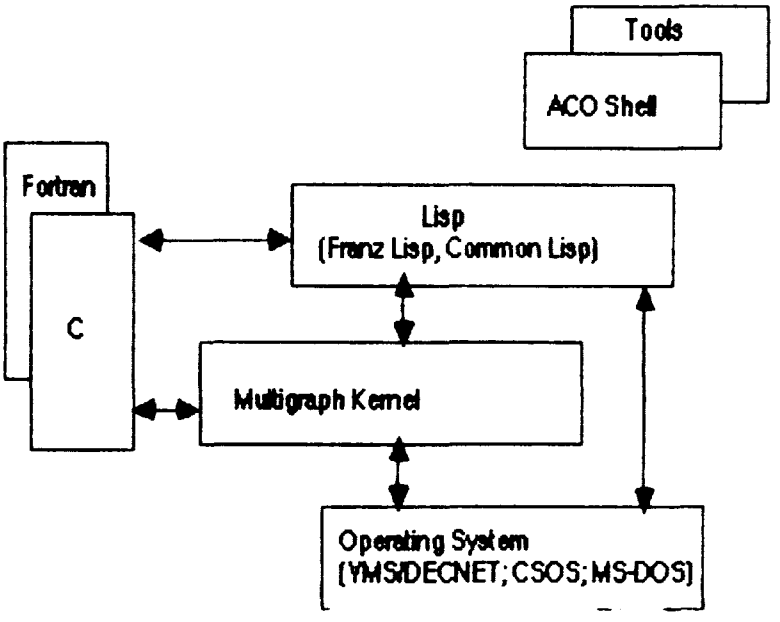
Note that we use this hierarchy only as a model and not as a representation of the design of an architecture . The use of a hierarchy is a powerful and commonly used tool to aid in the understanding of a design. For a particular system design, the layering will be unique to that design.

It is this variability in layer definition that makes evaluating and understanding the impact of innovations in AI architecture and technology difficult to gauge. The only recourse,in many cases, is to actually build prototypes and run experiments. In contrast, the hierarchical layering in numerically oriented architectures is relatively more stable. Increments in performance and functionality are easier to gauge when adding a function in a particular layer or by speeding an existing one up.

A major shortcoming of this model is that the structure or microarchitecture of the execution environment is not obvious. The reason for this is that the execution environment in conventional designs depends on functions at several different layers. Consequently, the structure or microarchitecture of the execution environment is not optimal either in performance or in representation (programming) of real-time object handling.

A major reason for the relatively low performance levels of today's symbolic processors is that they are based on an incremental design approach of hosting AI software on conventional, numerical processing oriented architectures which in turn often suffer from a weak run-time microarchitecture.

An example of what we call a microarchitecture model of the run-time environment is given by Sztipanovits of Vanderbilt University in the following figure. He calls it the structure of the execution environment.



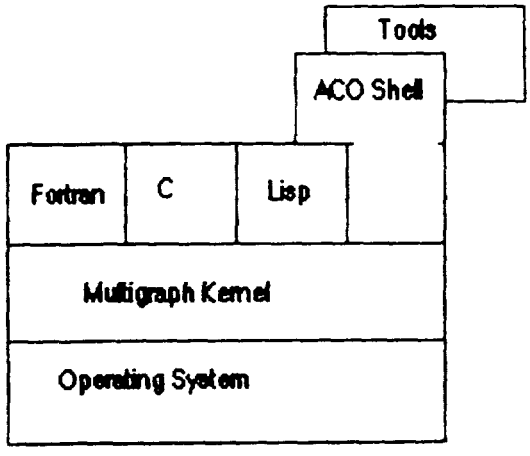
Structure of the Execution Environment(Sztipanovits)

Figure 5

The focus of his research is to use the Multigraph Architecture(MA) to study intelligent systems operating in a real-time, parallel computing environment.. Since, it is a requirement to host this work on a variety of computers he has to include a number of interfaces for facilitating portability and flexibility. We view the following as primarily interfaces of the Multigraph Kernel(MK) being there for these roles: Fortran, C and the Operating System interface.

Let us consider what would happen to the components of this model if we were trying to design a "lean and mean" run-time environment . One step toward this goal would be to minimize the interfaces that the (MK) has. It is interesting to note that many applications using objects and symbolic processing are in runtime environments that are as complicated as this one. This is one example of why existing object based applications do not perform favorably with their numerical counterparts.

Our preliminary design of a very compact microarchitecture for the (MA) is given below.



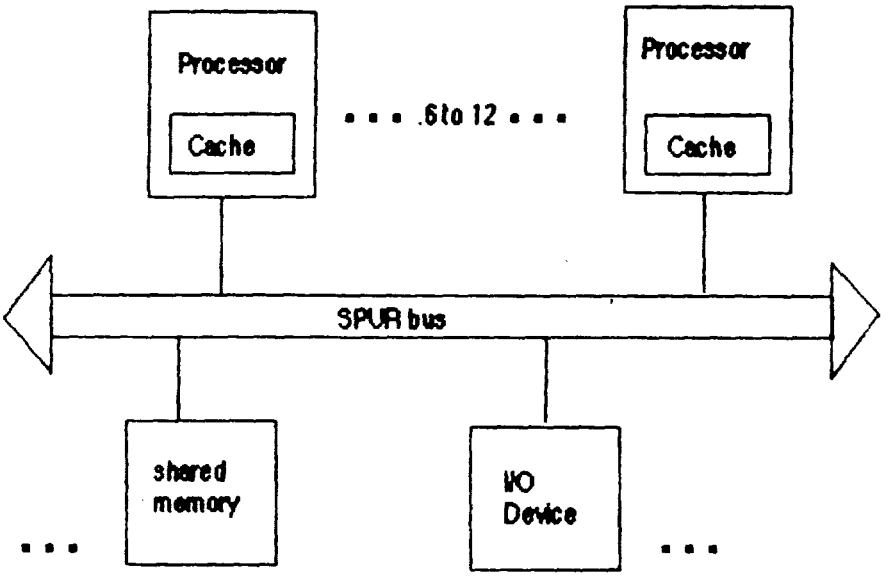
Example of Compact Execution Environment

Figure 6

The interfaces are fewer and simplified. The upper interface of the kernel supports all of the languages equally. The operating system(run-time aspects) is pushed up to where the kernel is. A single object oriented run-time resource manager is part of the kernel. By design, the kernel and hardware interface may be greatly simplified. The (ACO) shell may have direct support in the kernel. This is an optional design tradeoff and depends on how important it is to "hardwire" knowledge base facts directly into the kernel.

The last major architectural issue that remains is how to efficiently mix and match the parallel execution of programs using numeric and symbolic data. The problem is that procedural languages such as C and Fortran and functional languages such as Lisp have some major differences in terms of efficient, high performance data representation and processing.

The SPUR(Symbolic Processing Using RISCs) is a multiprocessor design that does addresses this issue.



Block Diagram of Berkeley SPUR

Figure 7

The design consists of identical processor modules/boards each of which support both symbolic and numeric operations. This commonality is achieved on the board by having special purpose processors that process floating point, and list(symbolic) data separately. An elaborate onboard caching scheme is used to move data to and from the coprocessors and the global memory and to identify the data as to whether what type it

is. Up to 12 processors, a global virtual memory of 256G-bytes , and I/O devices are all connected together by a common bus. The SPURbus is 64 bits wide is based on a modified Texas Instruments NuBus.

Research for the SSP will undoubtably look hard at the features of this design. The tradeoffs of whether to use a bus or network and whether to hid special purpose processors in a module that connects to the bus or network or to hang them directly off the connection media will be a interesting computer engineering trade. One design tradeoff used in the SPUR that is relevant to the flight environment is the size of the onboard caches. In a flight environment, the requirement to power a wide backplane is up against a major power constraint. Since the SPUR is designed to used as a low cost workstation the SPURbus is slow when compared to similar designs using multiprocessors that are aimed at replacing large uniprocessors. The solution was to put in relatively large caches for both instruction and data.

There is one interesting requirement that the designers of the SSP should consider that does not seem to be possible with the SPUR. The development of computer chips and modules that support higher level abstraction above just tagging data is important. Hardware support for Black board functions may soon become a reality. In the future, special purpose hardware for such things as Intelligent Agents and cooperating expert systems may be desirable to support at the computer component and architecture level.

As a consequence, the scope of the SSP in terms of supporting AI technology in flight has the important role of being a pathfinder in how such developments could be configured into a flight system. At one end, we have a stand alone Lisp processor in space running an embedded expert system, and at the other, we have the possibility of multiple blackboards whose knowledge sources are able to share information across disjoint domains.

ACKNOWLEDGEMENTS

I would like to thank the participants in GSFC's Flight Telerobotic Servicer Skunk Works of 1986 for a most simulating and challenging time where I got inspiration for these concepts.

LW 0864

ROBOT HANDS AND EXTRAVEHICULAR ACTIVITY

BY

BETH MARCUS, PH.D.

ARTHUR D. LITTLE,. INC.

MAY 14, 1987

N89-10097

INTRODUCTION

Extravehicular activity (EVA) is crucial to the success of both current and future space operations. As space operations have evolved in complexity so has the demand placed on the EVA crewman. In addition, some NASA requirements for human capabilities at remote or hazardous sites have been identified. One of the keys to performing useful EVA tasks is the ability to manipulate objects accurately, quickly and without early or excessive fatigue. The current suit employs a glove which enables the crewman to perform grasping tasks, use tools, turn switches, and perform other tasks for short periods of time. However, the glove's bulk and resistance to motion ultimately causes fatigue.

Due to this limitation it may not be possible to meet the productivity requirements that will be placed on the EVA crewman of the future with the current or developmental Extravehicular Mobility Unit (EMU) hardware. In addition, this hardware will not meet the requirements for remote or hazardous operations.

In an effort to develop new ways for improving crew productivity, NASA's Johnson Space Center awarded a contract to Arthur D. Little, Inc., to develop a prototype anthropomorphic robotic hand (ARH) for use with an extravehicular space suit (contract #NAS9-17454). The first step in this program was to perform a design study which investigated the basic technology required for the development of an ARH to enhance crew performance and productivity. This paper summarizes the design study phase of the contract and some additional development work which has been conducted at Arthur D. Little, Inc., after the conclusion of the Phase I effort.*

OBJECTIVE

The study had three major objectives. They were:

- To characterize the EVA environment and develop the operational requirements placed on the gloved hand which could be performed by an ARH.
- To survey the technology relevant to developing an ARH.
- To develop a concept which satisfies the requirements within both overall NASA and ARH program constraints.

The subsequent development work objectives were:

- To build a test bed for analyzing the study recommendations.

*The majority of this work was conducted under contract no. NAS9-17454 for NASA Lyndon B. Johnson Space Center and reported in the report entitled "Design Study of a Prototype Anthropomorphic Robotic Hand for Use with an Extravehicular Space Suit." This report was submitted in September 1986. The report contains a reference list with 196 titles.

- To utilize the prototype hardware to refine the concepts and improve our understanding of hand master and slave design.

DESCRIPTION OF STUDY METHODOLOGY

The first step in this project was to develop the operational requirements placed on the gloved hand in past, current and projected future EVA's by analyzing the tasks that comprised six representative scenarios (Figure 1). Following the development of the operational requirements, both bare-handed and suited data were compiled to quantify each requirement. From these requirements and other relevant data, the design goals and constraints were developed. These were divided into two categories: improvement to be maximized over the gloved hand (e.g., dexterity, range of motion and comfort) and degradation to be minimized over the gloved hand (e.g., safety, mental load, and training).

After the design goals and operational requirements were developed, the technology which might be used to fulfill those requirement and goals was surveyed. The design of an ARH shares many of its goals and constraints with other manipulator program in robotics, teleoperation, prosthetics and orthotics. In addition to surveying previous experience in these areas, a study of individual technologies for major system components (e.g., transmissions, actuators, sensors) was also performed. This study was aimed at uncovering technologies but, which for one reason or another had not previously been used in manipulator systems but which, on the basis of our goals and requirements, could be applicable to the ARH.

Finally a number of ARH concepts were developed, analyzed and optimized for fulfillment of the design goals within the NASA and ARH program constraints. From this trade-off analysis an optimized concept was produced, and a set of development steps required to yield a device which could enhance EVA performance and productivity was formulated.

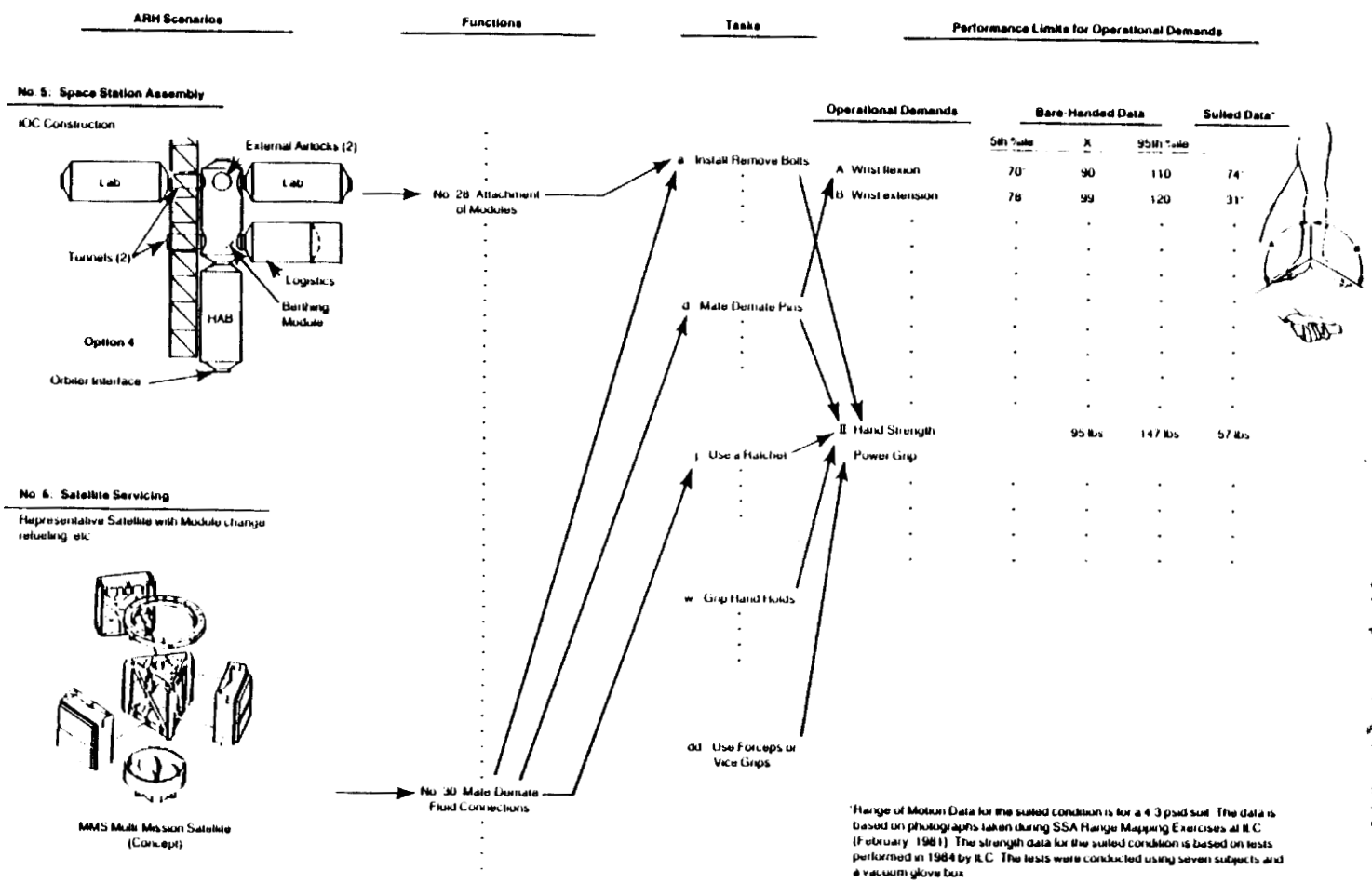
SUMMARY OF STUDY RESULTS

Operational Requirements Development

The task analysis showed that for the six chosen EVA scenarios the most prevalent operations were:

- forearm supination/pronation
- finger motion
- power grip
- palmer grip
- finger pull

FIGURE 1 ARH TASK ANALYSIS APPROACH



- o shoulder medial/lateral rotation, and
- o wrist adduction/abduction.

These operations were seen as essential to successfully performing EVA tasks.

Technology Survey

From the technology survey on manipulator programs in robotics, teleoperation, and prosthetics and orthotics we learned that:

- o Several dexterous hands which might meet the ARH operational requirements had been built but none of them were suitable or easily modifiable for this application.
- o Autonomous control of these hands for a range of tasks had not been accomplished.
- o No kinematically correspondent master for teleoperated control of these hands had been used in performing useful tasks.
- o A large number of important tasks can be accomplished with less than anthropomorphic hands.

From the survey of individual technologies for system components we learned that:

- o Actuators with appropriate frequency response and output forces and torques have been demonstrated or can be reasonably expected to meet the requirements.
- o A variety of transmission methods exist which could be used for an ARH.
- o A wide variety of sensors are available for providing data about the ARH and the environment around it.
- o Display of sensory information has been achieved and successfully used in a variety of methodologies.

Thus we concluded that although there was no similar system already in existence, the available technology could be used to create a system which would meet the goals within the constraints.

Concept Development

A variety of options ranging from simple to complex were developed and studied (Figure 2). These options ranged from ARH designs which attached to the space suit arm in place of the glove to Third Arm configurations which attached to the torso or life support system and employed control systems and end effectors with various degrees of complexity. From these options, seven were chosen for their potential performance and to span the range of options. They were developed and designed in sufficient detail to permit performance prediction and trade-off analysis.

Concurrently a trade-off analysis methodology was developed utilizing the operational requirements and design goals and constraints. An equation was developed for each design goal and constraint, relating data describing each design to the bare hand and 3000 series gloved hand. The goals and constraints were ranked by NASA personnel and the ADL team, and weightings were developed (Table 1). From these weightings, the equations, and the design data developed for each of the seven concepts, overall scores were obtained (Table 2). The scoring results indicated that the following features were primary discriminators between concepts relative to enhancing EVA performance and productivity:

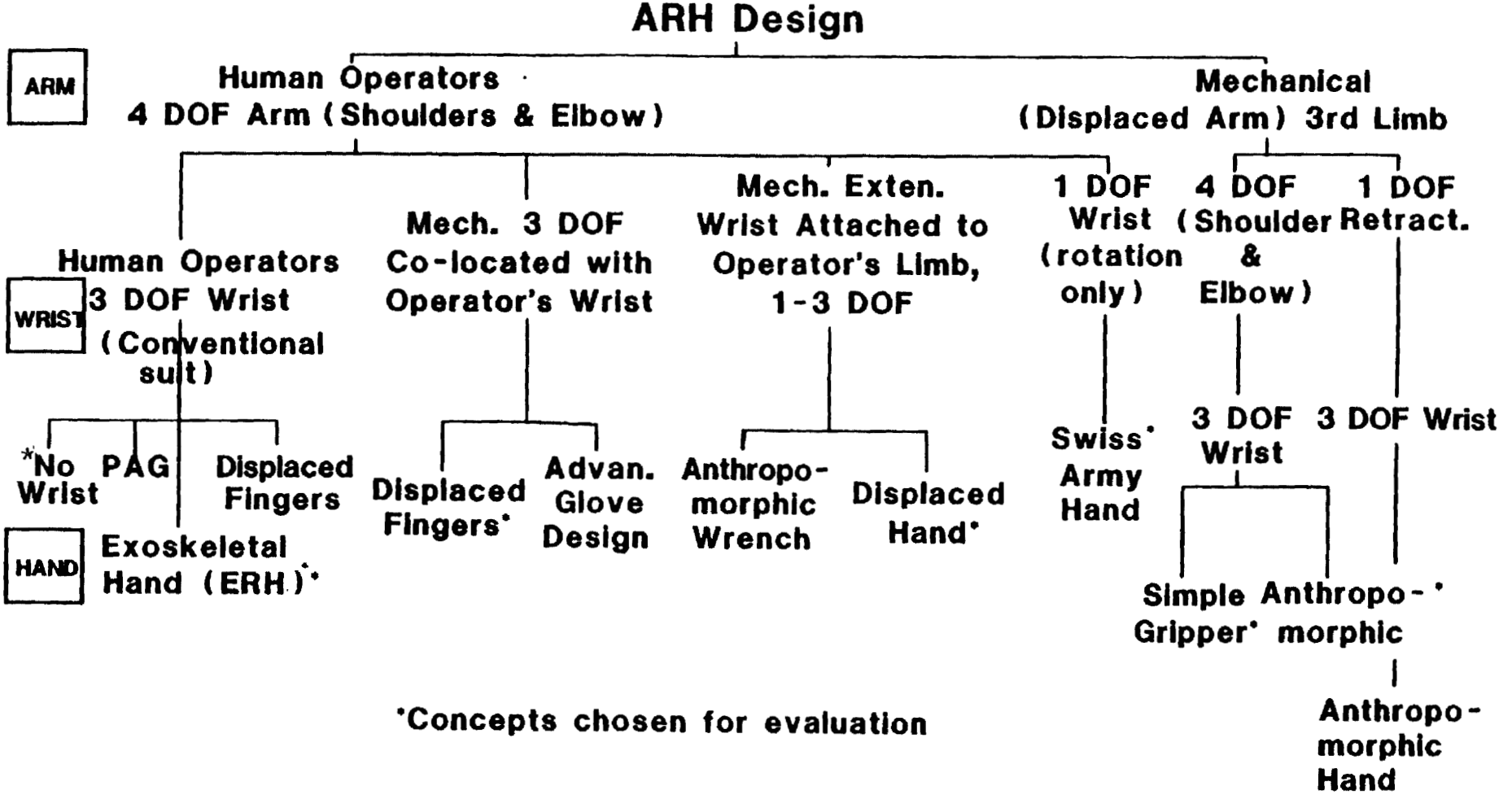
- o wrist degrees of freedom (DOF) and range of motion
- o grasp and wrist lock
- o number of fingers and DOF of each
- o mechanism complexity
- o feedback of ARH status to the operator.

Using the results of this analysis, a design was formulated that could be built during the time allocated to the prototype phase of this contract. This device, the optimized displaced fingers (ODF), was a direct mechanically linked master/slave that would be worn on the end of the suit arm. It had three, three jointed fingers, a 2 DOF thumb, a locking wrist, grasp lock and a 3 DOF wrist co-located with the operator wrist (Figure 3). This design as then evaluated using the trade-offs methodology and received the highest score of all the designs considered.

STUDY CONCLUSIONS

Based on the results of this study, we have concluded that the "optimized, displaced fingers" (ODF) design is the approach which best meets the requirements for an ARH as defined in this project. The device would have good dexterity, mobility, and fatigue characteristics and would provide improved performance over the present glove for the EVA tasks selected as typical for this study.

FIGURE 2 Design Options



*Concepts chosen for evaluation

TABLE 1 RANKING OF DESIGN GOALS & CONSTRAINTS

Improvement to be Maximized Over Gloved Hand:

		Weight (%)
1.	Dexterity	9.0
2.	Maximum Force	5.3
3.	Maximum Torque	6.0
4.	Range of Motion	6.2
5.	Arm Dynamics	4.5
6.	Feedback	7.6
7.	Muscle Fatigue	7.7
8.	Comfort	6.7
9.	Impact Tolerance	5.2

Degradation to be Minimized Over Gloved Hand:

10.	Safety (during operation)	11.0
11.	Training	3.4
12.	Mental Load	7.6
13.	Failure	11.4
14.	Maintenance	2.5
15.	Power Required	3.2
16.	Cost	2.7
TOTAL		100

TABLE 2 FINAL DESIGN SCORES

	Displaced Fingers	Displaced Hand	Third Arm/Simple	Third Arm/Complex	No Wrist	Swiss Army Hand (SAH)	Exoskeletal Robotic Hand (ERH)	Optimized Displaced Fingers
DESIGN GOALS	1	2	3	4	5	6	7	8
Subtotal for Maximized:	83.72	71.96	-151.05	-48.24	67.69	-4.23	11.19	131.70
Subtotal for Minimized:	-83.22	-96.25	-56.02	-210.20	-28.00	-109.23	-34.16	-81.81
Total Score	0.49	-24.29	-207.07	-258.44	39.70	-113.46	-22.97	49.89

ORIGINAL PAGE IS
OF POOR QUALITY

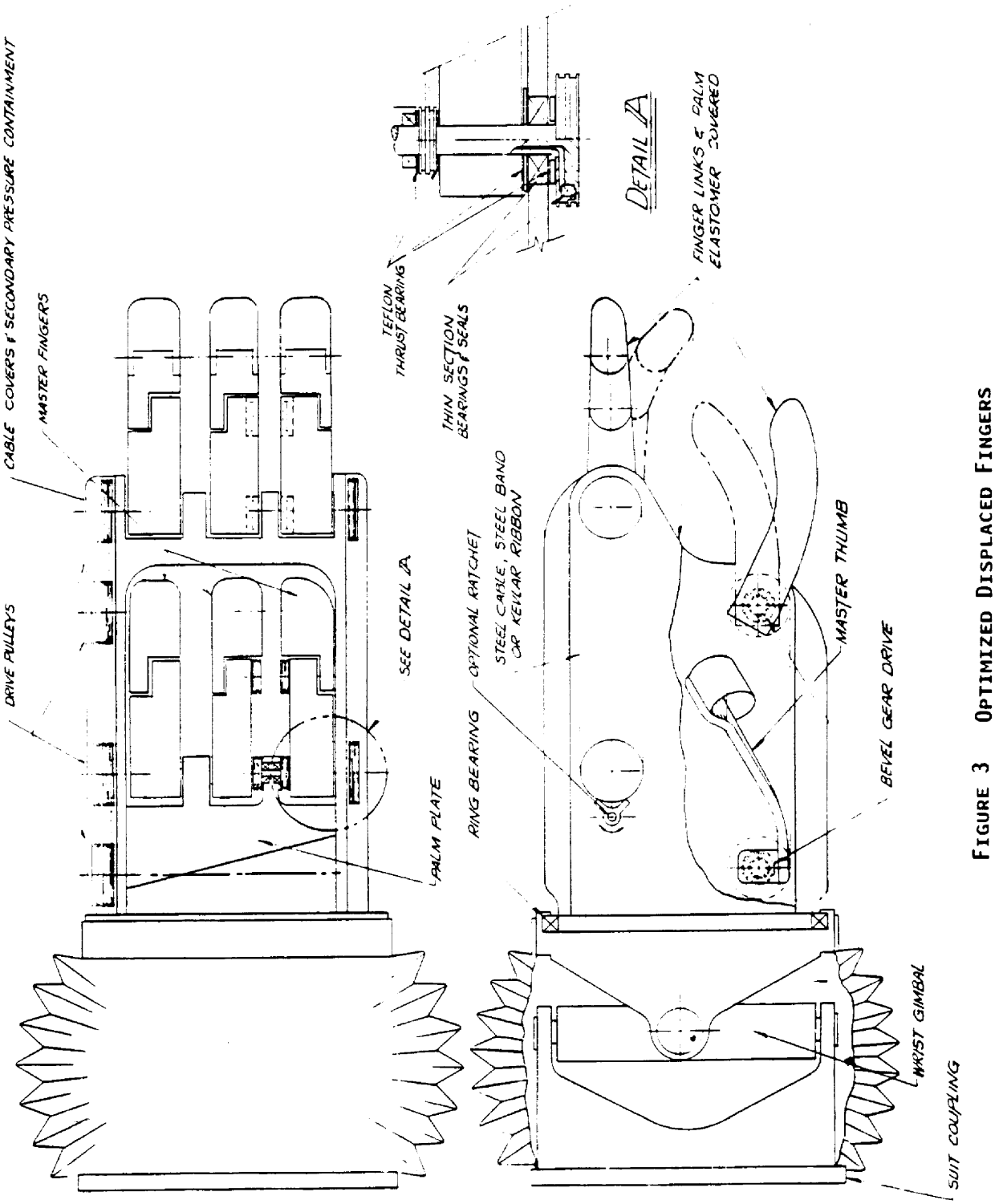


FIGURE 3 OPTIMIZED DISPLACED FINGERS

An important consideration in the evaluation of design alternatives was an assessment of the state of development of dextrous manipulators. Thus, the recommended design represents, in a way, a compromise between a device which could be developed to provide enhanced performance drawing on demonstrated technology and the ultimate advanced, powered, dextrous manipulator. The approach does, however, provide a basis for making significant advances in the state-of-the-art and is a logical first step in the development of the advanced design. Specific advances include:

- Direct feedback: The design incorporates direct feedback and will allow a determination of how the information is used and how much feedback is desirable for future systems.

- Master-slave design: The design of a kinematically correspondent directly coupled master for an anthropomorphic end effector has not been done. This device requires a master which tracks individual fingers to operate the slave. The master would be linked mechanically to the slave in this case, but in advanced designs the master could be equipped with kinesthetic transducers and force reflection coupled with sensors on the slave to provide remote operations.

- Task performance: The device will permit evaluation of the usefulness of an ARH in performing specific tasks and therefore will be useful in developing design specifications for more advanced systems.

For this device to fulfill the objective of enhancing EVA crew productivity, it must be acceptable to the crew and must operate safely in the EVA environment. The ODF is inherently less acceptable and arguably less safe than the third arm options (Figure 2) considered because it seeks to replace the human hand function and also modifies the pressure suit. The human hand is a very complex and well designed device. Even with the mobility, dexterity and fatigue limitations imposed by the current EMU gloves it provides a wide range of capabilities essential to EVA including adaptability to unforeseen conditions.

A perfect ARH would be as flexible and adaptable as the human hand and would not fatigue. This type of device is not currently within the state-of-the-art and could not be developed for space applications in the near term (2 - 3 years). The ODF provides the closest approximation to that capability compatible with the current state-of-the-art while providing a basis for developing future hands which approach the capability of the human hand.

Even with this first step, the development of the ultimate hand system which would be safe and acceptable would not be possible until the 1990's. In the meantime, EVA crewmen will need productivity aids, and therefore, a short-term, safe, acceptable solution should also be developed. We believe that the best alternative is a version of the simple third arm

(Figure 4). This device could be safely interfaced with a crewman and does not require the loss of use of the gloved hand. If powered or manually locked it provides a three arm working capability. It would provide near term operational experience with robotics in EVA.

The Third Arm design should have the following characteristics. It should be simple to use and capable of performing a limited subset of tasks reliably and effectively. It's end effector should provide only grasping and limited manipulations. Therefore, in the short term, dexterous operations would continue to be performed by the gloved hand.

If gloves improve significantly, this simplified ARH (Third Arm) may be the only capability required. If suits go to higher pressures, gloves do not improve sufficiently, or the ODF leads to a highly dexterous ARH with capabilities approaching the human hand, an ARH attached to the suit arm may become desirable.

In summary;

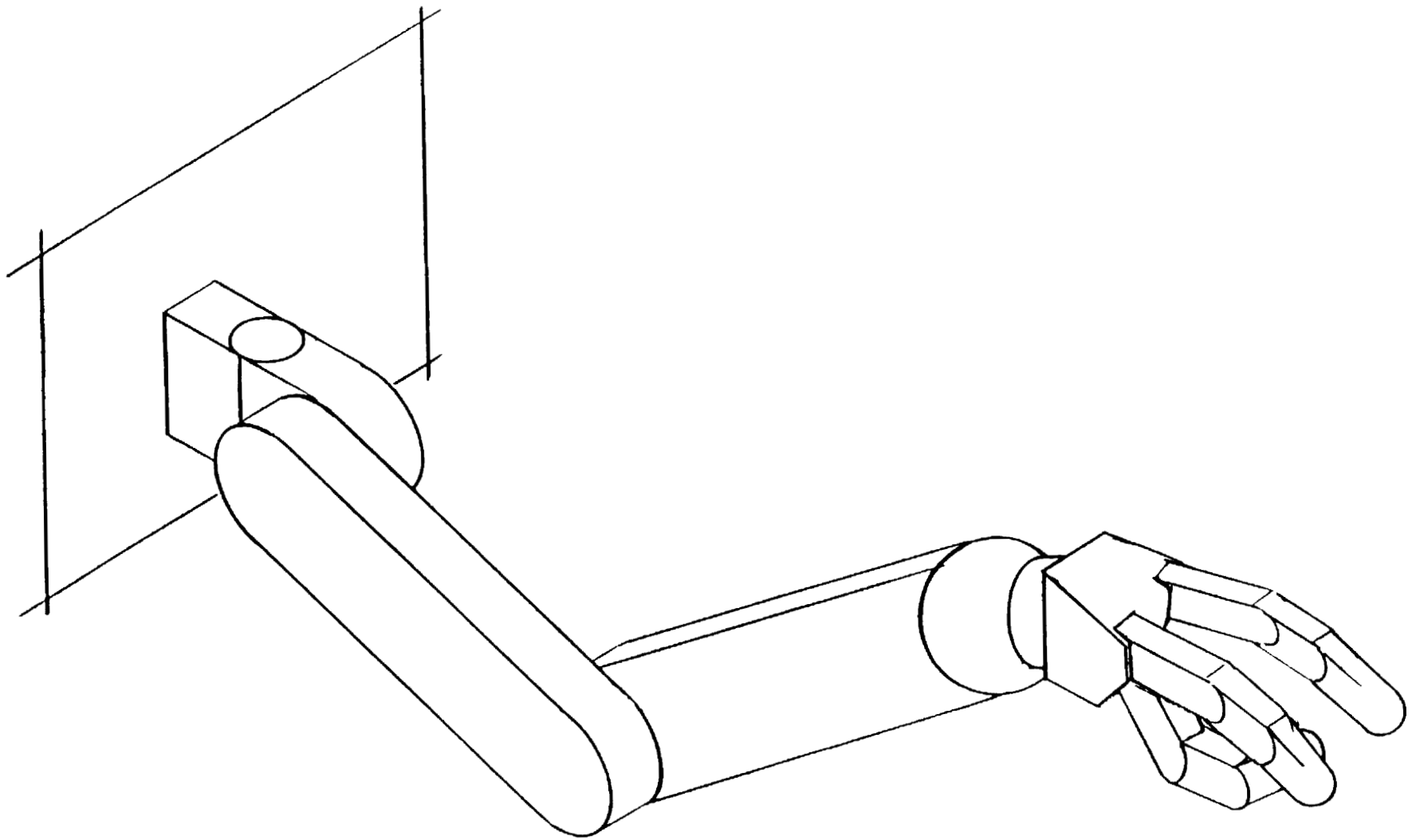
- o The ODF is a useful ARH and can provide information crucial to future hand developments which could be applicable to an advanced ARH or Third arm.
- o A highly dexterous hand/arm system requires significant R&D to become an operational space tool.
- o A simple Third arm is an acceptable short term EVA productivity aid.
- o The Simple Third arm could evolve to a useful complex device or be replaced by an advanced ARH depending on parallel developments in other areas.

METHODOLOGY OF ADDITIONAL IN-HOUSE DEVELOPMENT WORK

In the course of studying hand function, robot hand design and developing the previously described concepts, we identified a number of important design issues requiring further work prior to entering a complete system hardware development. They were:

- o Design of a master to interface with a human hand without severely restricting its motion.
- o Stiff transmission design to minimize friction and backlash.
- o The utility of the overall link configuration including the function of the passive third link.

FIGURE 4 Simple Third Arm Concept



Arthur D. Little, Inc.

To study approaches to these issues, we designed and built a two-finger master/slave test bed (Figure 5). We chose this particular configuration to fully test the finger kinematics, test the thumb position and allow simple grasping to be able to assess potential performance. The system was designed so that additional fingers could be accommodated to allow us to examine the feasibility of a nested shaft design and evaluate the lateral placement of the pulleys. The design approach selected was similar to that shown in Figure 3.

PRELIMINARY RESULTS OF DEVELOPMENT WORK

The hand test bed as originally conceived utilized what we considered to be a near anthropomorphic thumb placement. Once the device was built and testing began, we realized that one seldom utilizes the thumb in its relaxed orientation (Figure 6a). We found for grasping a range of objects the effective degrees of freedom provided by the palm and thumb together allow a more useful configuration (Figure 6b). As our designs do not have these degrees of freedom, we modified the geometry to reflect this more useful hand configuration. Further testing will determine whether this provides sufficient capability for the desired tasks.

In building the hand master for the test bed, we found that packaging the linkage to be compatible with a range of human hands with the volume allocated is a difficult problem. The center of rotation of the links should be at the center of rotation of the finger joints. If not, relative motion between the master and the human hand will result. For a small number of fingers (1-3) for a particular person matching the centers of rotation of the joints may be achievable in a system employing rigid links, but, with hand size variation and larger numbers of fingers, it becomes difficult. One solution which makes use of the human sensory motor adaptability is to design to permit a certain amount of relative motion. Preliminary findings show that this is a useful methodology, and further development will explore the limitations of this approach.

The passive third link was constructed and has been found, as predicted, to be useful in grasping a variety of objects. In the configuration we tested, its motion is directly proportionate to the proximal link. This configuration may not be optimal for grasping oddly shaped objects, but may be excellent for the tasks of interest. Further testing will lead to a better understanding of these issues, and perhaps the application of a certain degree of compliance to this linkage.

The initial transmission configuration used tensioned cables. This increased bearing friction and presented some operational problems. Other cable materials maximizing stiffness while minimizing the required pretensioning and other transmission concepts are currently being evaluated to improve this situation.

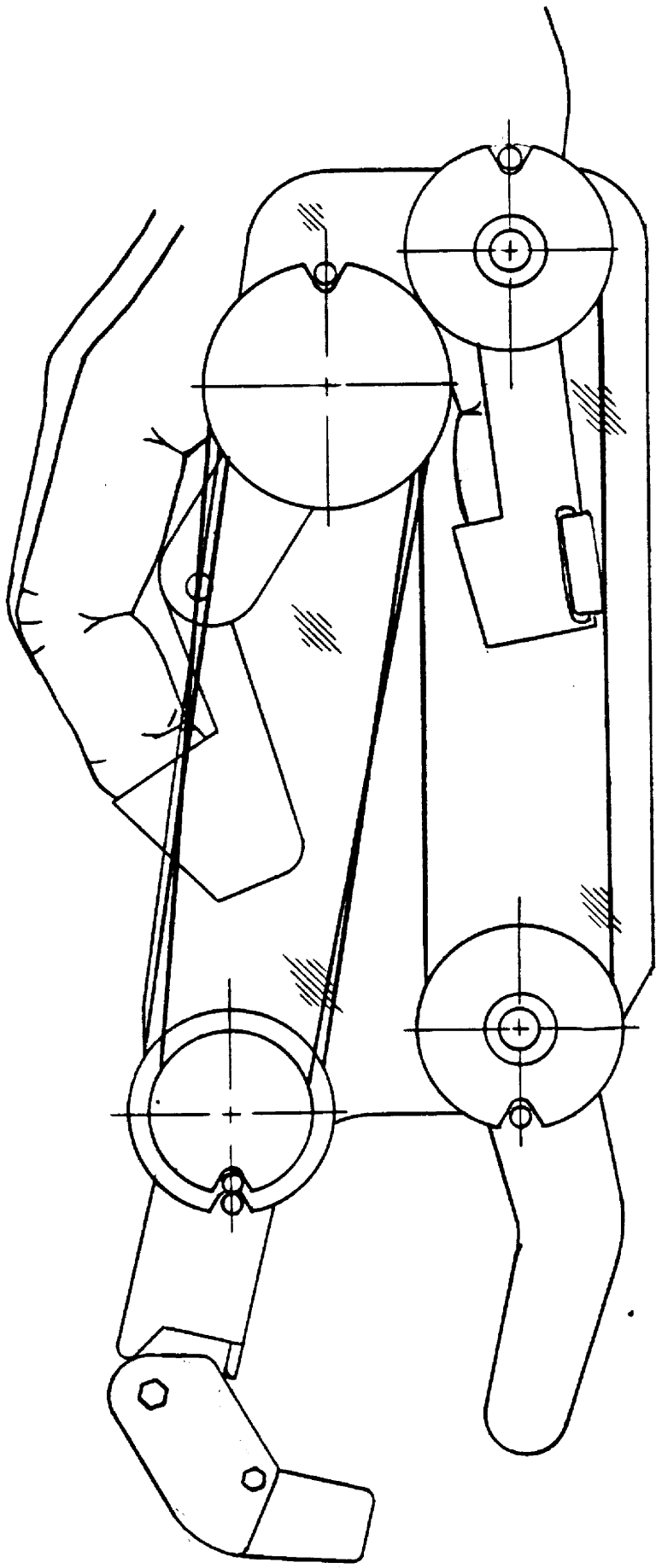


FIGURE 5 2 FINGER TEST BED

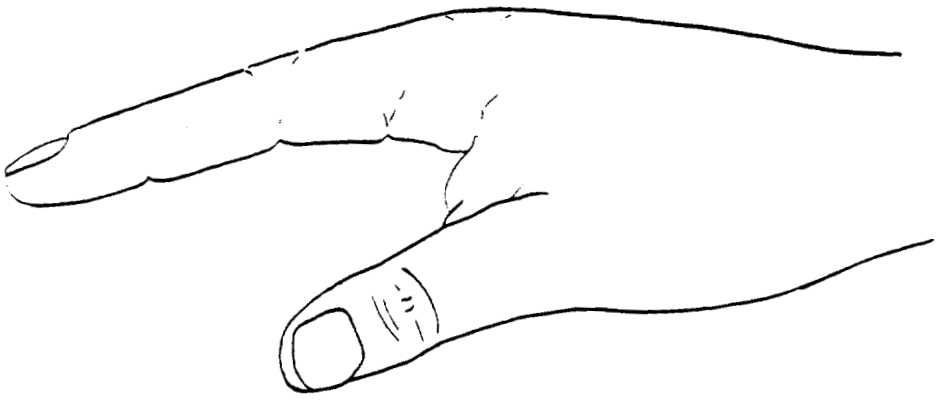


FIGURE 6A

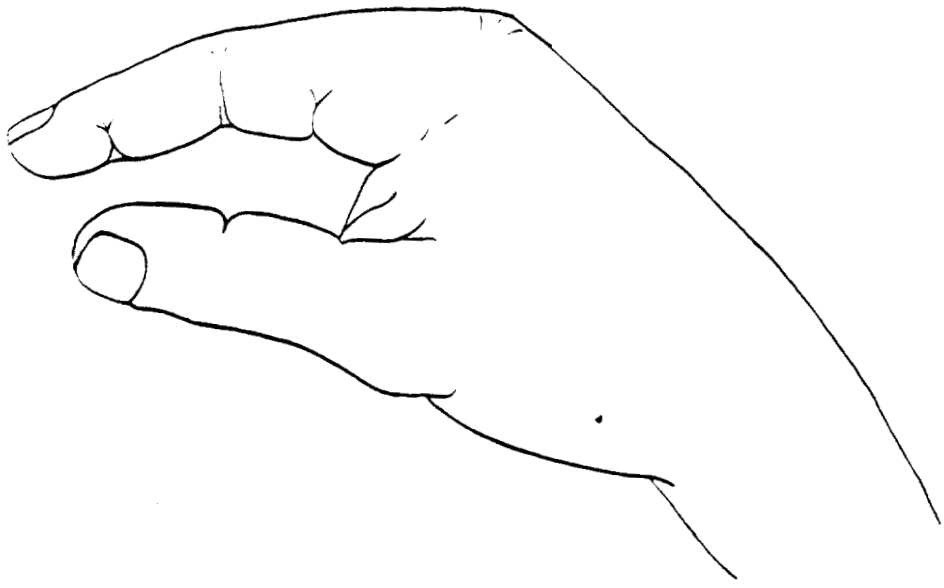


FIGURE 6B

Conclusions of Developmental Work

Although the work described is still in progress, the objectives of the work have been achieved. This work has shown the utility of studying simple devices, such as our test bed, to provide a solid foundation for future complex systems. Before dexterous hands can be designed for remote operation much more of this type of experimentation needs to be accomplished.

DYNAMIC TASK ALLOCATION FOR A MAN-MACHINE SYMBIOTIC SYSTEM*

L. E. Parker and F. G. Pin

Center for Engineering Systems Advanced Research
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831-6364

Paper Submitted to:

Goddard Conference on Space Applications
of Artificial Intelligence and Robotics
May 13-14, 1987
Greenbelt, Maryland

*Research sponsored by the Engineering Research Program of the Office of Basic Energy Sciences, of the U.S. Department of Energy, under contract No. DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

ABSTRACT

This paper presents a methodological approach to the dynamic allocation of tasks in a man-machine symbiotic system in the context of dexterous manipulation and teleoperation. This paper addresses symbiosis containing two symbiotic partners which work toward controlling a single manipulator arm for the execution of a series of sequential manipulation tasks. The proposed automated task allocator uses knowledge about the constraints/criteria of the problem, the available resources, the tasks to be performed, and the environment to dynamically allocate tasks to the man and the machine. The presentation of the methodology includes discussions concerning the characteristics of the man-machine symbiotic system, the interaction of the knowledge areas, the flow of execution, and the dynamic nature of the task allocation.

1.0. INTRODUCTION

During the last few decades, there has been a growing awareness and belief that automation-related technologies and intelligent machines will play an increasing role in improving the development and operation of complex and advanced systems. In this context, research and development has taken place on a broad range of technologies aimed at achieving automated systems varying from fully remotely-controlled systems such as advanced teleoperators and servomanipulators to fully autonomous intelligent robots involving artificial intelligence, super-computing, machine vision, and advanced control. Within this large spectrum of technological research, work has recently been initiated on what is proposed to be a new class of automated systems which appear promising for improving the productivity, quality, and safety of operation of advanced systems. This new type of automated system is referred to as "Man-Machine Symbiosis" and would utilize the concepts of machine intelligence and remote-control technology to achieve full man-machine cooperative control and intelligence [2].

The ultimate function of such symbiotic systems would be to dynamically optimize the division of work between the man and the machine and to facilitate their cooperation through shared knowledge, skills, and experiences. The optimization of the man-machine partnership in both the electromotive and intellectual domain would be realized by coupling a dynamic allocation of tasks between the human and the machine with an embedded system learning capability to allow the machine, an intelligent robotic system, to learn new tasks through assimilation of experience and observation of the human [3], [4], [5].

This paper presents a methodological approach to the dynamic allocation of tasks for a man-machine symbiotic system in a simplified case of dexterous manipulation and teleoperation. In this formulation, two symbiotic partners are considered: a human teleoperator and an intelligent robotic system. Both partners work toward controlling a single manipulator arm for the execution of a series of sequential manipulation tasks. Section 2 of the paper outlines the characteristics of the specific man-robot symbiont considered here, while section 3 presents a generalized task allocation procedure. For an example illustrating the results of the conceptual architecture in the context of remote manipulation, refer to [7].

2.0 CHARACTERISTICS OF A MAN-MACHINE SYMBIOTIC SYSTEM

The man-machine system addressed in this paper consists of two symbiotic partners, a human teleoperator and an intelligent robot system with its controller, which cooperate to perform a series of sequential manipulation tasks involving a single manipulator arm. To facilitate the division of work between the man and the robot, several automated modules are proposed to be incorporated into the system to perform responsibilities such as task subdivision, analysis, and allocation. Such a scenario can be depicted as shown in figure 1.

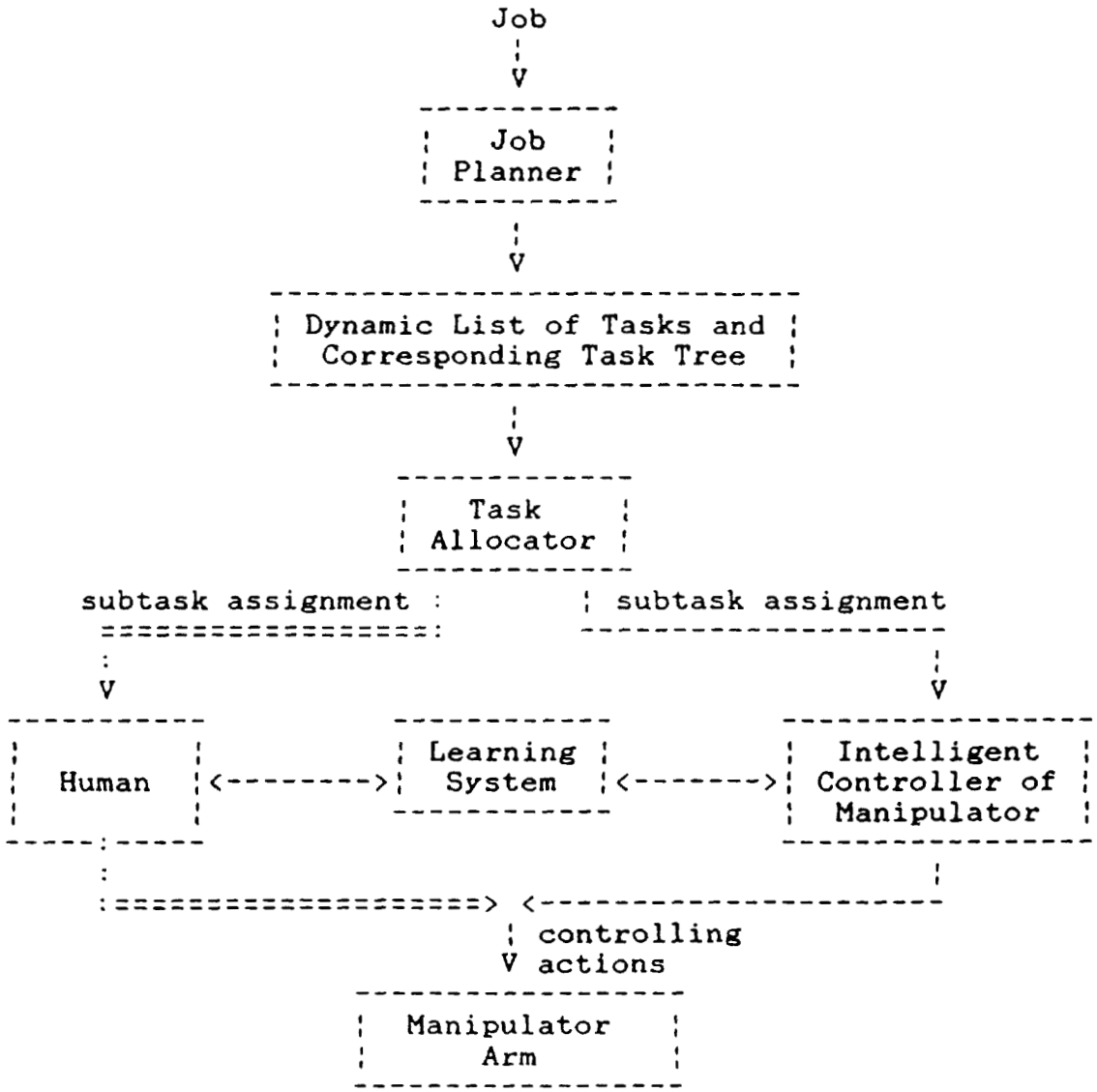


Figure 1

A job planner is responsible for decomposing the overall job to be performed (such as INSTALL ELECTRICAL EQUIPMENT) into its component lower-level subtasks (such as FIND WRENCH or GRASP WRENCH), indicating the order in which the subtasks must be performed. The resulting task decomposition tree (see section 3.1.3), is passed to the task allocator, which assigns a subtask either to the human or to the intelligent robot controller of the manipulator. The human or the intelligent robot controller then sends controlling actions to the manipulator arm for execution of the subtask. To improve its performance and to increase its range of capabilities, the intelligent robot controller of the manipulator arm must ultimately use an embedded learning system to learn new tasks through assimilation of experience, observation of the human, and direct instruction [3], [4], [5].

This paper is concerned only with the task allocator and its relationship to the other entities in the man-machine symbiotic system. This paper assumes that a complete description of the tasks to be performed is provided to the task allocator by either the human or an automated system. Research is currently being performed on automating the job planner. This paper also does not discuss any details related to the embedded learning system, which is currently being researched and will be discussed in future publications.

To determine the necessary characteristics of the task allocator in this symbiotic system, one can first observe that both intelligent resources (the human and the intelligent controller of the manipulator arm) are using the same medium (the manipulator arm) to execute the subtasks. The manipulator arm actuator can receive and respond to commands from a single source at any instant in time. Consequently, the human and the intelligent robot controller cannot command the arm simultaneously or independently. Therefore, the task allocator must deal with the allocation of sequential manipulation tasks, rather than concurrent tasks. However, it is likely that while the human or the machine is performing a subtask with the manipulator arm, other actions are occurring in the background, such as monitoring of the task execution, world modeling, planning, and learning. This aspect is necessary in order for the symbiotic system to function effectively. Nevertheless, as a first step, this work will focus on the sequential task problem of allocating a series of sequential manipulation subtasks to the man and the machine. Research is currently underway to extend this methodology to allow the human

and/or the machine to perform additional subtasks which compete for their time while the manipulation subtasks are being performed.

Another essential requirement of the task allocator in this man-machine system is its ability to be event-driven, responding to changes in the work constraints, physical environment, or unexpected events by altering the task allocation to adjust to new conditions. This dynamic nature of the task allocator allows the man-machine symbiont to cope with a changing environment, causing the resource most appropriate for performing a subtask to be assigned the subtask. In order for a dynamic allocation of subtasks to be successful, the human and the intelligent controller of the manipulator arm must be able to perform at least some of the subtasks interchangeably; otherwise, the allocation can be automatically pre-determined simply by assigning each subtask to the only resource that is able to perform it. Such a static allocation of subtasks is intolerant of faults, for if one resource failed in performing its subtask, another resource could not take over the operation of that subtask. The dynamic allocation of subtasks, however, does not usually suffer from this symptom, and can result in an effective use of the resources which is more tolerant to resource faults [1]. Note that even the dynamic method of task allocation will not be completely intolerant to resource faults during the execution of subtasks which can only be performed by one specific resource.

In summary, the task allocator in this symbiotic system must deal with the dynamic allocation of sequential manipulation subtasks to two resources, a human and an intelligent robot controller, responding to events during the subtask execution which lead to a reallocation of subtasks. The remainder of this paper will address a task allocation methodology having these characteristics.

3.0 DYNAMIC TASK ALLOCATION METHODOLOGY

3.1 KNOWLEDGE AREAS

The purpose of the task allocator in man-machine symbiosis is to attempt to dynamically optimize the division of work between the man and the machine. Since the exact interpretation of "optimal division of work" must be allowed to vary according to the requirements of each individual problem scenario, the task allocator must know what constraints and criteria are placed on the task allocation, what the requirements of the subtasks are, and information concerning the characteristics of the environment in which the problem is to be solved. The task allocator must also

have information about the capabilities of the human and the intelligent robot controller to determine the resource which is most appropriate for performing a subtask in a given scenario. The knowledge about these areas can be categorized into four main knowledge bases which are described in the following sections.

3.1.1 CONSTRAINTS/CRITERIA

The constraints/criteria are determined by a source external to the task allocator and place performance measures, limitations, restrictions, and/or regulations on the task allocation problem solution. The intent of the constraints/criteria is to alter the task allocation strategy to adapt to differing problem contexts. The task allocator must adhere to these constraints/criteria in determining the task allocation. These limitations may prevent the use of certain resources for some subtasks, or may mandate the use of certain resources for other subtasks.

Examples of possible constraints/criteria are as follows:

-- minimize time of job completion
-- maximize quality of result
-- minimize human involvement (e.g. in a hazardous environment or to prevent boredom or fatigue)

The task allocator must know how to handle any constraint that is placed on the solution. For example, if the constraint is to minimize the time of task completion, the task allocator must compute the estimated time each resource will take to complete a subtask (refer to sections 3.1.2 and 3.1.3 for further information) and then assign the subtask to the resource requiring the lesser time. For each application of the task allocator, certain constraints/criteria are initially in effect while other constraints/criteria are ignored. Although this paper only deals with situations having one constraint in effect at a time, this methodology has the potential for being extended to handle combinations of several constraints/criteria for the optimization of the solution.

3.1.2 RESOURCES

In this paper, resources are defined to be intelligent entities (such as humans or computers) which are available for performing subtasks to solve a problem, or to achieve a goal. In this paper, only two resources are considered: a human and an intelligent robot controller. Obviously, the task allocator must have some information concerning the available resources before it can begin the job of task allocation. The task allocator must know what capabilities

each of the resources possess, how well the resources use their capabilities in performing subtasks, how timely the resources use their capabilities to perform the subtasks, and the current status of the resources (i.e., when each resource will be available to perform subtasks). The capabilities of the resources are defined in this paper to be either the abilities the resources have to perform certain physical actions, or the knowledge the resources have of certain objects. The capabilities can be defined as needed for particular applications, and could include physical abilities such as MANIPULATION or VISION, or knowledge of objects, such as WRENCH or BOLT.

Each resource can have many capabilities. However, a resource will probably not have the same level of achievement of each of its capabilities, and it certainly will not exercise each capability with identical speeds. For example, although a human has capabilities of both COMPUTATION and VISION, he probably can examine a photograph (using VISION) much easier and better than he can add a few numbers in his head (using COMPUTATION). On the other hand, a computer may also have capabilities of COMPUTATION and VISION, yet it is much more difficult for it to examine a photograph than it is for it to add a few numbers.

The knowledge about the capabilities of the resources is initially given to the task allocator as input. The actual information stored about the capabilities of the resources is directly related to the constraints which might at some time be present in the problem scenario. For example, the constraint "minimize time of task completion" requires that "timeliness of achievement" factors be provided, while the constraint "maximize quality of result" requires that "level of achievement" factors be provided. Additional constraints placed on the problem may require the storage of further information on the capabilities of the resources.

Although the knowledge about the capabilities is quantified differently depending upon whether the capability refers to a physical ability or to a knowledge about an object, one evaluation number is obtained for each factor (such as level of achievement and timeliness of achievement) of each capability. The evaluation numbers are then used to help determine the appropriate task allocation. If the capability refers to a physical ability, the evaluation number indicates the skill with which the ability is performed, perhaps on a scale from 0 to 10, or from "unacceptable" to "superior". If the capability refers to a knowledge about an object, the evaluation number indicates how complete the knowledge of that object is, perhaps on a scale from 0 to 10, or from "unknown" to "always known". Depending on the constraints of the given problem and the

subtasks to be performed, the task allocator can select the suitable resources to perform the subtasks based on the characteristics of the resources. This is done by determining what capabilities are required to complete each subtask, finding the available resources which possess the required capabilities, and applying the constraints/criteria of the problem to compute the optimal allocation.

The task allocator would thus have information as follows for the resources:

resource	capa-bility	level of achievement	timeliness of achievement	availability
R_1	a_{11}	l_{11}	t_{11}	w units
	a_{12}	l_{12}	t_{12}	x units
	.	.	.	.
	.	.	.	.
	.	.	.	.
	a_{1n}	l_{1n}	t_{1n}	y units
R_2	a_{21}	l_{21}	t_{21}	w units
	a_{22}	l_{22}	t_{22}	x units
	.	.	.	.
	.	.	.	.
	.	.	.	.
	a_{2n}	l_{2n}	t_{2n}	y units
. . .				
R_m	a_{m1}	l_{m1}	t_{m1}	w units
	a_{m2}	l_{m2}	t_{m2}	x units
	.	.	.	.
	.	.	.	.
	.	.	.	.
	a_{mn}	l_{mn}	t_{mn}	y units

For example, information which could be obtained from a table such as this is as follows:

- The human has the capability of VISION, can perform VISION on a level of 10 (or "superior") with a "timeliness factor" of 2 (or "extremely fast"), and is currently available to perform VISION.

- The human has the capability of MANIPULATION, can perform MANIPULATION on a level of 7 (or "fairly good") with a timeliness factor of 4 (or "fairly fast"), but is not currently available to perform MANIPULATION. The

human will be available to perform MANIPULATION in 3 time units.

- The computer has the capability to RECOGNIZE WRENCH, can RECOGNIZE WRENCH on a level of 4 ("sometimes known") with a timeliness factor of 7 ("fairly slow") , and is currently available to RECOGNIZE WRENCH.

Some important observations can be made in examining this table. First, a resource can have more than one capability available at a time, and it can also use more than one capability at a time in the execution of a subtask. The use of more than one capability at a time should not be confused with the execution of more than one subtask at a time. The resource will only be performing one subtask at once, although it may use several capabilities to accomplish that subtask. For instance, a concurrent computer can use one processor for the capability VISION and another processor for the capability COMPUTATION. Likewise, humans can use the capability of VISION while using the capability of MANIPULATION to hammer a nail. Thus, the use of one capability of a resource does not necessarily mean that the other capabilities of that resource are inaccessible.

The second observation from examination of the table is that since only two resources are considered in this paper (a human and a machine), the above table in an actual application would have only two entries: R1 and R2. However, the extension to m resources is possible and would allow many resources to be considered in the execution of the sequential manipulation subtasks.

3.1.3 TASKS

A job planner must analyze and decompose the job to be performed into its component tasks, subtasks, and sub-subtasks. The role of the job planner can be fulfilled by either the human or an automated job planning system. The current paper does not address the operation of the job planner and assumes that the task breakdown is available as input to the task allocator. An automated job planner for the system will be addressed in a companion publication.

A typical task breakdown tree is shown in figure 2a.

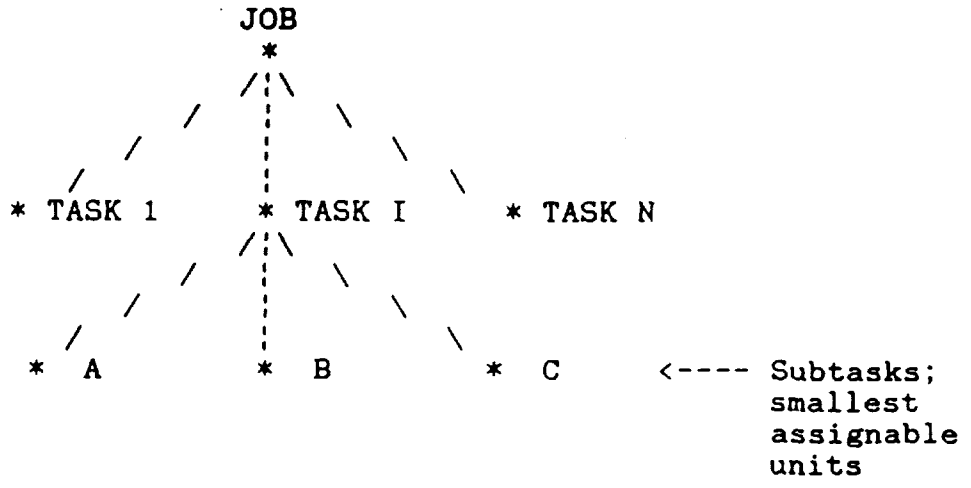


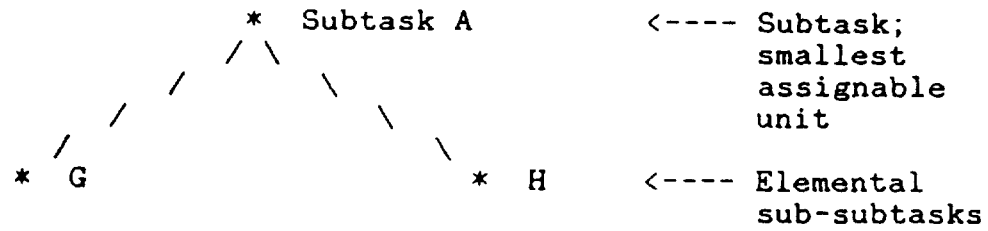
Figure 2a

The job is the highest-level description of a series of related tasks to be performed, such as ASSEMBLE MODULE. The job is decomposed into several tasks, such as INSERT ROD, which must be successfully completed by the resources in order to solve a problem, or to achieve a goal. Each task can be performed entirely by the human, entirely by the computer, or by the human and computer in cooperation. Each task is subdivided as much as needed until the smallest assignable units, or subtasks, are reached. These subtasks are the smallest units that can be feasibly assigned to a single resource. For example, a task UNPLUG CABLE could consist of subtasks FIND CABLE, MOVE TO CABLE, GRASP CABLE, and PULL CABLE. It would be senseless to assign smaller components of these subtasks to more than one resource. The concept of a "smallest assignable unit" is very important since it represents the smallest subdivision of the elements of a task which correlate with the physical mechanics of the actual operation of the symbiotic resources. The definitions of resources, capabilities, and smallest assignable units are, in general, system and task domain dependent.

In order to allocate the subtasks, the task allocator must know what capabilities are required to perform the subtasks and any merit factors associated with each capability. Due to the considerable differences between the intelligent robot controller and the human, the capabilities required for one of these resources to perform a subtask may be very different from those required by the other resource. Because of this, the subtasks must be further subdivided for each resource down to the elemental sub-subtasks which can

be characterized by one or more capabilities and merit factors which are independent of the environment or the context of the problem. An example of the subdivision is shown in figure 2b.

For R1:



For R2:

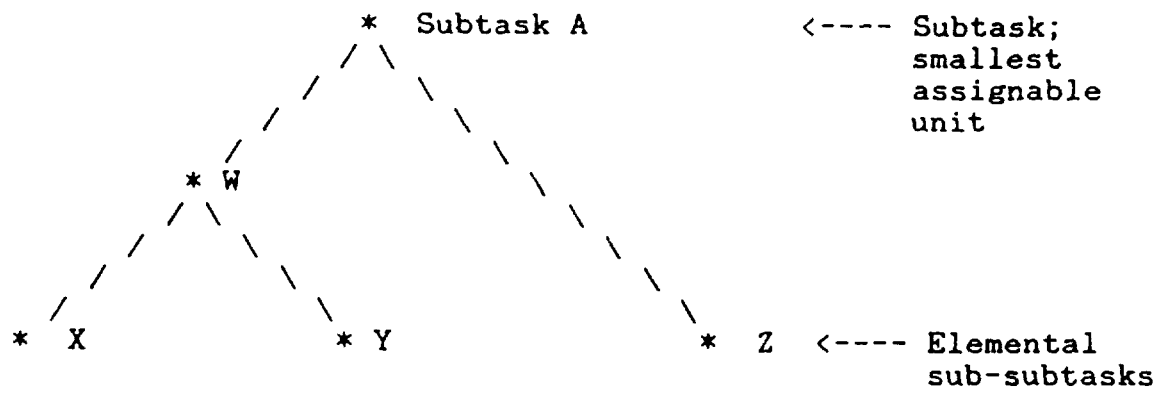


Figure 2b

The list of capabilities required for each subtask is obtained by traversing the lowest-level nodes (leaves), or elemental sub-subtasks, below the subtask in the task breakdown tree, noting all the capabilities required for the lowest-level nodes, or elemental sub-subtasks. This traversal must be performed for each resource, since the resources have different sub-subtask breakdowns, as shown in figure 2b. The merit factor associated with each capability indicates the importance of that capability in the successful performance of the elemental sub-subtask, relative to the other required capabilities. The merit factors are obtained for the capabilities in a manner similar to how the list of required capabilities is obtained -- by traversing the leaves of the subtask in the task breakdown tree. If any capability is required by more than one of the subtask's elemental sub-subtasks, the merit

factors associated with that capability are combined to result in one merit factor for each capability required by the subtask. At the beginning of the problem execution, these merits have initial values. However, as the subtasks are performed, the job planner (not addressed in this paper) can alter the merit factors as necessary after each subtask completion to reflect new knowledge about the tasks. The task allocator would then derive a new allocation based on the adjusted merit factors.

Thus, the task allocator must have information such as that shown in figure 3 concerning the capabilities required to perform a task.

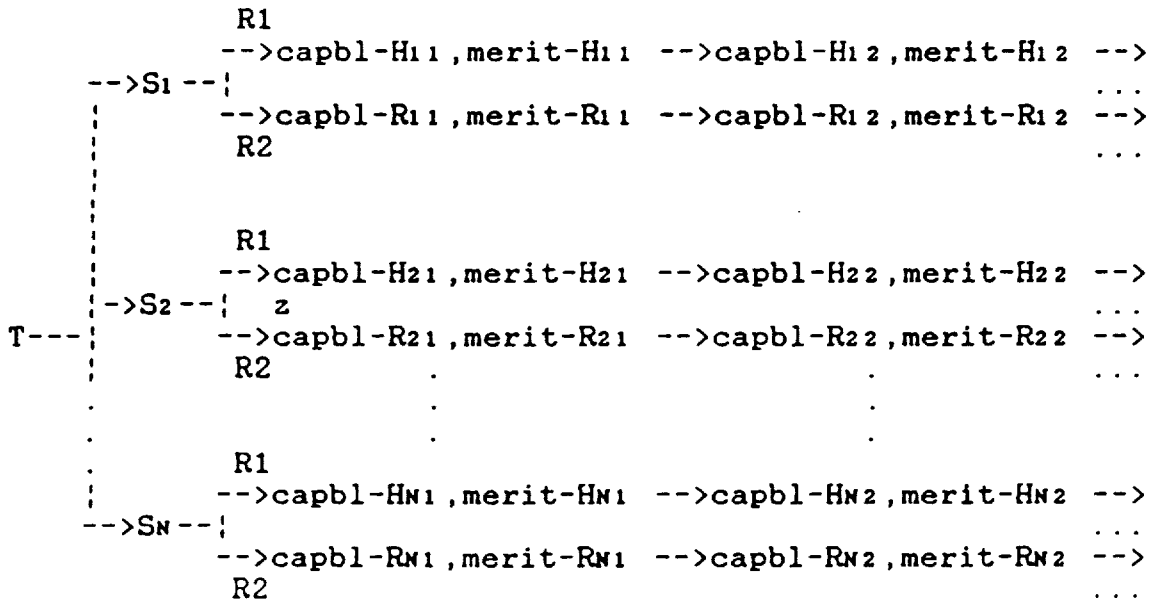


Figure 3

Figure 3 shows that task T consists of N subtasks S1 through SN. For each subtask, the task allocator knows the list of capabilities and merit factors required by each resource to perform the subtask. For example, to perform the subtask S2, the resource R1 must possess capabilities "capbl-H21", "capbl-H22", and so on, which have merit factors of "merit-H21", "merit-H22", and so on. The task allocator can then compare the list of capabilities required for a resource to perform a subtask (the task information) with the actual capabilities possessed by the resource (the resource information) to determine whether the resource is capable of performing the subtask. After completing these comparisons for both resources, the task allocator can obtain the optimal subtask allocation by determining which

resource most suitably meets the constraints/criteria of the problem, and then assigning the subtask accordingly.

Although this paper is addressing the allocation problem requiring only one manipulation subtask to be executed at a time (a sequential-task problem), the extension to several machines and multitasking could be possible with this methodology by incorporating into the task allocator the ability to handle information such as precedence constraints among the subtasks.

3.1.4 ENVIRONMENT

In order to satisfy the constraints and criteria of the problem, the task allocator may often need to have access to information about the environment. The details to be contained in the environmental knowledge base must include information on what is in the environment, what the environment looks like, and how the environment behaves. In addition, the presence of certain environmental conditions may activate certain new constraints/criteria which the task allocator must address.

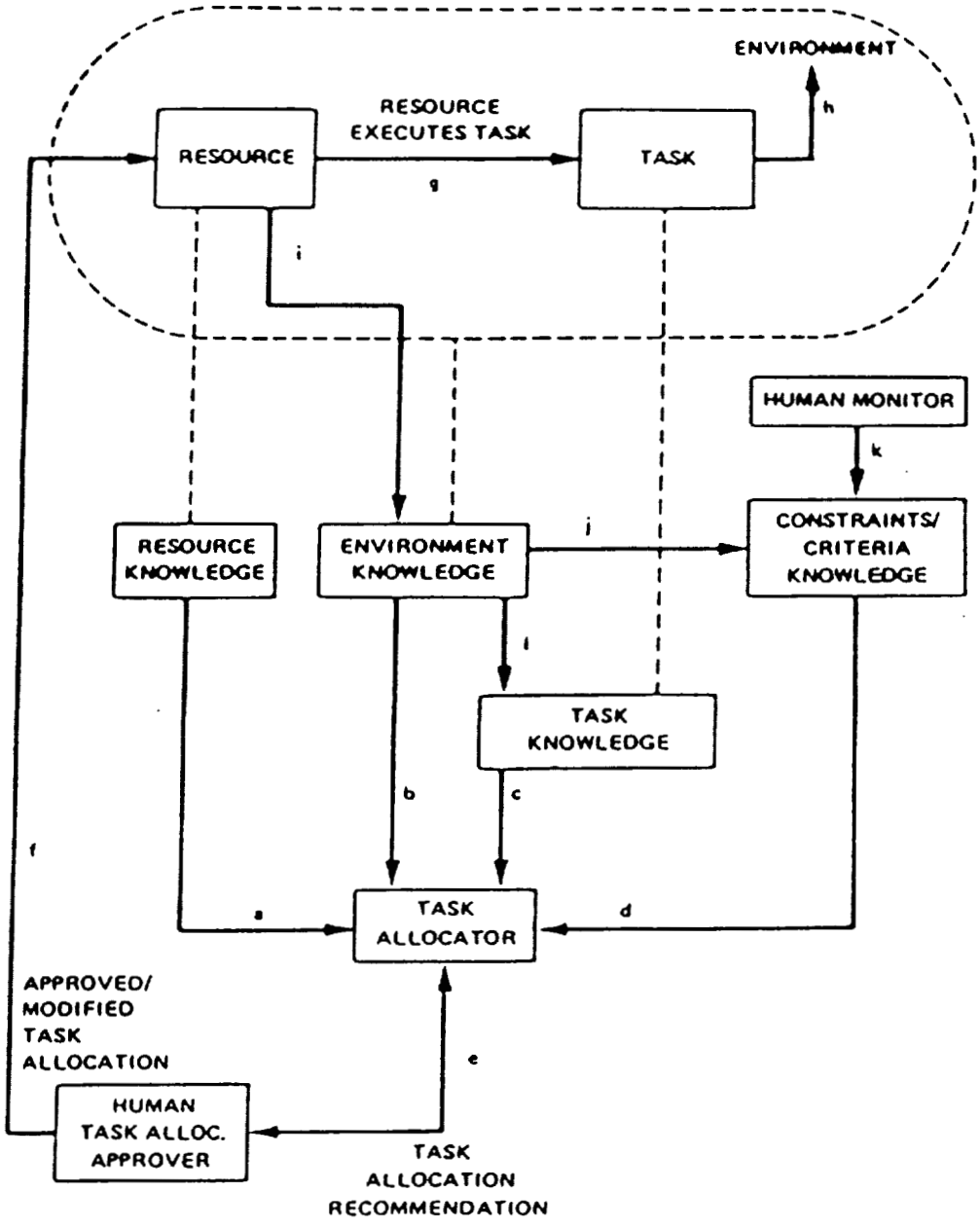
The environmental information will also be accessed by the resources to help them function effectively in their environment. For example, there may be obstacles to avoid or tools available for use in performing a subtask. If the robot were told to GET WRENCH, it must know what a wrench looks like and possibly have an idea of where to find it.

Of course, the human could conclude many things about the environment by simply observing it. However, the computer must operate with an automated representation of its environment. The specific representation of the environment is highly dependent on the application and would thus vary accordingly. Possible representations include frames, rules, scripts, and nets.

3.2 FLOW OF EXECUTION

The current information about the constraints/criteria, resources, tasks, and environment will be stored in separate computerized knowledge bases, and will be shared among all the entities which need the information. These knowledge bases will be kept current by the use of sensors which monitor the resources, the environment, and the tasks, or they could be directly updated by the resources. In order for the man-machine symbiotic system to work effectively, it is important that the knowledge areas be able to interact. Figure 4 depicts the relationship between the knowledge areas.

Fig. 4: PRIMARY INTERACTIONS IN A DYNAMIC TASK ALLOCATION PROBLEM



In figure 4 the dotted oval indicates the actual environment. The three double-dotted lines connecting the resource and the resource knowledge, the environment and the environmental knowledge, and the task and the task knowledge indicate a close association between the physical entities (resource, environment, task) and the knowledge of the entities. The information which can be obtained from either the physical entities or from the knowledge of the entities should be the same.

Figure 4 shows that the task allocator uses knowledge about the resources, environment, tasks, and constraints/criteria (links a, b, c, d) to make a task allocation recommendation. If necessary, the human task allocation approver may change this task allocation (link e). Note that although it is possible that the human task allocation approver is the same person who performs the subtasks, this does not necessarily have to be true. The resource is then assigned a subtask according to the approved/modified allocation (link f). As the resource executes the subtask (link g), the changing subtask status in itself modifies the environment (link h). Possibly, the resource will notice additional events or changes in the environment and will update the environmental knowledge directly (link i). As the environment changes, the constraints/criteria may need to be changed automatically to reflect the new conditions (link j), or manually by a human who monitors the problem execution (link k). Again, the human monitor need not necessarily be the same human who performs the subtasks or who approves the task allocation. Additionally, the list of subtasks to be performed might need to be altered because of environmental modifications (link l). Using the updated knowledge about the resources, the environment, the subtasks, and the constraints/criteria, the task allocator can replan the task allocation as necessary to repeat the cycle.

3.3 DYNAMIC NATURE OF TASK ALLOCATION

One of the key features of this task allocation methodology is its ability to be event-driven, responding to changes in the information about the constraints/criteria, resources, tasks, or the environment by altering the task allocation. Such a dynamic nature of the task allocation is essential to allow the man-machine symbiont to cope with a changing work context. The dynamic nature of the task allocator is directly related to the information in the knowledge bases. If the information in the knowledge bases never changed, the task allocation would never change. However, in a real-world problem, the information in each of

the knowledge bases will be undergoing continual changes to reflect the true state of the problem and the accumulation of experience. The following paragraphs explain how each of the knowledge bases can change.

First of all, although the constraints/criteria are initially set for a particular application, dynamic changes in the work context or environment may cause the constraints/criteria to be changed. The knowledge base changes can be made directly by some type of sensor, or they can be modified manually by a human. For example, the human might decide to change the effective constraint from "minimize time of task completion" to "minimize human involvement" after experiencing fatigue following a long series of manipulation tasks. The task allocator would then allocate the subtasks by attempting to assign as few subtasks as possible to the human.

Secondly, as the resources execute the subtasks, the level of achievement factors and the timeliness-of-achievement factors for their capabilities may change, reflecting new knowledge about the resources. Such changes can take place in two ways: through a learning scheme and through monitoring of the resources. The learning scheme (discussed in a companion paper) allows the robot to learn and improve its capabilities by observing the human. For example, suppose the subtask to be allocated is FIND WRENCH. Initially, the robot will not know what a wrench looks like, indicated by a level of achievement factor of zero or "unknown" for the capability RECOGNIZE WRENCH. The task allocator will therefore assign the subtask to the human, who is then observed by the robot as he performs the subtask. In observing the human, the robot learns what a wrench looks like, and its level of achievement factor is upgraded accordingly. The allocation of the next subtask requiring the ability to recognize a wrench will take into account the new capability factors and will possibly result in a new allocation.

The second method in which the level of achievement factors and the timeliness of achievement factors can change is through monitoring of the resources. It is very important that the knowledge of the resources be consistent with the actual resources themselves. To accomplish this, some type of monitor must observe and quantify the resource's performance to determine if there is a proper correlation between the resource and the knowledge about the resource. If not, the resource knowledge base must be corrected. For example, if the human has a level-of-achievement factor of 7 for the capability MANIPULATION, but does not perform at that level after several hours of work (possibly due to fatigue or boredom), the factor should be

appropriately updated in the knowledge base for use in future subtask allocations.

The information in the third knowledge base, the task information, is subject to change during the execution of the subtasks when environmental changes occur which require the job planner to update the list of subtasks to be performed. The task allocator should recognize these changes and be able to replan the task allocation appropriately. For example, if the event WRENCH DROPPED occurred, the subtask sequence would be reconfigured by the job planner to include the subtask PICK UP WRENCH. The task allocator should then respond to this event and reallocate the subtasks to reflect the change.

Finally, the fourth knowledge base, the environmental information, must be dynamic to allow for changes in the environment, such as successful subtask completion, and for unexpected events, such as subtask failure, to be detected. The changes to the environmental knowledge could come from information supplied directly by the resources, or from sensors separate from the resources. This dynamic feature is important to allow the task allocator to recognize the need for re-allocation of subtasks due to changes in the environment.

4.0 CONCLUSION

A methodological approach for dynamically allocating tasks to a human and an intelligent machine involved in a man-machine symbiotic system has been presented. The necessary knowledge areas and flow of execution have been outlined, and the proposed architecture has been shown to allow dynamic response and task reallocation due to changes in the work constraints, physical environment, and capabilities of the human and the machine, as well as to unanticipated events and human requests or controls. Major man-machine task allocation issues such as event-driven dynamics, knowledge updating through observation and learning, and performance-based work distribution have been discussed. Although this methodology was designed in the context of a remote-manipulation system involving only two symbiotic partners sharing control of a single manipulator arm to accomplish a series of sequential tasks, the methodology has been shown to have the potential for being extended to systems including more than two partners, multitasking operations, or multi-constraint situations. The architecture has been designed to be fully compatible with learning schemes and job-planning methodologies and future work will include the addition of automated monitoring, automated learning, and job planning modules to the current system.

REFERENCES

[1] Chu, Y., W.B. Rouse, "Adaptive Allocation of Decisionmaking Responsibility Between Human and Computer in Multitask Situations", IEEE Trans. Syst., Man, Cybern., Vol. SMC-9, No. 12, pp. 769-778, 1979.

[2] Hamel, W.R., Jorgensen, C.C., Weisbin, C.R., "Man-Robot Symbiosis: Schemes for the Evolution of Autonomous Systems", ORNL/TM-10396 (in process).

[3] Jorgensen, C.C., "Neural Network Recognition of Robot Sensor Graphs using Hypercube Computers", Second Conference on Hypercube Multiprocessors, Sept. 29-Oct. 1, 1986, Knoxville, TN.

[4] Jorgensen, C.C., "Neural Network Representation of Sensor Graphs for Autonomous Robot Navigation", to be presented IEEE International Conference on Neural Networks, June 21 - 24, 1987, San Diego, CA.

[5] Jorgensen, C.C., C. Matheus, "Catching Knowledge in Neural Nets", AI Expert, Vol. 1, December 1986.

[6] Licklider, J.C.R., "Man-Computer Symbiosis", IRE Trans. on Human Factors in Electronics, Vol. HFE-1, pp. 4-11, 1960.

[7] Parker, L.E., F.G. Pin, "Dynamic Task Allocation for a Man-Machine Symbiosis System", ORNL/TM-10397 (in press).

[8] Revesman, M.E., J.S. Greenstein, "Application of a Mathematical Model of Human Decisionmaking for Human-Computer Communication", IEEE Trans. Syst., Man, Cybern, Vol. SMC-16, No. 1, pp. 142-147, 1986.

[9] Rieger, C.A., J.S. Greenstein, "The Allocation of Tasks Between the Human and Computer in Automated Systems", Proceedings of IEEE 1982 International Conference on Cybernetics and Society, pp. 204-208.

[10] Rouse, W.B., "Human-Computer Interaction in the Control of Dynamic Systems", Computing Surveys, Vol. 13, No. 1, pp. 71-99, 1979.

[11] Rouse, W.B., "Human-Computer Interaction in Multitask Situations", IEEE Trans. Syst., Man, Cybern., Vol. SMC-7, No. 5, pp. 384-392, 1977.

NASREM -- STANDARD REFERENCE MODEL FOR TELEROBOT CONTROL

J. S. Albus, R. Lumia, and H. McCain
Robot Systems Division
Metrology Building Room B-124
National Bureau of Standards
Gaithersburg, MD. 20899

ABSTRACT

A hierarchical architecture is described which supports space station telerobots in a variety of modes. The system is divided into three hierarchies: task decomposition, world model, and sensory processing. Goals at each level of the task decomposition hierarchy are divided both spatially and temporally into simpler commands for the next lower level. This decomposition is repeated until, at the lowest level, the drive signals to the robot actuators are generated. To accomplish its goals, task decomposition modules must often use information stored in the world model. The purpose of the sensory system is to update the world model as rapidly as possible to keep the model in registration with the physical world. This paper describes the architecture of the entire control system hierarchy and how it can be applied to space telerobot applications.

1. INTRODUCTION

One of the major directions on which the robot research community has concentrated its efforts is concerned with planning and controlling motion. Given a specific task, a motion plan must be calculated which meets the task requirements. Then, the plan must be executed; there must be sufficient control for the robot to adequately effect the desired motion.

Trajectories are often planned as straight lines in Cartesian space [1]. Whitney [2,3] developed the resolved motion rate control method for Cartesian straight line motions. Paul [4,5,6] used homogeneous coordinate transformations to describe a trajectory as a function of time, and Taylor [7] used coordinated joint control over small segments to keep the trajectory within a specified deviation of the desired straight line trajectory.

While the research described above employs a "kinematic" approach to robot control, another direction of research takes the manipulator "dynamics" into account in the description of robot motion. The dynamic equations of motion are described either by the Lagrangian formulation [8] or by the Newton-Euler equations [9]. Algorithms and computer architectures have been suggested which promise real-time dynamic robot control [10,11].

Another aspect of motion control is concerned with the variables being controlled. The research described to this point was concerned primarily with position control. The robot moved from an initial position to a goal position. While this is perhaps the most common mode, there are many applications for robots which suggest that other variables should be controlled. For example, force control would be desired for assembly operations. Raibert and Craig [12] suggest a method for hybrid position/force control of manipulators.

These examples point to the more general problem of sensory processing. For a great deal of robot motion research, sensory processing has been limited to joint positions, velocities, and accelerations. However, other sensors are often required to accomplish tasks. The control community has concentrated on the control aspects of the robot and as a result, little emphasis has been placed on sophisticated sensory processing.

Machine vision, an offshoot of image processing research, has recently been associated with advanced robot applications. One of the most interesting directions in this research area is concerned with sensor controlled robots. Operating with the constraints imposed by real-time robot control, early methods used structured light and binary images [13,14,15,16]. These approaches, though developed at different institutions, shared many concepts. One of the important subsequent research efforts went toward the development of model-based image processing. Bolles and Cain [17] used models of objects to guide the algorithms in a hypothesis/verification scheme known as the

local feature focus method. The concept has recently been extended from two dimensional (i.e. nearly flat) objects to three dimensional objects [18]. Although the approaches described here have led to a better understanding of real-time vision processing, the systems lacked a sophisticated interconnection with the robot control system.

The Automated Manufacturing Research Facility (AMRF), developed at the National Bureau of Standards, is a hierarchically organized small-batch metal machining shop [19]. It separates sensory processing and robot control by a sophisticated world model. The world model has three complementary data representations. Lumia [20] describes the CAD-like section of the model. Shneier, Kent, and Mansbach [21] describe the octree and table representations supported by the model. The model generates hypotheses for the features which are either verified or refuted by empirical evidence. The sensory system's task is to update the appropriate parts of the world model with new or revised data as rapidly as possible. The control system accesses the world model as desired to obtain the current best guess concerning any aspect of the world. Shneier, Lumia, and Kent [22] describe the sensory system and its operation in greater detail. The AMRF was the first deliberate attempt to tie together sensory processing, world modeling, and robot control in a generic fashion. The system developed for the AMRF is applicable to more than manufacturing. This paper describes its use in space telerobotics.

2. A FUNCTIONAL SYSTEM ARCHITECTURE

The fundamental paradigm is shown in Figure 1. The control system architecture is a three legged hierarchy of computing modules, serviced by a communications system and a common memory. The task decomposition modules perform real-time planning and task monitoring functions, and decompose task goals both spatially and temporally. The sensory processing modules filter, correlate, detect, and integrate sensory information over both space and time in order to recognize and measure patterns, features, objects,

events, and relationships in the external world. The world modeling modules answer queries, make predictions, and compute evaluation functions on the state space defined by the information stored in common memory. Common memory is a global database which contains the system's best estimate of the state of the external world. The world modeling modules keep the common memory database current and consistent.

2.1. Task Decomposition - H modules (Plan, Execute)

The first leg of the hierarchy consists of task decomposition H modules which plan and execute the decomposition of high level goals into low level actions. Task decomposition involves both a temporal decomposition (into sequential actions along the time line) and a spatial decomposition (into concurrent actions by different subsystems). Each H module at each level consists of a job assignment manager JA, a set of planners PL(i), and a set of executors EX(i). These decompose the input task into both spatially and temporally distinct subtasks as shown in Figure 2. This will be described in greater detail in section 4.

2.2. World Modeling - M modules (Remember, Estimate, Predict, Evaluate)

The second leg of the hierarchy consists of world modeling M modules which model (i.e. remember, estimate, predict) and evaluate the state of the world. The "world model" is the system's best estimate and evaluation of the history, current state, and possible future states of the world, including the states of the system being controlled. The "world model" includes both the M modules and a knowledge base stored in a common memory database where state variables, maps, lists of objects and events, and attributes of objects and events are maintained. By this definition, the world model corresponds to what is widely known throughout the artificial intelligence community as a "blackboard" [23]. The world model performs the

following functions:

1. Maintain the common memory knowledge base by accepting information from the sensory system.

2. Provide predictions of expected sensory input to the corresponding G modules, based on the state of the task and estimates of the external world.

3. Answer "What is?" questions asked by the executors in the corresponding level H modules. The task executor can request the values of any system variable.

4. Answer "What if?" questions asked by the planners in the corresponding level H modules. The M modules predict the results of hypothesized actions.

2.3. Sensory Processing - G modules (Filter, Integrate, Detect, Measure)

The third leg of the hierarchy consists of sensory processing G modules. These recognize patterns, detect events, and filter and integrate sensory information over space and time. The G modules at each level compare world model predictions with sensory observations and compute correlation and difference functions. These are integrated over time and space so as to fuse sensory information from multiple sources over extended time intervals. Newly detected or recognized events, objects, and relationships are entered by the M modules into the world model common memory database, and objects or relationships perceived to no longer exist are removed. The G modules also contain functions which can compute confidence factors and probabilities of recognized events, and statistical estimates of stochastic state variable values.

2.4. Operator Interfaces (Control, Observe, Define Goals, Indicate Objects)

The control architecture defined here has an operator

interface at each level in the hierarchy. The operator interface provides a means by which human operators, either in the space station or on the ground, can observe and supervise the telerobot. Each level of the task decomposition hierarchy provides an interface where the human operator can assume control. The task commands into any level can be derived either from the higher level H module, or from the operator interface. Using a variety of input devices such as a joystick, mouse, trackball, light pen, keyboard, voice input, etc., a human operator can enter the control hierarchy at any level, at any time of his choosing, to monitor a process, to insert information, to interrupt automatic operation and take control of the task being performed, or to apply human intelligence to sensory processing or world modeling functions.

The sharing of command input between human and autonomous control need not be all or none. It is possible in many cases for the human and the automatic controllers to simultaneously share control of a telerobot system. For example a human might control the orientation of a camera while the robot automatically translates the same camera through space.

2.4.1 Operator Control interface levels

The operator can enter the hierarchy at any level. The operator control interface interprets teleoperation in the fullest sense: a teleoperator is any device which is controlled by a human from a remote location. While the master-slave paradigm is certainly a type of teleoperation, it does not constitute the only form of man-machine interaction. At different levels of the hierarchy, the interface device for the human may change but the fundamental concept of teleoperation is still preserved. Table 1 illustrates the interaction an operator may have at each level.

The operator control interface thus provides mechanisms for entering new instructions or programs into the various control modules. This can be used on-line for real-time

supervisory control, or in a background mode for altering autonomous telerobot plans before autonomous execution reaches that part of the plan.

2.4.2 Operator monitoring interfaces

The operator interfaces allow the human the option of simply monitoring any level. Windows into the common memory knowledge base permit viewing of maps of service bay layout, geometric descriptions and mechanical and electrical configurations of satellites, lists of recognized objects and events, object parameters, and state variables such as positions, velocities, forces, confidence levels, tolerances, traces of past history, plans for future actions, and current priorities and utility function values. These may be displayed in graphical form, for example using dials or bar graphs for scalar variables, shaded graphics for object geometry, and a variety of map displays for spatial occupancy.

2.4.3 Sensory processing/world modeling interfaces

The operator interface may also permit interaction with the sensory processing and/or world modeling modules. For example, an operator using a video monitor with a graphics overlay and a light pen or joystick might provide human interpretative assistance to the vision/world modeling system. The operator might interactively assist the model matching algorithms by indicating with a light pen which features in the image (e.g. edges, corners) correspond to those in a stored model. Alternatively, an operator could use a joystick to line up a wireframe model with a TV image, either in 2-D or 3-D. The operator might either move the wireframe model so as to line up with the image, or move the camera position so as to line up the image with the model. Once the alignment was nearly correct, the operator could allow automatic matching algorithms to complete the match, and track future movements of the image.

2.5. Common Memory

2.5.1. Communications

One of the primary functions of common memory is to facilitate communications between modules. Communications within the control hierarchy is supported by a common memory in which state variables are globally defined.

Each module in the sensory processing, world modeling, and task decomposition hierarchies reads inputs from, and writes outputs to, the common memory. Thus each module needs only to know where in common memory its input variables are stored, and where in common memory it should write its output variables. The data structures in the common memory then define the interfaces between the G, M, and H modules.

The operator interfaces also interact with the system through common memory. The operator displays simply read the variables they need from the locations in common memory. If the operator wishes to take control of the system, he simply writes command variables to the appropriate locations in common memory. The control modules that read from those locations need not know whether their input commands derived from a human operator, or from the next higher level in the autonomous control hierarchy.

2.5.2 State Variables

The state variables in common memory are the system's best estimate of the state of the world, including both the external environment and the internal state of the H, M, and G modules. Data in common memory are available to all modules at all levels of the control system.

The knowledge base in the common memory consists of three elements: maps which describe the spatial occupancy of the world, object-attribute linked lists, and state variables.

3. LEVELS IN THE CONTROL HIERARCHY

The control system architecture described here for the Flight Telerobot System is a six level hierarchy as shown in Figure 3. At each level in this hierarchy a fundamental transformation is performed on the task.

Level 1 transforms coordinates from a convenient coordinate frame into joint coordinates. This level also servos joint positions, velocities, and forces.

Level 2 computes inertial dynamics, and generates smooth trajectories in a convenient coordinate frame.

Level 3 decomposes elementary move commands (E-moves) into strings of intermediate poses. E-moves are typically defined in terms of motion of the subsystem being controlled (i.e., transporter, manipulator, camera platform, etc.) through a space defined by a convenient coordinate system. E-move commands may consist of symbolic names of elementary movements, or may be expressed as keyframe descriptions of desired relationships to be achieved between system state variables. E-moves are decomposed into strings of intermediate poses which define motion pathways that have been checked for clearance with potential obstacles, and which avoid kinematic singularities.

Level 4 decomposes object task commands specified in terms of actions performed on objects into sequences of E-moves defined in terms of manipulator motions. Object tasks typically define actions to be performed by a single multiarmed telerobot system on one object at a time. Tasks defined in terms of actions on objects are decomposed into sequences of E-moves defined in terms of manipulator or

vehicle subsystem motions. This decomposition checks to assure that there exist motion freeways clear of obstacles between keyframe poses, and schedules coordinated activity of telerobot subsystems, such as the transporter, dual arm manipulators, multifingered grippers, and camera arms.

Level 5 decomposes actions to be performed on batches of parts into tasks performed on individual objects. It schedules the actions of one or more telerobot systems to coordinate with other machines and systems operating in the immediate vicinity. For example, Level 5 decomposes service bay action schedules into sequences of object task commands to various telerobot servicers, astronauts, and automatic berthing mechanisms. Service bay actions are typically specified in terms of servicing operations to be performed by all the systems (mechanical and human) in a service bay on a whole satellite. This decomposition typically assigns servicing tasks to various telerobot systems, and schedules servicing tasks so as to maximize the effectiveness of the service bay resources.

Level 6 decomposes the satellite servicing mission plan into service bay action commands. Mission plans are typically specified in terms of satellite servicing priorities, requirements, constraints, and mission time line. The level 6 decomposition typically assigns satellites to service bays, sets priorities for service bay activities, generates requirements for spare parts and tool kits, and schedules the activities of the service bays so as to maximize the effectiveness of the satellite servicing mission. To a large extent the level 6 mission plans will be generated off line on the ground, either by human mission planners, or by automatic or semiautomatic mission

planning methods.

4. DETAILED STRUCTURE OF THE H MODULES

The H module at each level consists of three parts as shown in Figure 4: a job assignment manager JA, one or more planners PL(s), and one or more executors EX(s).

The job assignment manager JA is responsible for partitioning the task command TC into s spatially or logically distinct jobs to be performed by s physically distinct planner/executor mechanisms. At the upper levels the job assignment module may also assign physical resources against task elements. The output of the job assignment manager is a set of job commands JC(s), s=1, 2, ..., N where N is the number of spatially, or logically, distinct jobs.

For each of these job commands JC(s), there exists a planner PL(s) and a executor EX(s). Each planner PL(s) is responsible for decomposing its job command JC(s) into a temporal sequence of planned subtasks PST(s,tt). Planning typically requires evaluation of alternative hypothetical sequences of planned subtasks. The planner hypothesizes some action or series of actions, the world model predicts the results of the action(s) and computes some evaluation function EF(s,tt) on the predicted resulting state of the world. The hypothetical sequence of actions producing the best evaluation function EF(s,tt)max is then selected as the plan PST(s,tt) to be executed by the executor EX(s).

$$PST(s,tt) = PL(s)\ [JC(s), EF(s,tt)max]$$

where tt is the time sequence index for steps in the plan. tt may also be defined as a running temporal index in planning space, tt = 1, 2, ..., th where th is the value of the tt index at the planning horizon. The planning horizon is defined as the period into the future over which a plan is prepared. Each level of the hierarchy has a planning horizon of one or two expected input task time durations.

Each executor EX(s) is responsible for successfully executing the plan PST(s,tt) prepared by its respective planner PL(s). If all the subtasks in the plan PST(s,tt) are successfully executed, then the goal of the original task will be achieved. The executor operates by selecting a subtask from the current queue of planned subtasks and outputting a subcommand STX(s,t) to the appropriate subordinate H module at time t. The EX(s) module monitors its feedback FB(s,t) input in order to servo its output STX(s,t) to the desired subtask activity.

$$STX(s,t+n) = EX(s)\ [PST(s,t), FB(s,t)]$$

where n = the number of state clock periods required to compute the function EX(s). n typically equals 1. The feedback FB(s,t) also carries timing and subgoal event information for coordination of output between executors at the same level. When the executor detects a subgoal event, it selects the next planned subtask from the queue.

Executor output STX(s,t) also contains requests for information from the world model M module, and status reports to the next higher (i+1) level in the H module hierarchy. The feedback FB(s,t) contains status reports from the H module at the i-1 th level indicating progress on its current task.

5. CONCLUSION

This paper has described a hierarchically organized control system and has shown how this generic system can be applied to telerobotic applications in space by considering the requirements of a flight telerobotic servicer for the space station.

REFERENCES

[1] M. Brady, et.al., ed. Robot Motion: Planning and Control, (Cambridge, MIT Press, 1982).

[2] D.E. Whitney, "Resolved Motion Rate Control of Manipulators and Human Prostheses," IEEE Trans. Man-Machine Systems MMS-10, 1969, p. 47.

[3] D.E. Whitney, "The Mathematics of Coordinated Control of Prostheses and Manipulators," Journal of Dynamic Systems, Measurement, Control, Dec. 1972, p. 303.

[4] R.P. Paul, "Manipulator Path Control," IEEE Int. Conf. on Cybernetics and Society, New York, p. 147.

[5] R.P. Paul, "Manipulator Cartesian Path Control," IEEE Trans. Systems, Man, Cybernetics SMC-9, 1979, p. 702.

[6] R.P. Paul, Robot Manipulators: Mathematics, Programming, and Control, (Cambridge, MIT Press, 1981.)

[7] R. H. Taylor, "Planning and Execution of Straight-line Manipulator Trajectories," IBM J. Research and Development 23 1979, p. 424.

[8] J.M. Hollerbach, "A Recursive Formulation of Lagrangian Manipulator Dynamics," IEEE Trans. Systems, Man, Cybernetics SMC-10, 11, 1980, p. 730.

[9] J.Y.S. Luh, M.W. Walker, and R.P.C. Paul, "On-line Computational Scheme for Mechanical Manipulators," J. Dynamic Systems, Measurement, Control, 102, 1980, p. 69.

[10] C.S.G. Lee, P.R. Chang, "Efficient Parallel Algorithm for Robot Inverse Dynamics Computation," IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-16, No. 4, July/August 1986, p. 532.

[11] E.E. Binder, J.H. Herzog, "Distributed Computer Architecture and Fast Parallel Algorithm in Real-Time Robot Control," IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-16, No. 4, July/August 1986, p. 543.

[12] M.H. Raibert and J.J. Craig, "Hybrid position/force

control of manipulators," J. Dynamic Systems, Measurement, Control, June, 1981, p. 126.

[13] W.A. Perkins, "A Model Based Vision System for Industrial Parts," IEEE Trans. on Computers, Vol. C-27, 1978, p. 126.

[14] G.L. Gleason, G.J. Agin, "A Modular Vision System for Sensor-controlled Manipulation and Inspection," Proc. 9th Int. Symposium on Industrial Robots, 1979, p. 57.

[15] M.R. Ward, et.al., "CONSIGHT An Adaptive Robot with Vision," Robotics Today, 1979, p. 26.

[16] J. Albus, E. Kent, M. Nashman, P. Mansbach, L. Palombo, M.O. Shneier, "Six Dimensional Vision System," SPIE, Vol. 336, Robot Vision, 1982, p. 142.

[17] R.C. Bolles, R.A. Cain, "Recognizing and Locating Partially Visible Objects: The Local Feature-Focus Method," Int. Journal of Robotics Research, Vol. 1, 1982, p. 57.

[18] R.C. Bolles, P. Horaud, M.J. Hannah, "3DPO: Three Dimensional Parts Orientation System," Proc. of The Int. Joint Conf. on Artificial Intelligence, August 1983, p. 1116.

[19] J.A. Simpson, R.J. Hocken, J.S. Albus, "The Automated Manufacturing Research Facility of the National Bureau of Standards," Journal of Manufacturing System, Vol. 1, No. 1, 1983, p. 17.

[20] R. Lumia, "Representing Solids for a Real-Time Robot Sensory System," Proc. Prolamat 1985, Paris, June 1985.

[21] M.O. Shneier, E.W. Kent, P. Mansbach, "Representing Workspace and Model Knowledge for a Robot with Mobile Sensors," Proc. 7th Int. Conf. Pattern Recognition, 1984, p. 199.

[22] M.O. Shneier, R. Lumia, E.W. Kent, "Model Based Strategies for High-Level Robot Vision," CVGIP, Vol. 33, 1986, p. 293.

[23] A. Barr, E. Feigenbaum, The Handbook of Artificial Intelligence, (Los Altos, William Kaufman, 1981).

TABLE 1 -- OPERATOR INTERACTION AT EACH LEVEL

LEVEL	TYPE OF INTERACTION
At the servo	replica master, individual joint position, rate, or force controllers.
above level 1	joy stick to perform resolved motion force/rate control
above level 2	indicate safe motion pathways. Robot computes dynamically efficient movements
above level 3	graphically or symbolically define key poses. menus to choose elemental moves.
above level 4	specify tasks to be performed on objects.
above level 5	reassign telerobots to different service bays. insert, modify, and monitor plans describing servicing task sequences.
above level 6	reconfigure servicing mission priorities.

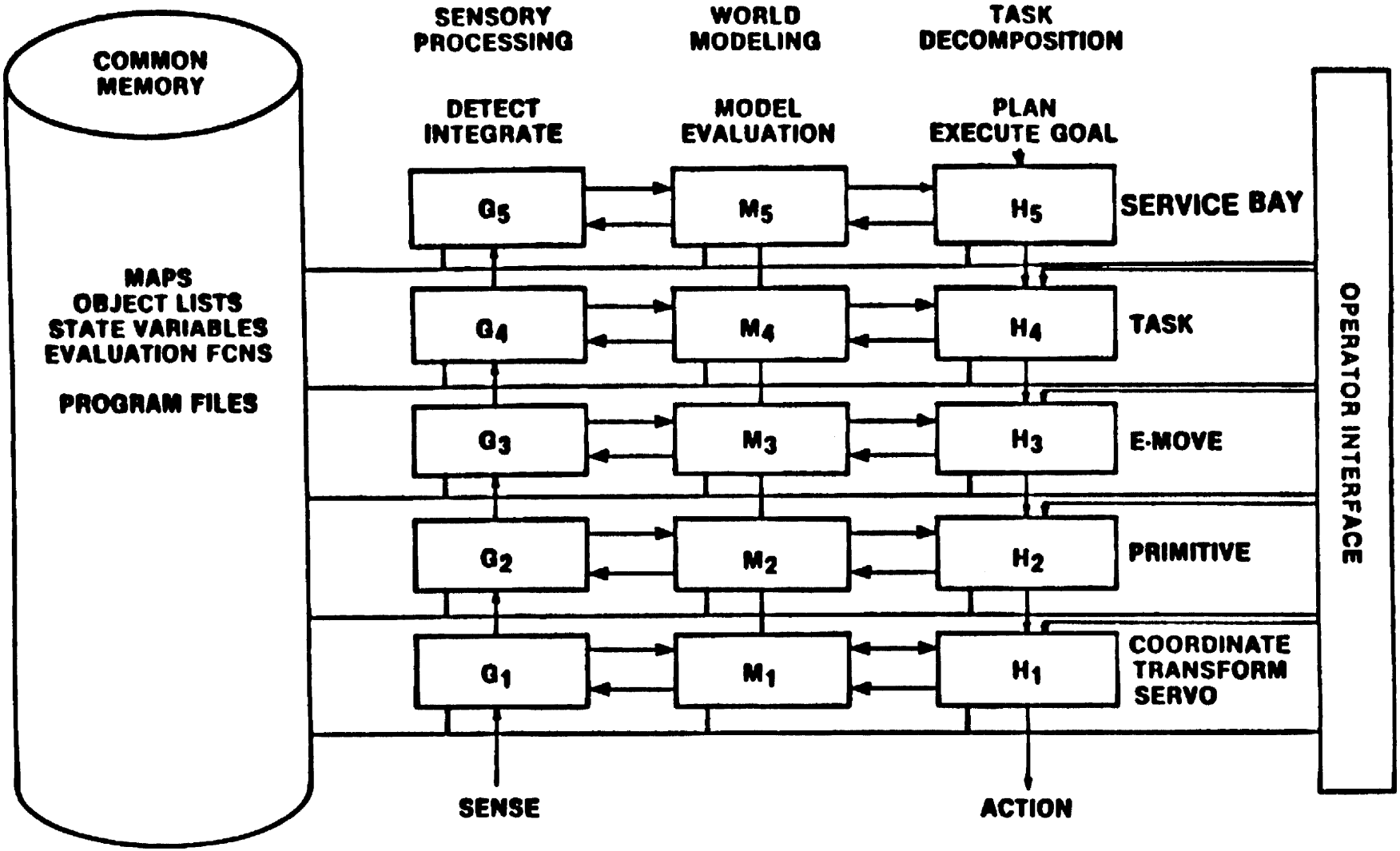


FIGURE 1 : A Hierarchical Control System Architecture for Intelligent Vehicles

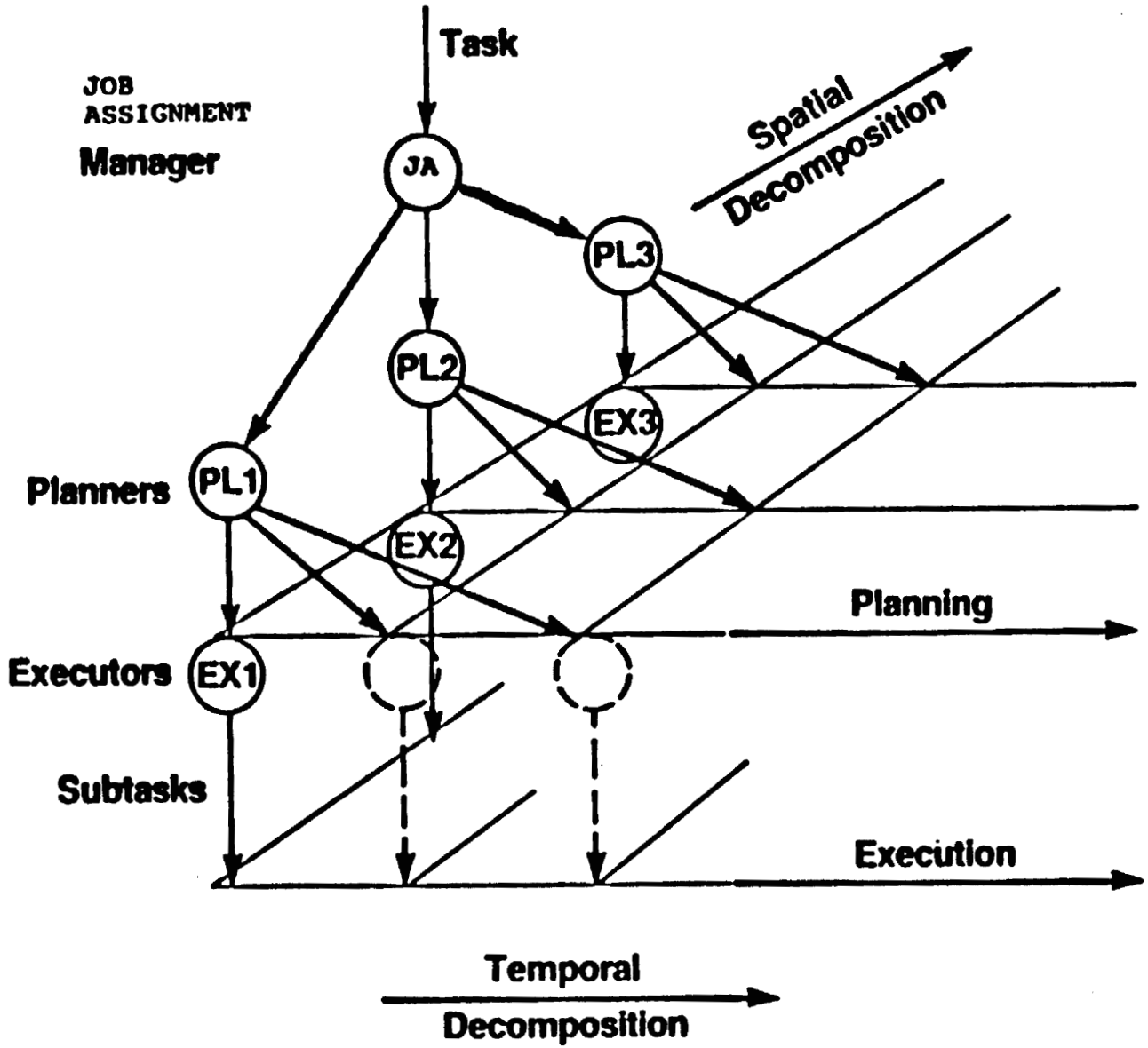


FIGURE 2: The job assignement JA performs a spatial decomposition of the task. The planners Pl (j) and executors EX (j) perform a temporal decomposition

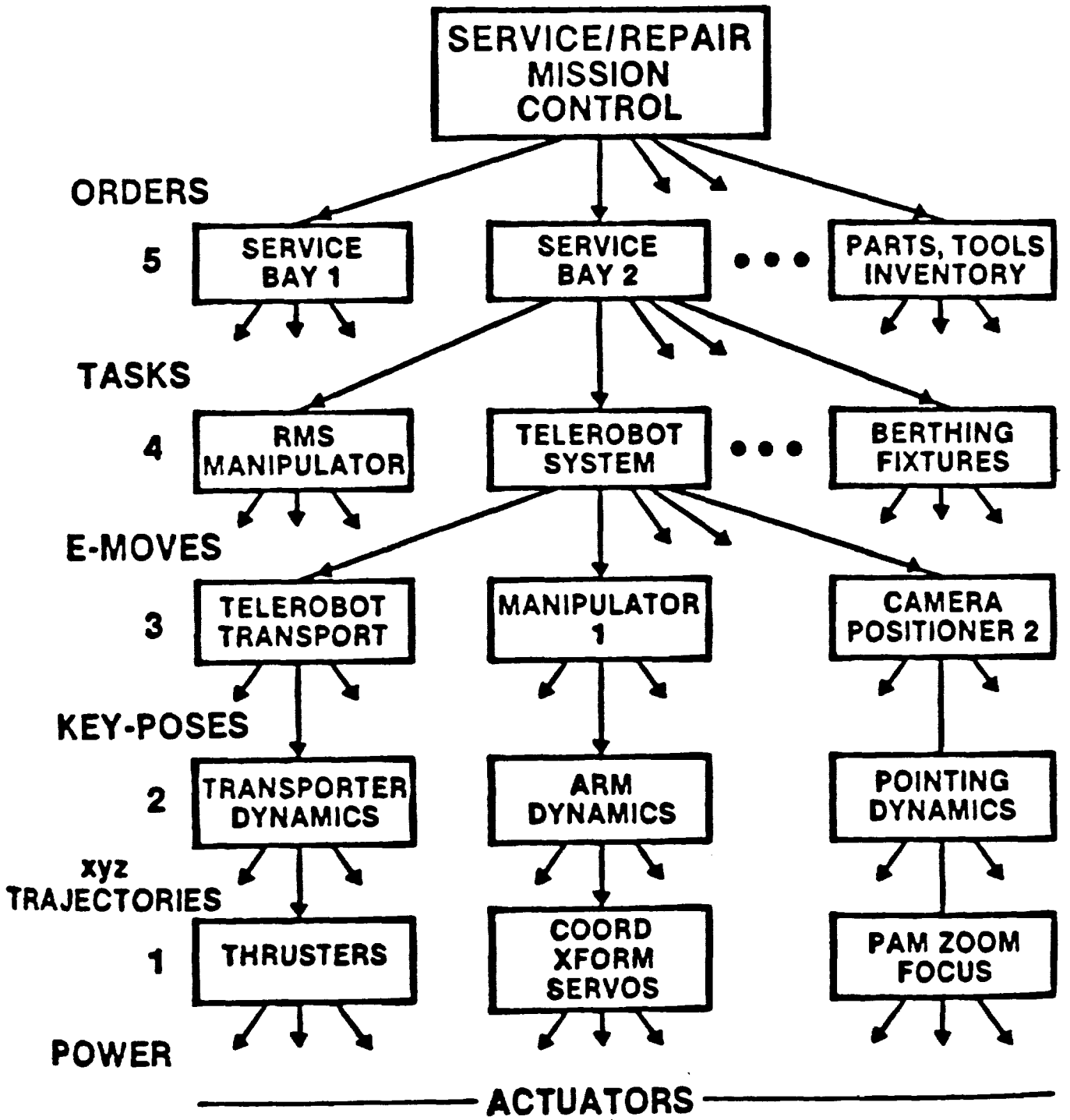


FIGURE 3 : A six level Hierarchical Control System Proposed for Multiple Autonomous Vehicles

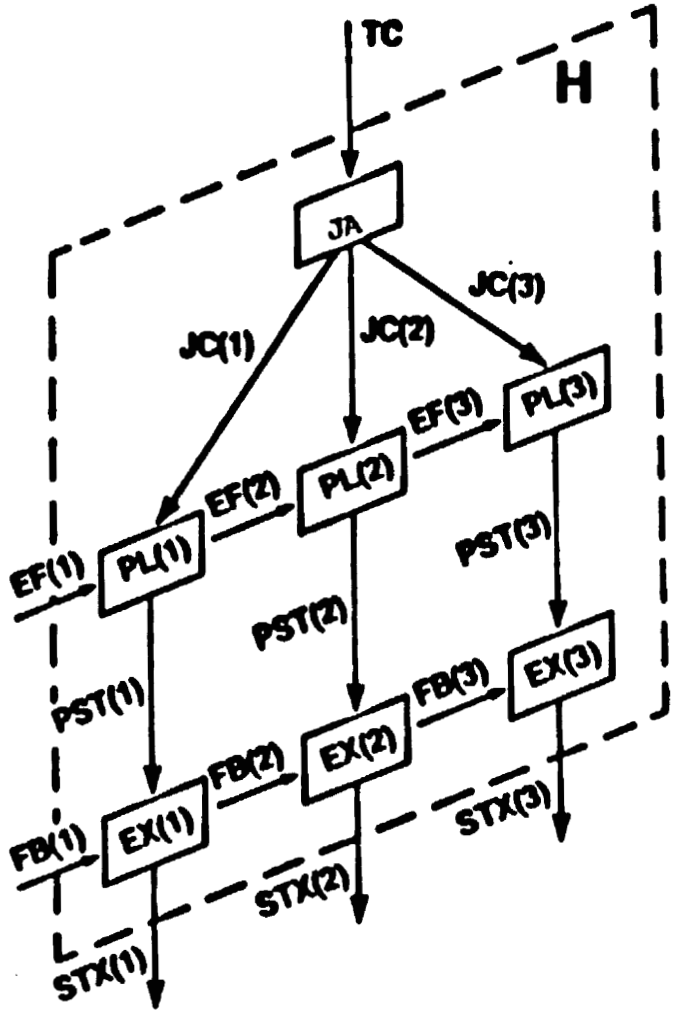


FIGURE 4 : THE H MODULE AT EACH LEVEL HAS THREE PARTS: A JOB ASSIGNMENT MODULE JA, PLANNERS PL AND A SET OF EXECUTORS EX.

KINEMATIC STUDY OF FLIGHT TELEROBOTIC SERVICER CONFIGURATION ISSUES

R.H. Lewis, R.D. Scott and W.S. Howard
RCA Astro Space Division
Princeton, NJ 08543-0800

Abstract

Several factors, such as body size and shape, and the number of arms and their placement, will influence how well the Flight Telerobotic Servicer (FTS) is suited to its potential duties for the Space Station Program. In order to examine the implications of these configuration options, eight specific 2, 3, and 4 armed FTS configurations were simulated and used to perform a Space Station Orbital Replacement Unit (ORU) exchange. The strengths and weaknesses of each configuration were evaluated. Although most of the configurations examined were able to perform the exchange, several of the 3 and 4 arm configurations had operational advantages. The results obtained from these simulations are specific to the assumptions associated with the ORU exchange scenario examined. However, they do illustrate the general interrelationships and sensitivities which need to be understood.

Introduction

The Flight Telerobotic Servicer (FTS) is intended to provide a supplemental EVA capability on the Space Station without the extra risks associated with actual manned EVA. It can be remotely controlled from within the Station's pressurized volume as an IVA activity. A number of tasks are under consideration as candidates for regular FTS assignments. The exchange of Orbital Replacement Units (ORU)s is one such work assigment. ORUs are standardized modules which contain replaceable Station and instrument subsystem elements. These ORUs will be located throughout the Space Station's trusswork on Station Interface Adapter (SIA) pallets. Transportation of the FTS around the Space Station will be provided by the Canadian Mobile Servicing Centre (MSC), which will carry the FTS at the end of one of its remote manipulator arms. The MSC arm also provides coarse positioning for the FTS over its local work area.

Assumptions

Before starting the FTS arm configuration kinematic study it was necessary to make a number of initial assumptions. In all exchanges, one arm is dedicated to providing stabilization by grasping the stability fixtures, located on the MSC and SIA, and the remaining arms move and operate the ORU. The exchange of a Work Package 4 electrical power

systwas baselined for all FTS configurations examined. These ORUssumed to be 23"x25"x12". Two versions of these ORUs were modr the two and three armed FTS configurations, the standard attadevice was assumed to incorporate a combined handle and attachanism. This system, based on RCA designs, requires only one with an appropriate end effector to both hold and operate the connsconnection mechanism. For the four arm FTS confi; an alternate version of the ORU was employed. Its attachanism consisted of two bolts which required one arm equip a special end effector to operate while a second arm equip another end effector was used to hold the ORU handle. Simusing this ORU required a minimum of three arms to compe ORUs were assumed to be mounted in a three by three array with arances between adjacent ORUs. The ORU to be replaced in all thions was the central ORU in the array. This represented the mult exchange in terms of reach and clearance. The SIA was locate of a standard, 16.4 foot (5 meters) on a side, truss cube. A generwith one six degree of freedom, 35.3 foot long manipulator arm wmed. An illustration of this common trade study enviro shown in Figure 1.

ational reach of each FTS configuration is a function of the manipngth, number of manipulators involved, and the size, shape and lahe body used. Rather than model several different manipms, only one design was used and all arms were identical. This puseful redundancy and flexibility during simulations. Specid effectors were assumed to be available, but their actual designot modeled in the simulations. A generic stub, 14 inches long, n after the wrist joint represented this class of device. The separaance between the shoulders of the FTS was an important study er. Given identical arms, a greater shoulder separation distanc allow the FTS a greater reach.

Derivements

Trs influence the size and shape of the FTS in a direct way. The rec that the FTS must be IVA servicable implies that the entire Fst have overall dimensions which are compatible with the Space hatches and passageways. The other factor is the

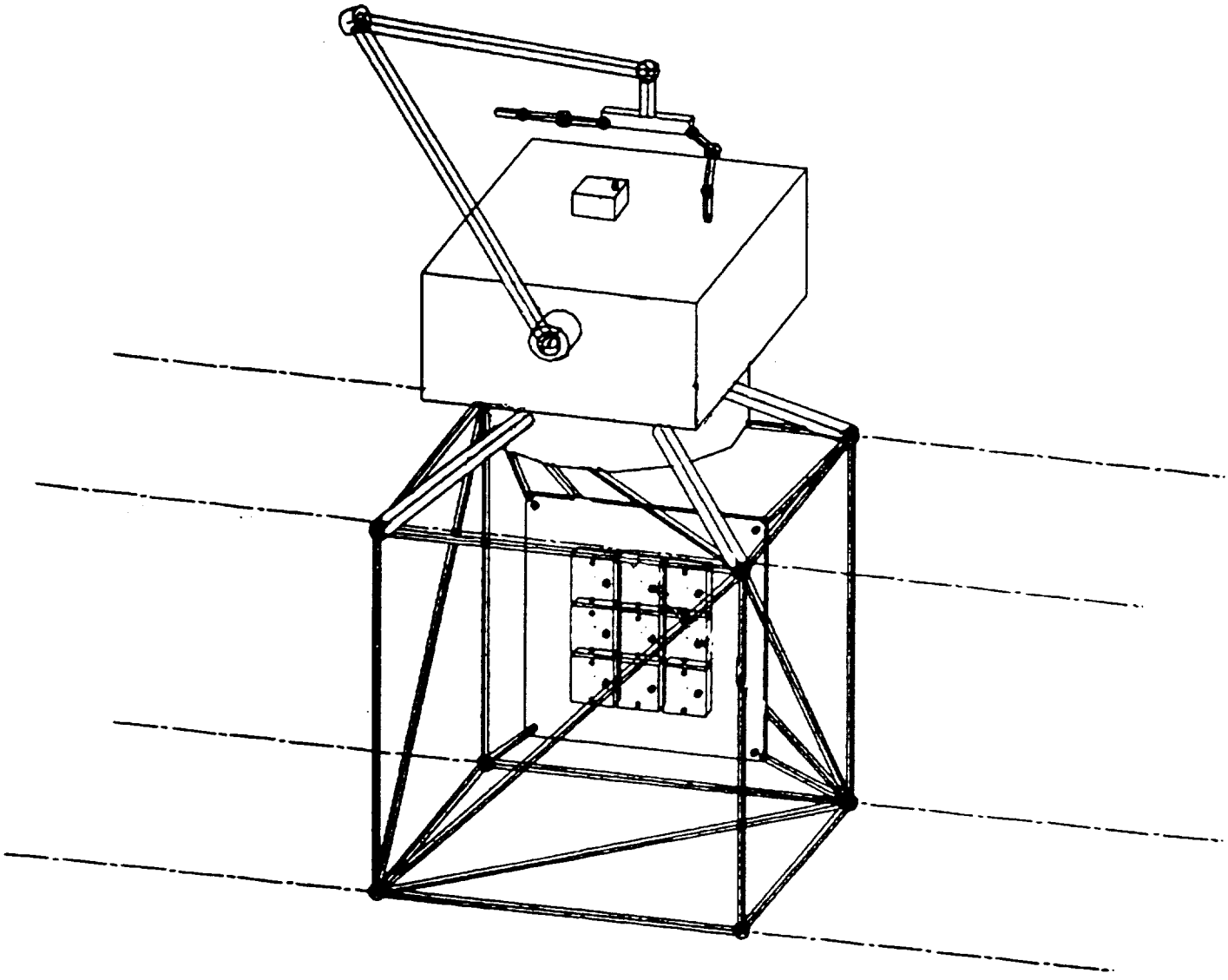


Figure 1. Common Trade Study Work Environment

requirement that the FTS is not responsible for its own transport. Because the MSC transport system positions the FTS in its proper work space, the FTS need not be capable of simultaneously reaching the five meter distance from one truss node to the next. (Physical contact with the truss members under normal conditions is not allowed). Elimination of the need for a five meter reach promotes compatibility with the hatch size requirement. All of the FTS configurations examined will fit through a Space Station hatch as it is currently known.

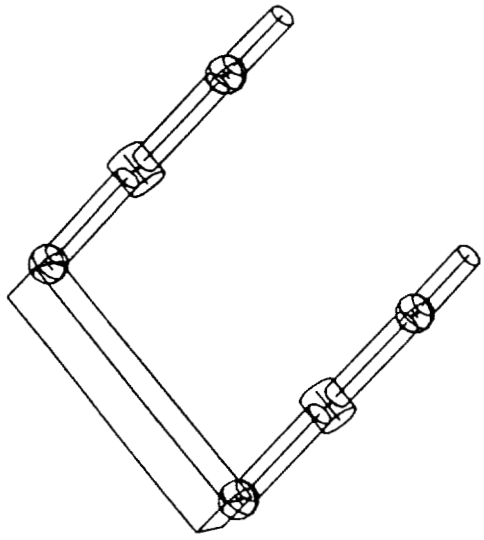
Arm Configuration Descriptions

Every FTS configuration examined used identical six degree of freedom manipulator arms as a common component. When necessary a seventh degree of freedom could be added. Although the arms were less agile than a human arm, they were of anthropomorphic design. The shoulder provided pitch and yaw, the elbow provided pitch, and the wrist provided pitch, roll, and yaw. The shoulder and elbow position the wrist, and the wrist orients the end effector relative to the work area. The arm is made up of a 22 inch upper arm and a 22 inch forearm.

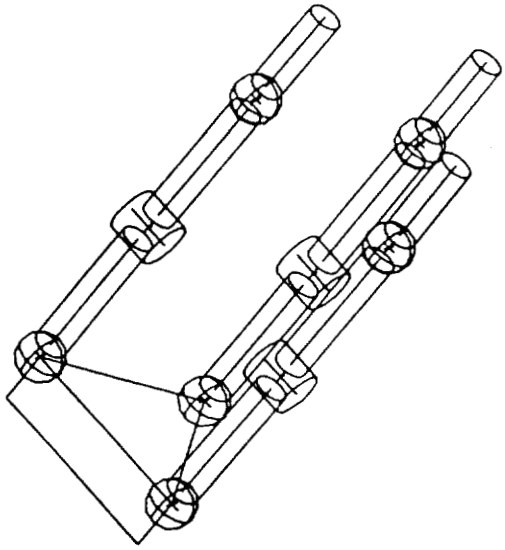
The bodies of the simulated FTS configurations fell into three basic categories based on the number of arms used. Two, three, and four arm FTS configurations were modeled. Within each of these three categories the major variable was the separation distance between adjacent shoulders. The following eight configurations were modeled:

A two arm bar shaped robot with a 48 inch shoulder separation
A three arm equilateral triangle shaped robot with 24 inch shoulders
A three arm equilateral triangle shaped robot with 48 inch shoulders
A four arm square shaped robot with 24 inch shoulders
A four arm square shaped robot with 36 inch shoulders
A four arm square shaped robot with 48 inch shoulders
A four arm rectangular shaped robot with both 24 and 48 inch shoulders
A four arm kite shaped robot with both 24 and 48 inch shoulders

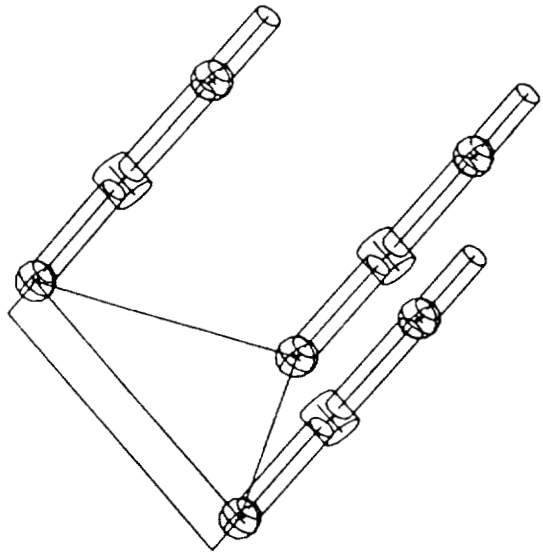
These configurations are illustrated in Figures 2 and 3.



48 Inch Two Arm FTS

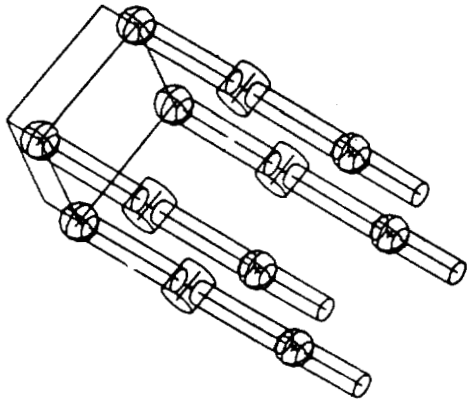


24 Inch Three Arm FTS

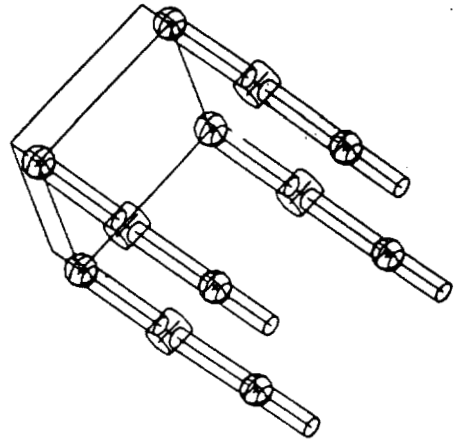


48 Inch Three Arm FTS

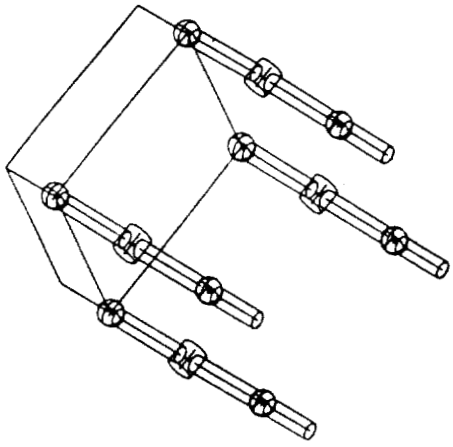
Figure 2. Two and Three Arm FTS Simulation Configurations



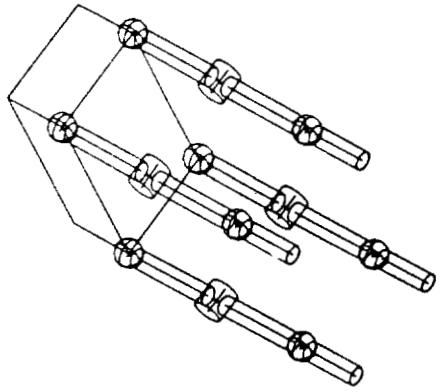
24 Inch Four Arm FTS



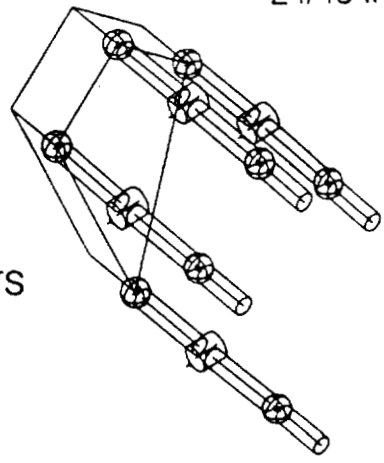
36 Inch Four Arm FTS



48 Inch Four Arm FTS



24/48 Inch Four Arm FTS



Kite Four Arm FTS

Figure 3. Four Arm FTS Simulation Configurations

Configurational Trades, Two Arm FTS ORU Exchange Simulation

A two arm FTS with a shoulder separation of 48 inches was used in this simulation. The ORU exchange simulation started with the FTS being moved inside the truss structure by the MSC arm. In order to pass through the trusswork the FTS orients its arms to minimize its chance of collision. Once inside the trusswork the MSC arm positions the FTS in front of the SIA containing the ORUs. The FTS then grasps a support point on the SIA with one of its arms to stabilize itself relative to the work site. The other arm is used to remove the ORU identified for replacement. Once the ORU has been removed and is clear of the SIA, the FTS releases the support point on the SIA and assumes a posture which minimizes its chance of collision. The MSC arm then transports the FTS and ORU back to the base of the MSC. Once positioned properly, the FTS grasps a support point on the MSC base with its free arm, attaches the ORU to the MSC, and picks up the new replacement ORU. The MSC then transports the FTS and replacement ORU back inside the truss structure. The FTS grasps a support point on the SIA, and places the new ORU in the spot previously occupied by the removed ORU. The FTS is then transported back to the MSC base.

The grasping of the support points on both the SIA and MSC by the FTS is a crucial step in the ORU exchange. In a real exchange, the support point permits the FTS to identify its relative position. The FTS calculates its position in space precisely, since the arm joint angles, location of the work site, and the support point position are known. Automated routines can then be inititated. The support point also allows the FTS to carry the loads associated with connecting and disconnecting the ORU against itself rather than the MSC arm. In the simulation, the locations of the support points were examined to verify their usefulness. Figure 4 shows the two arm FTS removing an ORU. The MSC and other details have been removed for clarity.

The results of this simulation demonstrate that it is possible to exchange an ORU with a two arm FTS. However, two arms are the absolute minimum number necessary to complete the exchange. The dimensions of this particular FTS configuration were compatible with those of the work site. It would be advantageous to have a third arm on the FTS to carry the replacement ORU along when the MSC arm positions the FTS within the truss structure. This would eliminate the need for an extra trip through

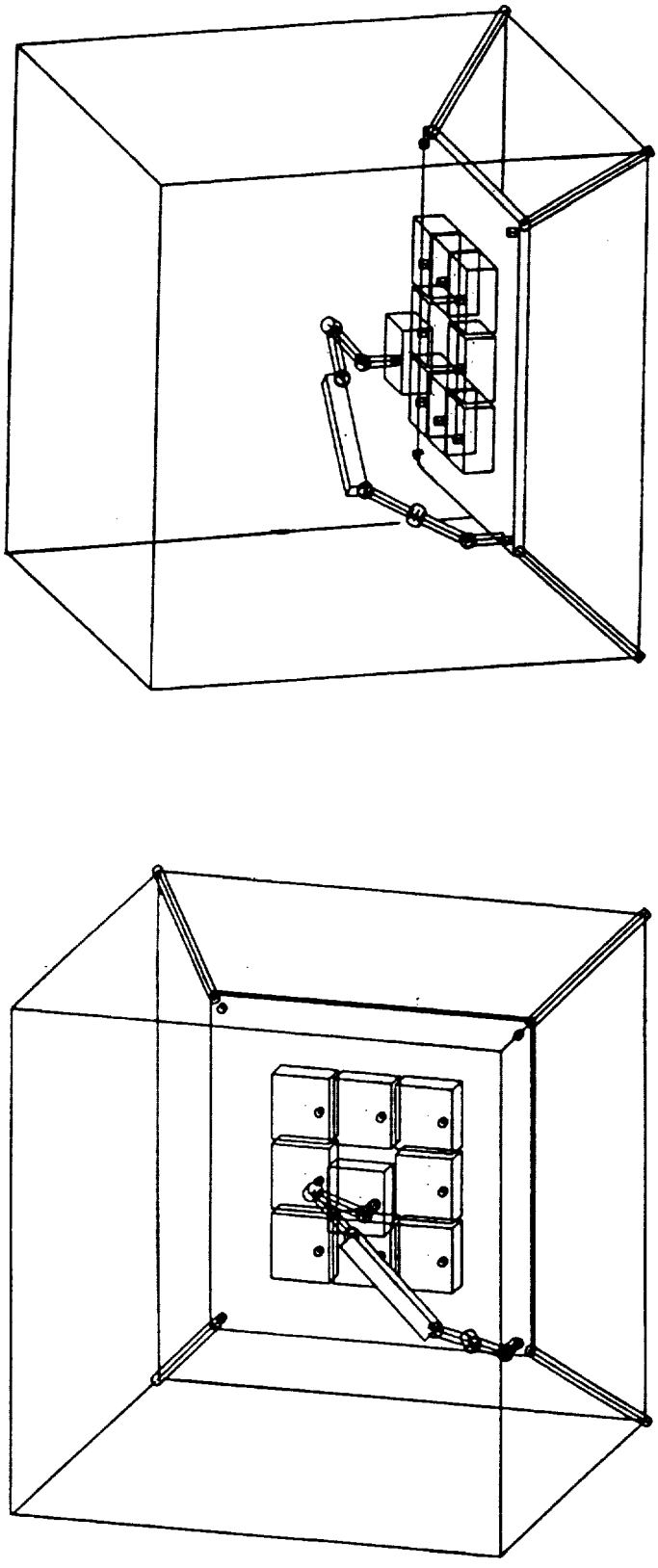


Figure 4. Orthogonal Views of a Two Arm FTS Removing an ORU

the trusswork. The simulation also demonstrated the need for close coordination between the MSC operator and the FTS operator. There were many instances where a lack of coordination between the two operators could result in damage. The movement through the truss and the approach to the SIA and MSC base are of particular concern.

Configuration Trades, Three Arm FTS ORU Exchange Simulation

Two versions of a three arm FTS were simulated. Both versions had equilateral triangle bodies which differed only in size. One used 24 inch shoulder separations, and the other used 48 inch separations.The ORU exchange scenario used for these two simulations was similar to that used previously in the two arm case. Once again, the scenario begins with the MSC arm transporting the FTS. The first stop this time is the MSC base. The FTS uses one arm to grasp a support point and then uses another arm to attach to the replacement ORU. The ORU is disconnected from the MSC and lifted away, the other arm then releases the support point. The MSC arm then transports the FTS and ORU into the truss structure to a location in front of the SIA. In order to do this, both FTS configurations had to assume postures which reduced their collision cross section. This posture is shown in Figure 5. The FTS then uses one arm to grasp the support point on the SIA. While keeping the replacement ORU safely out of the way, the FTS uses its third arm to remove the target ORU. Please refer to Figure 6. The target ORU is moved out of the way and the replacement ORU is moved and attached to the SIA. The FTS releases the SIA support point and the MSC arm transports it back through the trusswork to the MSC base. The FTS grasps a support point and connects the target ORU to the MSC base. The exchange is then complete.

The three arm FTS configurations both have distinct operational advantages over the two arm FTS. An exchange requires significantly less use of the MSC arm and is therefore safer and faster. The 24 and 48 inch shoulders were both capable of performing the ORU exchange. However, the 24 inch shoulder represents the smallest feasible size given the work site dimensions and the length of the FTS arm. With this smaller shoulder separation, the FTS had to stretch its arms to full length to accomplish the exchange. The 48 inch shoulder separation allowed the FTS improved reach under less constrained conditions. The three arm FTS could also emulate the two arm FTS, if necessary. The three arm FTS is capable of

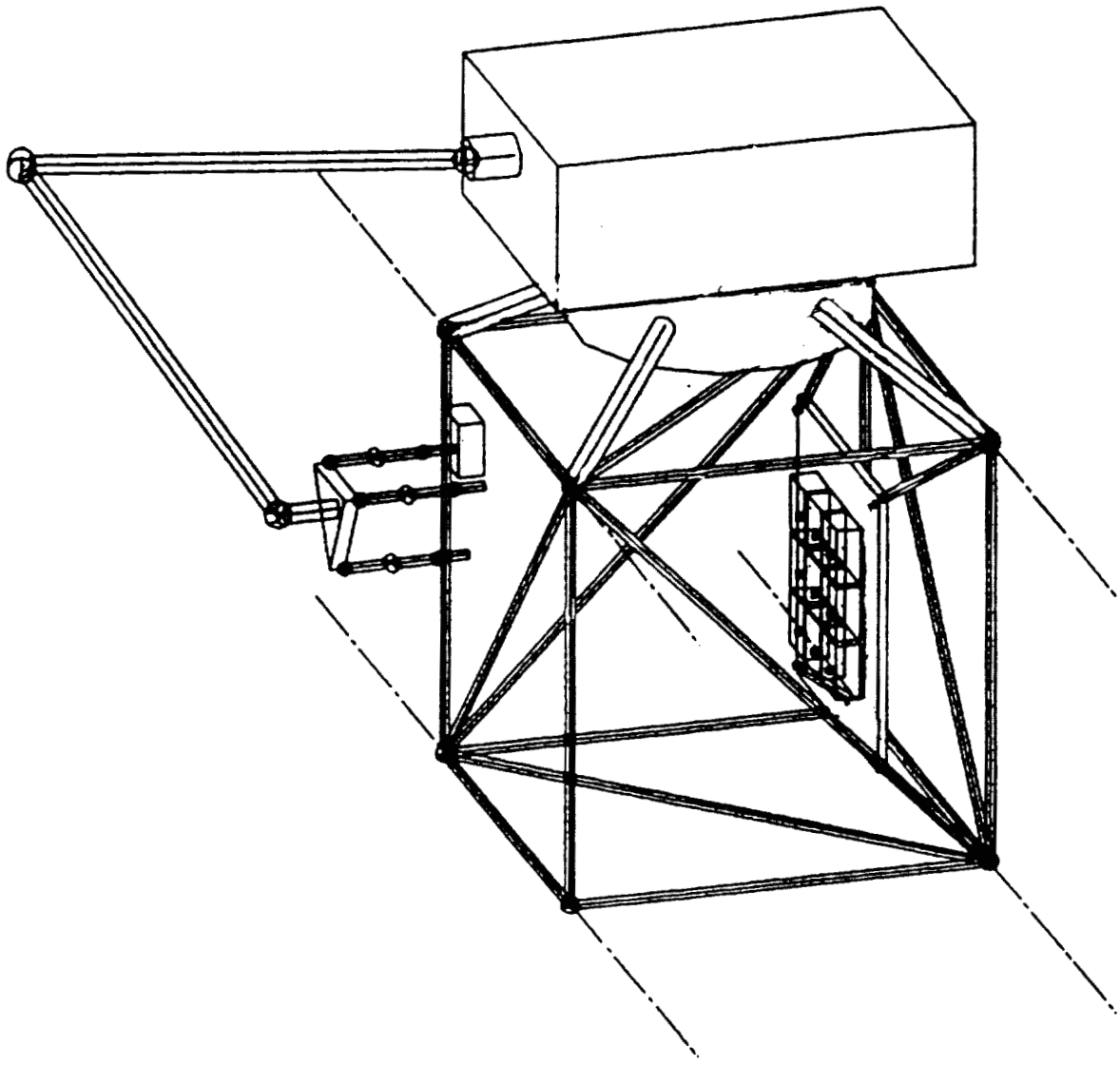


Figure 5. Three Arm, 48 Inch Shoulder, FTS Prior to Insertion into the Truss Cube.

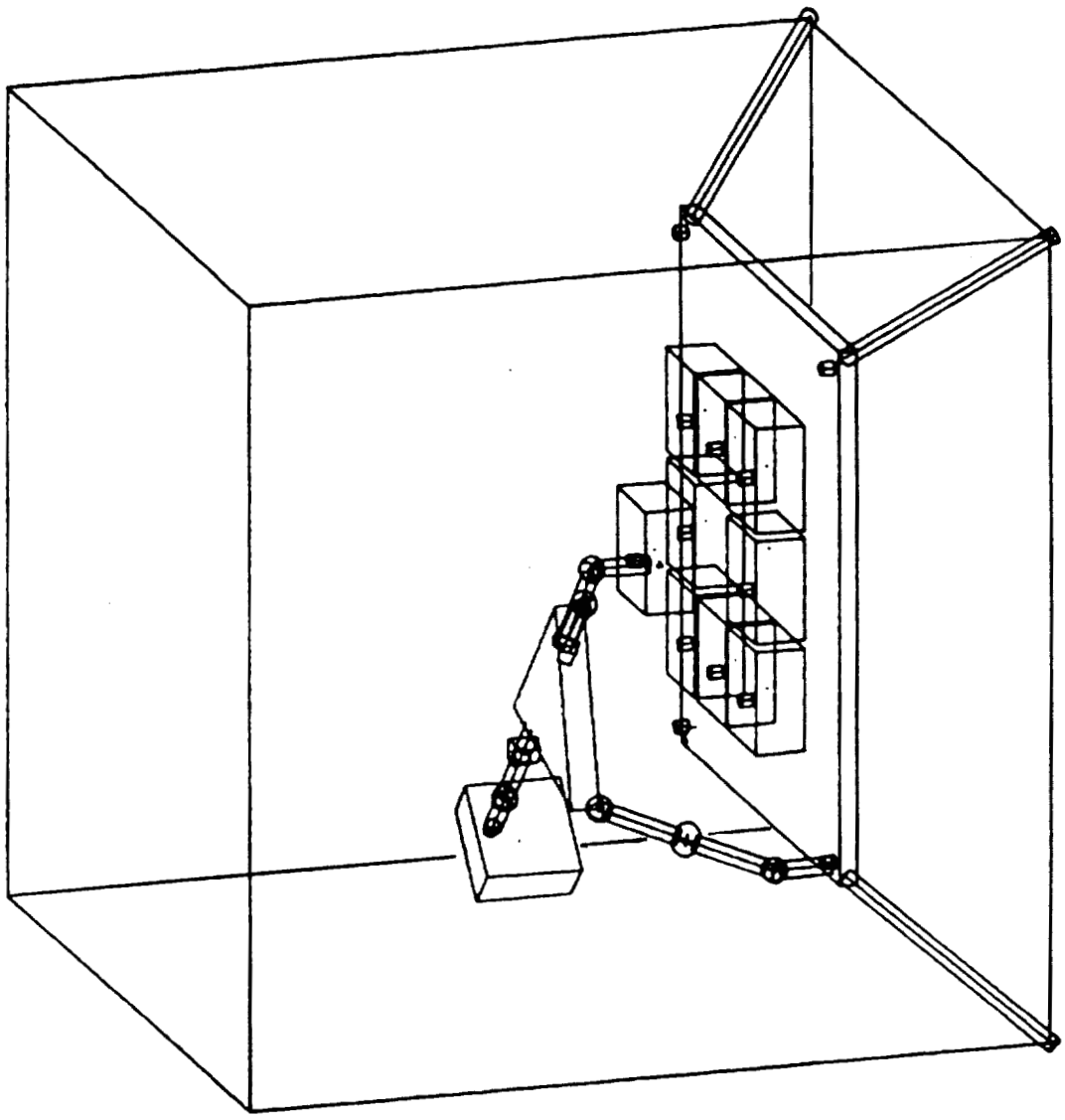


Figure 6. A Three Arm FTS, with a 48 Inch Shoulder Separation, Exchanging an ORU

holding and connecting or disconnecting an ORU which required two arms to operate. This use was not investigated for the three arm configurations, but was baselined for the four arm simulations.

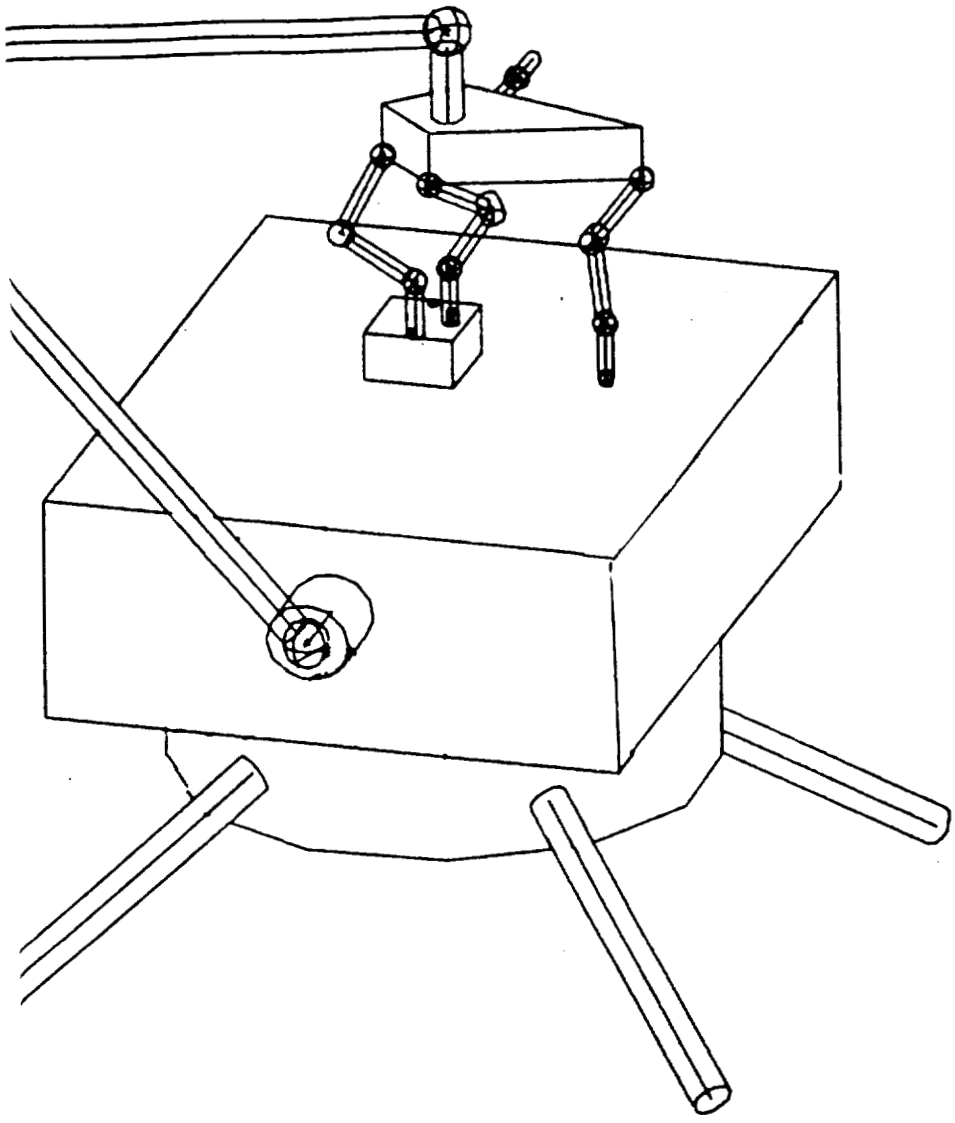
Configuration Trades, Four Arm FTS ORU Exchange Simulation

Five variations of a four arm FTS were simulated, including three square FTS configurations with 24, 36, and 48 inch shoulder separations. A rectangular FTS with two 24 inch and two 48 inch shoulders, and a kite shaped FTS with two adjacent 24 inch shoulders and two adjacent 48 inch shoulders were also investigated.

The ORU exchange scenario simulated for the four arm configurations used ORUs which require two separate arms. One FTS arm holds the ORU and a second operates the connection and disconnection mechanism. Instead of having a specially designed single handle which can be used to both grasp and release/attach the ORU, the new ORU has a handle used only to hold the ORU, and two tie down bolts used to release or attach the ORU. The size of the ORU remains the same.

The scenario used for the simulation started with the MSC arm transporting the FTS to a point near the MSC base. The FTS then uses one arm to grasp a support point, and another to grasp the ORU handle. Using the remaining two arms, the FTS disengages the ORU from the MSC base. These arms are then moved out of the way and the third arm releases the MSC support point. The MSC arm then transports the FTS and ORU to the SIA inside the truss structure. The FTS uses one arm to grasp the SIA support point, and uses the other two arms to grasp the ORU handle and disengage the target ORU. The target ORU is then moved to a point out of the way by one arm. The replacement ORU is then moved into position and the free arm is used to connect it to the SIA. The FTS then releases its hold on the SIA support point and is transported by the MSC arm to the base of the MSC. Once there, the FTS grasps the MSC support point and one arm moves the ORU into place, where the other two arms connect it to the MSC. Figures 7 and 8 illustrate two steps in this scenario for the kite shaped FTS and the 24 inch square FTS.

The tasks simulated in this scenario were considerably more challenging than those used in the previous simulations. The ability of each of the five configurations to reach both the SIA support point with one arm and the proper ORU on the SIA with the remaining three arms was



**\ Kite Shaped Four Arm FTS Removing an ORU from the
: MSC**

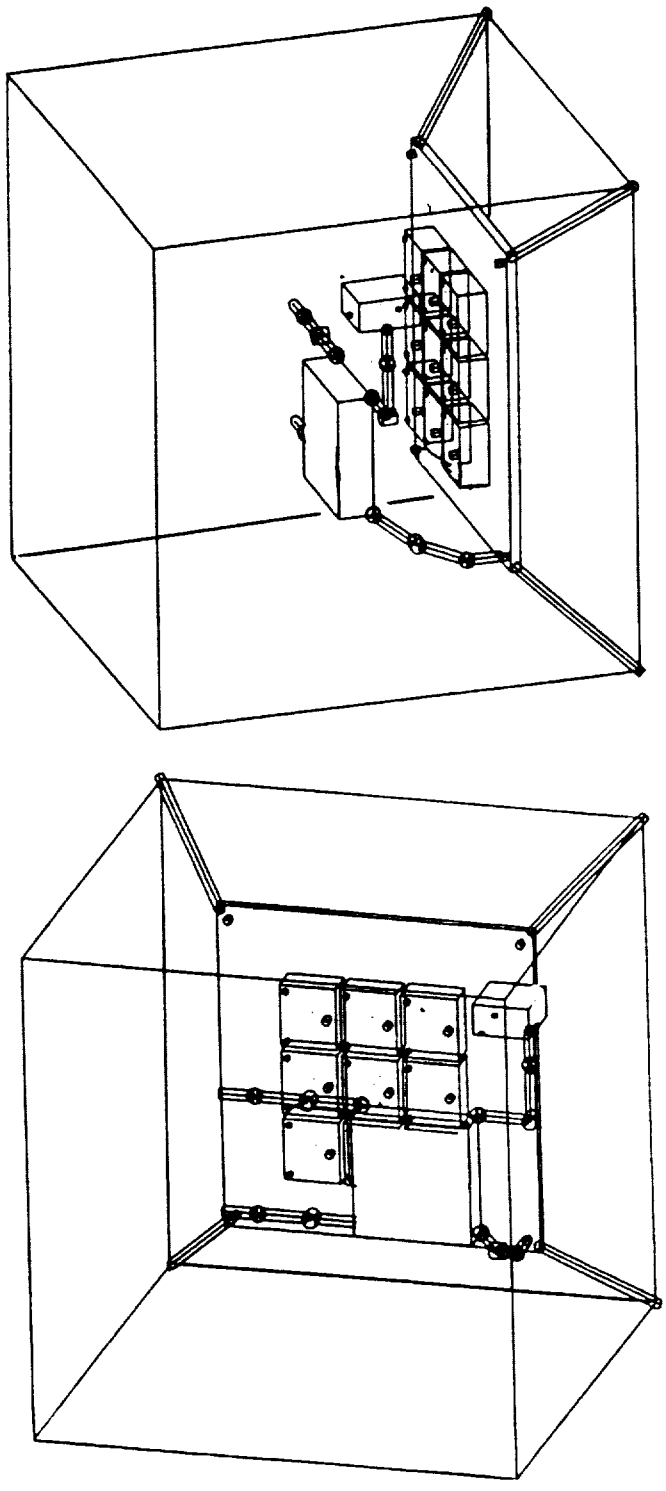


Figure 8. A 48 Inch Square Four Arm FTS Attempting to Exchange an ORU on the SIA

estquare Fable to accomplish this.
s wo its arpt reach far enough. The
areinable ish the same tasks for the
asooulder arms of the FTS too far
w tiether tne ORU. The rectangular
so tete theven though its 48 inch
allowhold thoint and another to hold
indlcould rom the support point end
o tBince thm carried the replacement
ld r Only our arm FTS
ns, are FTcite shaped FTS, were able
ully ORU es it was intended.
3 inwas wethe exchange scenario, its
ieitl too shte shaped FTS represents
tedscenarits arms is located
/ free. Thble to reach easily
e Oport poIA, and still have three
togi the coJ exchange. This example
es trly undich tasks the FTS must
preptimized overall design.
the apparne four arm FTS
Tlfour ar in close proximity
withignificaype of situation, six
reee not cient. Seven degrees of
vidry flexiid interarm interference.
ed, gree oras incorporated at the
hisoility re intervention to

nuown thdesign can benefit from a
tanded u also shown that seven
reessary peration of more than one
tric. The also pointed out that the
hethe ope FTS need to work in close
to safetyspecially true during the
ouucture il approaches to the MSC
A.

All FTS configurations examined used one arm for stabilization at the local work site. Thus, two arms are the minimum number necessary for the baseline ORU exchange using the RCA ORU design. The two and three arm FTS configurations used the RCA ORU design with the combined handle and attachment mechanism. This was the only type of ORU that the two arm FTS could use. A generic ORU with separate handle and attachment mechanism, which requires at least two arms to hold and work, was also simulated. This design was used in all of the four arm FTS configuration simulations. The two arm FTS configuration with 48 inch shoulder separation was compatible with the exchange scenario and the local work sites' dimensions and layouts.

A third arm produces an operational advantage over the two arm FTS in that it can be used to carry the replacement ORU during the first passage through the Station truss and thereby eliminate an extra trip back to the MSC base. A four arm configuration also has this operational advantage. The use of an extra TV camera at the end of a free arm would also provide a remote operator with a useful alternate viewpoint when the work space becomes visually congested.

Both the 24 and 48 inch shoulder separation, equilateral triangle, 3 arm FTS configurations were compatible with the exchange scenario and local work environments. However, the 48 inch shoulder separation had better reach characteristics than the 24 inch design.

The 24 inch shoulder square, 48 inch shoulder square, and rectangular (24 and 48 inch shoulders) FTS designs were not compatible with the baseline work environment and could not accomplish the ORU exchange. The 36 inch shoulder square and 24/48 inch shoulder kite FTS configurations were compatible with the baseline work environment and were able to perform the ORU exchange. Of these two, the 24/48 kite FTS represented the most appropriate design for the baseline generic ORU exchange scenario.

This simulation exercise illustrates how the tasks and work environment associated with one specific ORU exchange scenario influenced the success of each FTS configuration examined. The actual FTS will be expected to be able to accomplish a minimum number of tasks in a minimum number of work environments. Simulations of each of these situations and scenarios will be needed before a serious FTS design can be produced. The final FTS design will, in all probability, be the best

compromise achieved between the task optimized designs used in these simulations. The ability of the FTS to perform future tasks, which are currently unrecognized, will depend on how well these new tasks and work environments can be understood and modeled and on how much they differ from those used in designing the FTS originally.

Acknowledgements

The simulation exercise was funded by the NASA Goddard Space Flight Center as part of contract NAS 5-29400.

ACTUATORS FOR A SPACE MANIPULATOR

W. CHUN AND P. BRUNSON

ADVANCED AUTOMATION TECHNOLOGY (ROBOTICS) GROUP
MARTIN MARIETTA AEROSPACE
DENVER, CO 80201

ABSTRACT

The robotic manipulator can be decomposed into distinct subsystems. One particular area of interest of mechanical subsystems is electromechanical actuators (or drives). For this paper, we will define a drive as a motor with an appropriate transmission.

This paper will give an overview of existing, as well as state-of-the-art drive systems. The scope is limited to space applications. A design philosophy and adequate requirements are the initial steps in designing a space-qualified actuator. We will focus on the d-c motor in conjunction with several types of transmissions (harmonic, tendon, traction, and gear systems). We will evaluate the various transmissions and key performance parameters will be addressed in detail. Included in the assessment is a shuttle RMS joint and a MSFC drive of the Protoflight Manipulator Arm. We will also investigate compound joints.

Space imposes a unique set of requirements for designing a high-performance drive assembly. Its inaccessibility and cryogenic conditions warrant special considerations. This paper will present some guidelines concerning these conditions. The goal is to gain a better understanding in designing a space actuator.

Introduction

The primary manipulator is usually a serial design of drives and arm structure, but additional work is being done on parallel manipulators. In this paper we will concentrate on the drive, otherwise referenced as the motor/ transmission package.

The manipulator may be a two-link design with the joints separated by arm segments. [1] The first choice to be made is do we want distributed actuators or do we want an integrated design? Distributed actuators afford the capability to be modular, maintainable and easily upgradable. As a result, it sacrifices a compact design and a less-than-optimal inertia distribution. The choice in effect also limits the possible transmission options. We will discuss this more later in this paper.

Space represents several advantages as well as disadvantages for a manipulator. (Refer to Table 1 on design factors for a space manipulator.) Microgravity is favorable because it reduces each joint's torque requirement. The lubricant must be space compatible with low outgassing. Moreover, the materials and process must also be compatible and stable. Thermal management is a major concern with several aids such as passive control through thermal coatings and dynamic control with rod-heaters or tapes.

Table 1 Design Factors for Space Manipulators

HOSTILE ENVIRONMENT	POWER SOURCE	WEIGHT & SIZE	RELIABILITY
-Extreme operating temperatures (-67 F to 437 F is common)	-Electric	-Desire a light and compact design	-Must withstand the launch loads
-Plastics and rubber outgas in a vacuum	-Minimum power consumption is desirable	-A design that can be stored compactly	-A simpler design is reliable
-Micro gravity eliminates weight but not its inertia		-Modular design can be reconfigured to satisfy several requirements	-Design should be based on a mature technology

The mechanisms must be demonstrated under thermal vacuum conditions. The biggest hurdle will be the inaccessibility of the hardware in space. As a result, the manipulator needs to incorporate mature technology with a reputable service life. General longevity is a major concern.

This paper will start with the electric motor and be followed by an review of suitable transmissions. We will look at the particular characteristics that make a good actuator with a design methodology. We will discuss several design considerations for each type of transmission. Finally, we will discuss a compound joint.

Motor

The motors in a space manipulator will be electric instead of hydraulic, hydrokinetic, hydrostatic, or pneumatic. Even though the hydraulic output torque is higher per unit weight than electric, the support equipment (pump, compressor, accumulator, etc) and the potential for leakage make hydraulics undesirable. The use of a DC servo motor is a proven technology. Its usage and installation are very clean. The brushless motor [2] is used for the following reasons:

- Brushless units may be operated at much higher speeds and at full torque at those speeds;
- The stator may be mounted in a substantial heat sink to minimize temperature rise and prolong bearing life;
- The brushless motor does not have the brush wearout or the presence of brush wear particles (debris);
- The electromechanical interference (EMI) normally associated with arcing of the brush-commutator interface is eliminated;
- Where long life is required, the limitation of the motor is increased to match the life expectancy of the bearings;
- Less preparation for a space environment (vacuum operation).

One disadvantage with the brushless motor is the added complexity to the commutation electronics. However, this problem has been worked.

The brushless motor consists of a stator that supports the armature coils and shaft bearings, and a rotor that carries permanent-magnet poles. A typical motor uses Alnico magnets. For the next space application, the magnets used will be from the rare-earth family, probably samarium-cobalt [3]. The rare-earth magnet (samarium-cobalt, Neodymium-Iron-Boron) has a higher maximum energy product rating than commercial magnets like Alnico, as shown in Fig. 1.

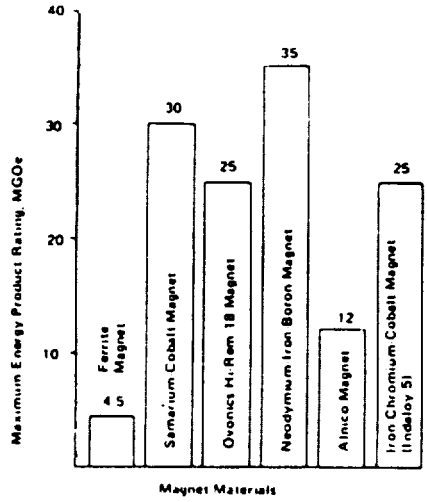


Figure 1 Maximum Energy Product Ratings for Different Magnets

The stronger magnets have a higher airgap flux density, that produces a higher torque for a given volume. Neodymium-Iron-Boron has the potential to produce even greater output torque, but its output has been inconsistent and requires further testing. If the magnets are exposed to temperatures above the magnet cure temperature, there is degraded performance.

The magnets are formed into teeth to accommodate the high flux density. The toothed geometry contributes to high inductance and high iron losses. As a result, the motor has a tendency to cog at low speed. The speed at which cogging is evident is inversely proportional to its pole count. As inductance rises, the transistors that commutate the coils require more current.

The flux levels in the teeth vary as the motor rotates. This causes hysteresis and eddy current lags that may account for more than half of the total losses in conventional brushless motors. It also may cause cogging. One remedy is to skew the orientation of the magnet with respect to the rotor. Unfortunately, this can be costly with difficult manufacturing processes that may result in lower performance.

New motors are being developed to remedy some of these problems. They include: toothless armatures, multipoles, and hybrid motors. The result should be greater performance out of the brushless motors.

Transmissions [4, 5, 6, 7]

The following transmissions to be discussed have applications to robotic manipulators. They are:

1) Direct Drive (no transmission)
2) Traction Drive
3) Harmonic Drive
4) Tendons
   a) Bands
   b) Cables
   c) Chain
5) "Roto-Lok"
6) Gears
   a) Spur
   b) Squirm
7) Torque Tubes
8) Multiple Linear Actuators
9) Linkage
10) Cycloidal Speed Reducers

The majority of the aforementioned transmissions are in use today on commercial as well as research manipulators. This list might not be all-inclusive, but we feel it represents the majority of the available transmission technology today.

Review

Direct Drive [8] - In direct drive, the motor is coupled to the load without any form of mechanical leverage. The result is a simple system without a transmission, as depicted in Fig. 2.

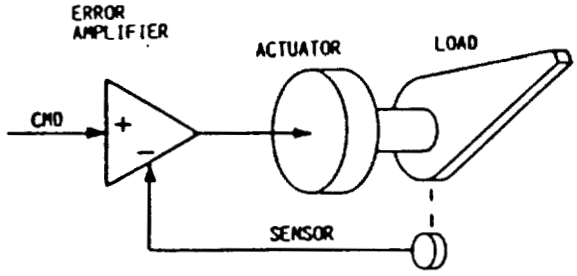


Figure 2 Direct Drive

Takeo Kanade (CMU) and Haruhiko Asada (MIT) have built the first direct drive arm, DD1. CMU has built a second arm, DD2, which is a SCARA design. In the commercial area, Adept Co., Sanyo and United States Robot have used direct drive motors for selected joints.

Traction Drive [9] - Traction drive (Fig. 3) operation is based on the rolling contact between two smooth and unequally sized rollers. The traction forces are transferred via the traction fluid. The normal forces must be large enough to prevent destructive slipping of the rollers. They rely on friction to transport torque.

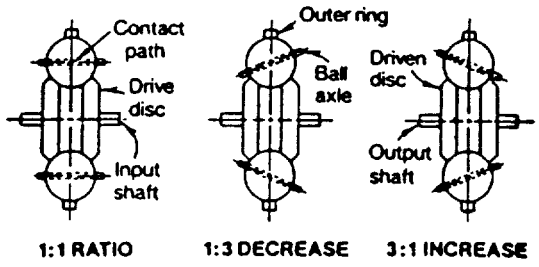


Figure 3 Traction Drive

Harmonic Drive [10] - Harmonic drives in Fig. 4 are precision gear-type speed reducers. Its design is based on elastic body mechanics as opposed to rigid body mechanics for conventional gearing. Harmonic drive uses controlled deflection in the flexspline to reduce speed and multiply torque.

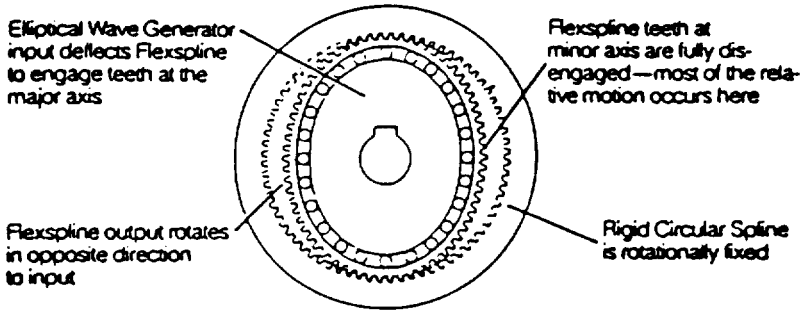


Figure 4 Harmonic Drive

High speed is input into the wave generator. The rigid circular spline is fixed, while the nonrigid flexspline drives the load at reduced speed. The wave generator deflects the nonrigid flexspline into an elliptical shape which meshes the teeth at both ends of the wave generator major axis. Because the flexspline has two fewer teeth than the circular spline, rotation of the wave generator produces relative motion between the two splines.

Currently, R. Cipra at Purdue University uses harmonic drives in their planar arm. NOSC-San Diego uses them in their underwater arm and they are also used by Telerobotics International and Robotics Research.

Tendons - Under certain circumstances, it would be desirable not to collocate the axis of actuation and the motor. There are transmission devices such as cables, belts, bands, and chains to transfer motion from one location to another. Each transfer device has distinct advantages depending on load, distance, and size. For example, steel cables are desirable for small robots. Larger robots use synchronous belts with trapezoidal teeth that fit into sprocket wheels for positive motion. Roller chains are used for longer distances than belts. Central Research Laboratories utilizes steel bands to drive their teleoperated robot in nuclear hot cells. The University of Utah uses a composite band of kevlar and dacron to power its hand, as in Fig. 5.

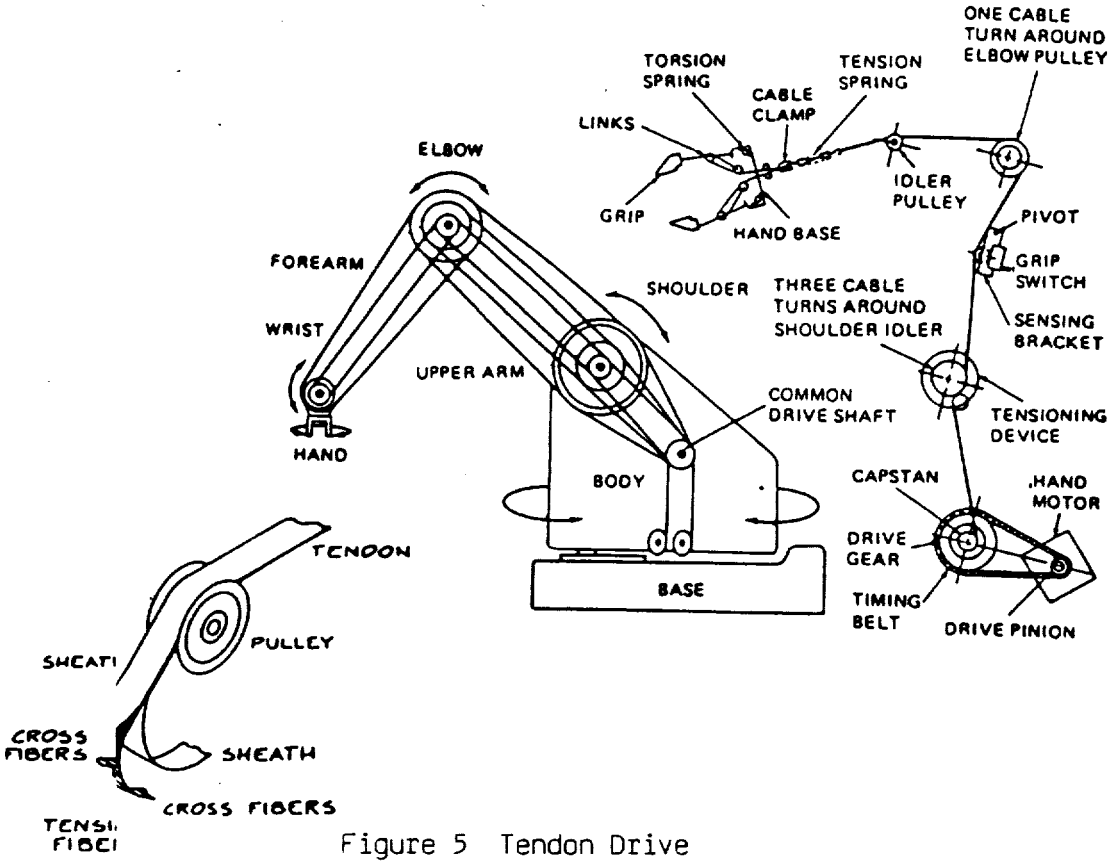


Figure 5 Tendon Drive

Rot11] - A Roto-Lok drive smoothly couples a drive to a load using standar ble cables in a figure-eight configuration. This friction-drive technol patented by TRAX Instrument Corporation. The system uses simple cylindr apes and spring-compensated cables. Roto-Lok was originally develop critical positioning of astronomical instruments. Each cable is preload spring so that the driving and driven shafts (cylinders) are linked pretensioned cable members in a "figure-eight" wrap, one pulling in each ion (see Fig. 6).

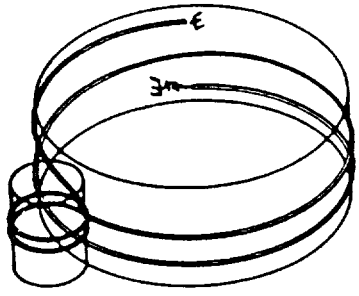


Figure 6 Roto-Lok Technology

Gears [12]

A. Spur Gear - Spur gears (reference Fig. 7) constitute one of the best means for transmitting motion from one shaft to another. They are usually cylindrical in shape and the teeth are straight and parallel to the axis of rotation. Mechanical energy can be transferred from one rotating shaft to another by meshing the teeth of two gears.

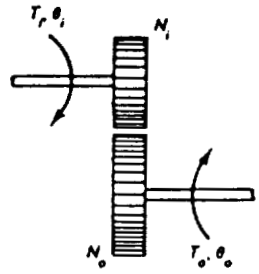


Figure 7 Spur Gears

A gear train is made up of two or more gears used to change the angular velocity, torque, force, etc, of the output relative to the input. Even though gears look ordinary their design, engineering, and manufacturing are highly developed.

B. Squirm Drive [13, 14] - Squirm drive (Fig. 8) is a second-generation worm drive developed by Maxaxam Corp. The worm wheel carries a number of free-spinning roller spindles on its periphery. The spindles engage the roller screw thread and produce rolling motion on the point of contact between roller wheel and screw, thus reducing sliding contact friction common in conventional worm design.

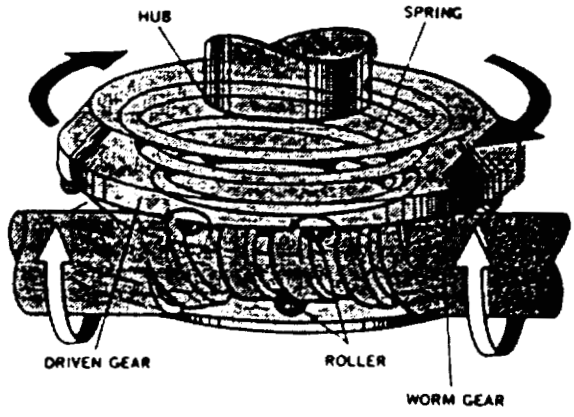


Figure 8 Squirm Drive

Torque Tubes - Torque tubes in Fig. 9 are very light drive shafts that can transmit power from one point to another. Coupled to a bevel gearset, they are used to drive wrist bend-type joints. It is possible to run several of these tubes concentrically. Oak Ridge National Laboratory uses torque tubes in their Advanced Servo Manipulator (ASM) as well as Cincinatti-Milacron in their industrial robot wrists.

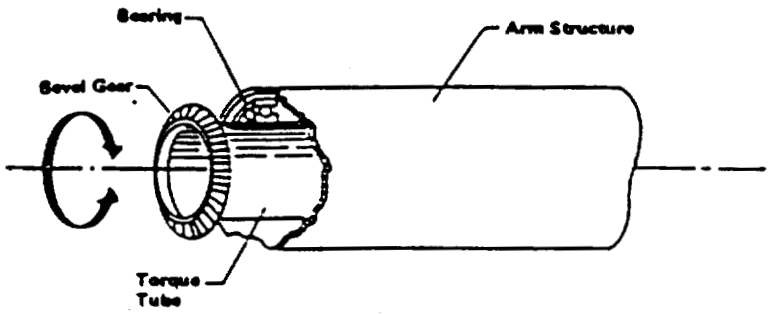


Figure 9 Torque Tubes

Multiple Linear Actuator [15] - In this system one motor drives two counter-rotating leadscrews (Fig. 10) in forming a mechanical "power bus". The "power bus" can drive up to 16 linear actuators. Each actuator module houses a pair of electromagnetic brakes and recessed inside each brake is a freely rotating lead screw. When its brake is energized, the nut locks in

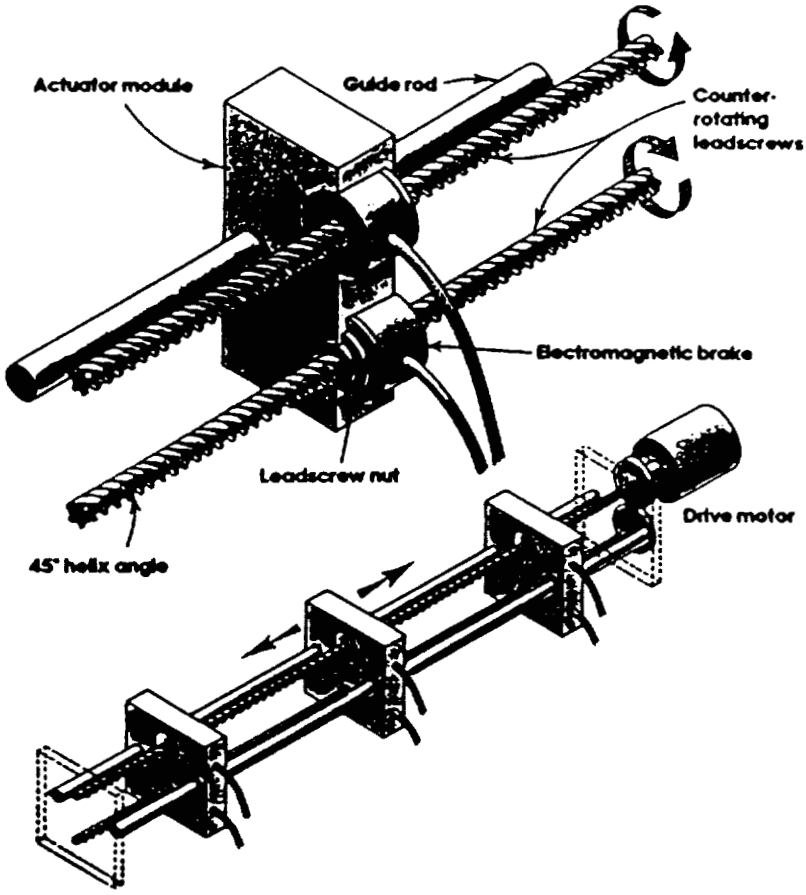


Figure 10 Multiple Linear Actuator

place and moves the module along the lead screw. By cutting the power to the brake, it releases the nut and stops the module in a new position. Energizing the other brake moves a module in the opposite direction. The bidirectional "power bus" has been built by Victory Enterprise Technology Inc., and is incorporated in their new robotic hand.

Linkages - Linkages are often the simplest and most economical way to generate machine motion. Every mechanism can be represented by a skeleton diagram, which is the most basic representation of the specific mechanism that will produce a required motion; Fig. 11.

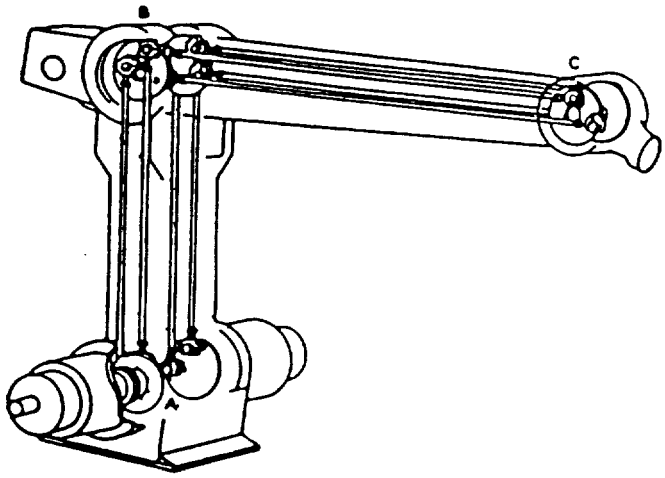


Figure 11 Linkages

Two major type of linkages are the parallelogram and the pantograph. The parallelogram can be characterized by being a compact mechanism that provides a large work envelope. The pantograph is similar to the parallelogram but is capable of magnifying its input.

Linkage mechanisms are used in the GE and Bendix industrial robots. The legs of Odex 1 by Odetics are powered by linkages as well as one of the JPL manipulators, CURV.

Speed Reducers - Speed reducers are transmissions that reduce movement and proportionately amplify torque. They perform the same functions as a gear reducer. The better known cycloidal drives are mentioned here.

Dojen Orbital Drive - The Dojen actuator (Fig. 12) consists of a housing, dual track cam, input shaft, and a output shaft. The dual track cam is sandwiched between the housing and output shaft. Each trochoidal-shaped track mates to a corresponding roller. There is one more roller than the number of lobes on each matching track. A rotary input causes the cam to orbit inside the set of rollers and phase shift occurs causing an angular displacement between each jet of rollers around the axis of the main bearing.

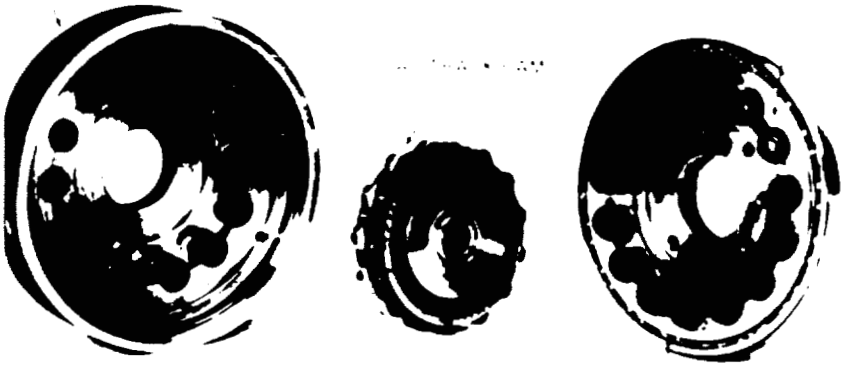


Figure 12 Dojen Orbital Drive

i-Friction Drive - Figure 13 is representative of another cycloidal
The input shaft drives an eccentric that causes the inner ring to
ıside the corresponding outer ring. A second inner ring, which is
ed to the primary inner ring, orbits inside a second outer ring
rigidly connected to the output shaft.

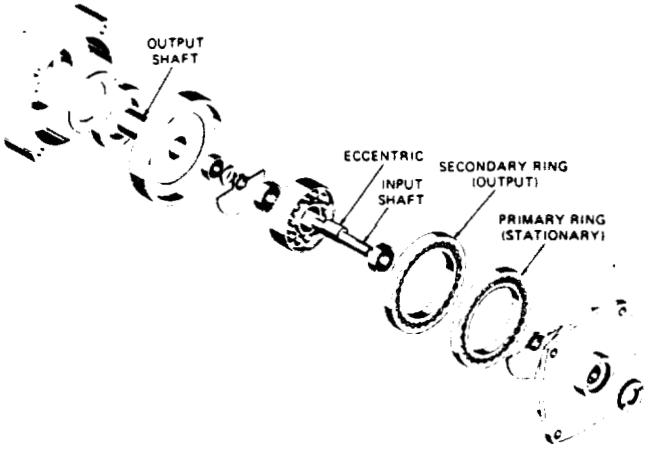


Figure 13 Anti-Friction Drive

orbiting members are designed not to mesh with the outer rollers.
, there are two more lobes in the outer ring than in the inner
Rollers separate inner and outer rings. There is one less roller
ter ring lobes. Because of the difference in the number of lobes
lers, rotation of the orbital drive produces a continuous angular
ement.

SM-Servo-Match Precision Torque Multiplying Component - Built by Sumitomo Machinery Corp., the precision unit in Fig. 14 is used for torque multiplying. It is a planetary design and both input and output shafts are eccentric. There are only three major moving parts. The gear teeth are cycloidal shaped and are much stronger than conventional involute gears. All torque is transmitted through rollers to minimize friction and wear.

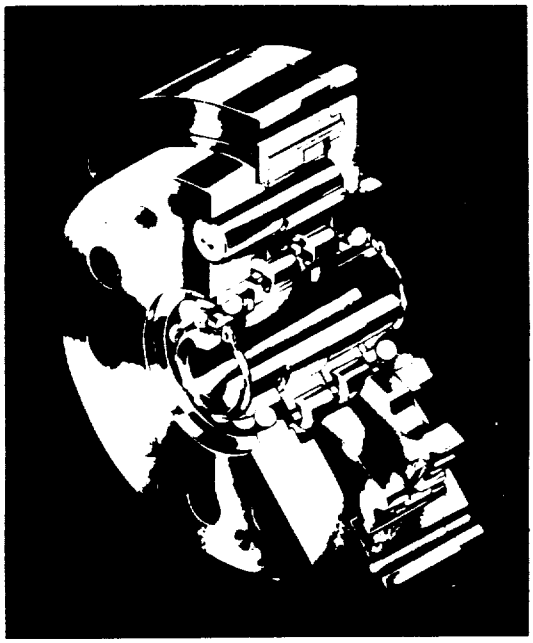


Figure 14 SM-Servo-Match Torque Multiplying Component

Actuator Transmission Criteria

In determining the optimal manipulator actuation scheme, the designer would have to approach the various choices from both the system requirement level and the component level. For example, what is the geometry of the arm? Is it a two-link manipulator or is it a gantry design? Some general decisions must be made up front and for other decisions, there might not be a choice [16].

See Fig. 15 for a systematic flow chart in determining the optimum transmission. The first fundamental question is whether the actuators be distributed or integrated? For distributed actuation, the driver of the joint is collocated at the point of flexure while integrated actuation would have the driver and point of flexure separated by some distance. Either choice will limit the type of transmission. Another major consideration is the environment in which the manipulator will operate; a battlefield scenario has distinct differences from a space environment.

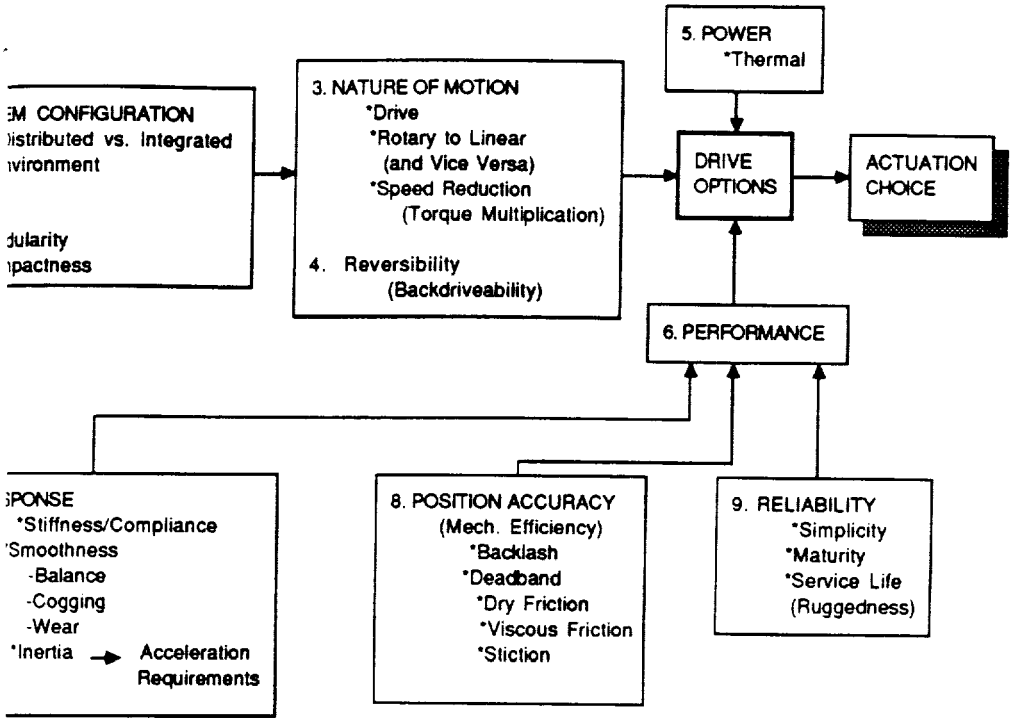


Figure 15 Transmission Criteria Flow Chart

t system requirement is the size and geometry of the arm. Two main his system will be the modularity issue for growth and maintenance kaging of each particular actuator.

case of space we have chosen a motor and need to choose the n. There are three primary motions for the transmission: to drive a motion, to convert rotary to linear motion and vice versa, and to d which consequently increases torque. In addition, the reversi- he transmission is a parameter. The desirability to backdrive a have an impact on the type of transmission chosen. For example, a s not backdriveable.

point, the designer goes into more specific analysis. The emark was made in reference to an industrial manipulator. "The ransmission systems is determined by the nature of the actuators ucture of the robot. The choice often comes down to the question: iculation to be moved is already set in a given position and the iosen, where can it be placed?'" [17] Unfortunately, there is more the transmission than location.

Power consumption is a key parameter for space and is primarily determined by the motor. Any inefficiencies of the transmission will directly affect power consumption. Moreover, thermal considerations are closely related to power.

The most important area for a transmission is its performance. Performance is the summation of response, position accuracy, and reliability. It is imperative that the joint respond accordingly. Compliance in the hardware will adversely affect its desired motion. Its movement should also be smooth. An unbalanced rotary motion could cause a cyclic output motion, but cogging is even worse . They lead to unnecessary wear on components. The load inertia with respect to the motor inertia will impact its acceleration requirements.

The ability to position a joint accurately is instrumental for precision tasks. The accuracy of the joint or its mechanical efficiency is characterized by not having backlash, deadband, dry friction, viscous friction and startup friction (stiction). Each nonlinearity detracts from the accuracy of the joint and should be eliminated or at least minimized.

The last performance deterrent is reliability. If the arm is inaccessible as in space, it needs to be dependable with a good service life. A mature technology is important. When designing the arm, not enough can be said about simplicity and its success in the field. Unfortunately, not everything works that way. As a result, the choice for the arms actuation scheme is a balance of system requirements with specific parameters for its performance.

Design Considerations

In comparing the various transmissions, they should be grouped by the various natures of motion. They will be divided as follows:

Drive Function--
- Direct Drive
- Tendon
- Torque Tube
- Linkage

Rotary to Linear Motion (vice versa)--
- Multiple-Linear Actuator

Speed Reduction--
- Traction Drive
- Harmonic Drive
- Roto-Lok
- Gears
- Cycloidal

Drive Motion - Direct drives are reliable and have no backlash. They have low friction and low compliance. The absence of the transmission eliminates cogging and stiction that are inherent in most transmissions. However, there are a few disadvantages. The first deficiency is in its torque output. With no torque multiplication, the motor must handle all the load. It is very evident when working in 1-g. To meet its torque requirement, the motors tend to be very large and heavy. The biggest problem is that the motor will take a lot of power to operate. Secondly, its large size at each joint will take away from having a slim manipulator to reach into tight places. By not having a torque multiplier (speed reducer), the motor becomes sensitive to loads when its inertia changes.

Linkages are an alternative to driving a joint remotely. In some cases, they can magnify torque like in a pantograph. Linkages can be extremely accurate and stiff. The positional accuracy of a link is dependent upon link flexibility, the link mode of deflection, and joint clearances. Of these factors, joint clearances are the prime source of linkage errors. Linkages are fast and reliable. The relocation of the motors closer to the base optimizes the arms weight distribution and dynamic characteristics. A parallelogram minimizes the motor inertia of the servomotor. It provides a large work envelope for a compact mechanism. Linkages are ideal for certain conditions. A major disadvantage is its limited travel and rotation. It is very difficult to get a full revolution at a pitch (rotary) joint. The output motion of the linkage can be nonlinear. The limitations on its life are a function of its bearings and connecting shafts. The stiffness of the system is dependent on the link and its material. The optimal link design may prove to be heavy or complex.

The torque tube primarily transfers power in a straight path. For the most part, they are usually used in conjunction with gears. Torque tubes are designed to minimize unnecessary weight. By doing so, its compliance increases and its dynamic response decreases. Conversely, stiffening the tube for torsion increases the weight of the tube. A major concern is the fatigue life of the tube material which is sensitive to overloading or shock loading that might twist or crack the tube. In using concentric tubes, we have found a substantial increase in weight due to the bearings.

Combining belts, bands, cable and chains under the group called "tendons" could be a misnomer. Besides the advantage of relocating the drive motors closer to the base, tendons are the simplest to use. Belts, bands, and cable perform very similarly. They are the smoothest transmissions. Its stiffness is a function of its material. They are limited by the material fatigue strength as well as the minimum pulley size. Belts, bands, and cables are not very efficient. This can be contributed to high belt tension and bearings that are highly preloaded. It is possible to have the belt skip off a pulley through wear and high speed operation. Tracking problems occur as the tendons get longer.

Chains perform like the previously mentioned tendons. They can produce a low reduction ratio by utilizing different pitch sizes of sprockets and pulleys. Chain operation is not very smooth and is more susceptible to wear and lubrication problems.

Rotary to Linear - In addition to the multiple-linear actuator system, there are several conventional mechanisms that do motion conversion. Most designers are familiar with rack and pinions and ball screws. The stiffness of rack and pinions is a function of the length of the linear motion. Without a preload there will be inaccuracies due to backlash.

Ball screws are highly efficient and offer a large mechanical advantage. They are moderately smooth with medium stiffness. Ball screws are repeatable and have high positioning accuracy. They can be preloaded or used in pairs to minimize backlash. It should be noted that an increase in preload will directly preclude an increase in friction. This is nominal for all gear-type systems. Ball screws are a very mature technology. The biggest concern with ball screws is torsional vibration and windup. The systems tend to be heavy and not always backdriveable.

On the other hand, the multiple linear actuator is a new technology that has not been proven. It can be compact and versatile. A microprocessor controls the various parameters like force, damping, and gain. The result is a delicate control system. It can reverse directions instantly by having low inertias. The motions are not backdriveable due to the high helix angle of the bidirectional bus.

With one motor actuating 16 separate joints, the elimination of individual motors reduces the weight of the system. However, conservation of energy tells us that for a limited energy source, the utilization of several joints simultaneously should reduce the performance of one or all of the joints. Secondly, each joint is travel-limited and might not fulfill the requirements. Another concern is the potential for the lead screw to bend, causing the actuator module to bind or jam.

Speed Reduction - The standard for speed reduction is a gearbox or gear technology. It is the most rugged and proven transmission. There are several factors to consider in using gears. For example, gear material, its surface treatment, manufacturing precision, gear ratios, types of gears, the gear shaft support, center distances, and lubrication all have to be considered. In particular, spur gears can be packaged compactly to obtain high ratios. They produce minimal axial forces that result in a reduced emphasis in controlling play in the gear mount. In contrast, helical gears produce axial loads that must be constrained to maintain drive stiffness. On the positive side, helical gears have a higher contact ratio which results in a smoother and quieter output. The limiting factor in the gear train stiffness is the gear tooth itself.

Gears are backdriveable. The rule of thumb is that the lower the gear ratio, the more backdriveable the drive. The majority of the backdrive resistance is imposed by the motor due to back EMF and electrical friction. There is no set ratio that designates a drive as being backdriveable. This is a good place to mention the Remote Manipulator System (RMS) on the shuttle [18]. The RMS is a distributed actuator manipulator. Each joint has a low speed and high speed gear train in series. It uses a planetary gearset for the low speed portion, while the high speed portion uses spur gears. The three wrist drives have the same gear ratio--approximately 738:1. The wrist is reportedly backdriveable as well as the shoulder, having a 1842:1 gear ratio. The RMS joints have backlash but the arm is very functional for being 50 ft long and satisfying its requirement of positioning its tip within ± 2 in. and ± 1 deg.

The next closest space manipulator is the Protoflight Manipulator Arm (PFMA) at Marshall Space Flight Center [19]. This 7-DOF arm is also a distributed actuator design. The drives were designed and built at Martin Marietta Denver Aerospace and uses a spur gear train. The drive train uses a dual path design. One motor drives two mirrored gear trains that meet at a common internal ring gear. By preloading one of the paths we were able to virtually eliminate backlash without compromising friction. The result is a joint with a high spring rate and low static breakaway friction. The gear ratios in the PFMA range from 86.4 to 110. The joint is backdriveable and is a precise positioning mechanism.

We have tested the internal ring gear output against an external drive, a standard ring and pinion. The ring gear arrangement displayed improved static and dynamic friction. It was also twice as stiff as the external drive and can be packaged more compactly.

There are other parameters to be considered besides backlash and friction. For gear trains, avoid output shaft powers greater than half the rated motor peak power. The gear train vs speed curves and the power considerations determine the various motor choices--the larger the motor, the higher the duty cycle; especially for DC servo motors. One source of friction can be attributed to magnetic effects like hysteresis drag and cogging. A high torque-to-weight ratio will result in higher magnetic friction. If weight is a major concern, gear weight is dominant for high gear ratios and motor weight dominates for low gear ratio.

The squirm drive is a notable advancement in the worm gear. Preloaded rollers that eliminate backlash and reduce friction are imbedded in the worm gear. Its rolling contact instead of sliding makes this mechanism very efficient. Stiction is reduced and the transmission is backdriveable. It is 98% efficient and has minimum backlash; however, the technology is not mature and it is not compact.

The second most common speed reducer is the harmonic drive. It is low in weight, has a high single-stage reduction ratio, is highly reliable, has negligible backlash, and has a high torque output due to the large number of teeth meshed at any one time. Harmonic drives are accurate to arc seconds and possess excellent repeatability. Gear tooth wear is negligible and the life of the transmission is as good as the surface fatigue of the bearing inner race. With no lubricant, efficiency can drop to as low as 50% from the usual 80-90% rating when using a wet lubricant.

From our earlier tests, harmonic drives 30% additional torque to overcome static friction [20]. It exhibited substantial wind-up with a nonlinear torsional spring rate at low torque. It is not the smoothest transmission and has a tendency to cog at low speed. Furthermore, the inertia of the wave genertor may be too high for certain rapid-response servosystems. The drive can have substantial spring deflection. Compliance is reduced by stiffening the system, but friction increases--and vice versa.

Using a force sensor in the drive can minimize the disadvantages mentioned previously [21]. It should be noted that JPL has harmonic drives in space on their dual drive actuator for Galileo [22].

The Roto-Lok technology is a hybrid design that combines tendon-type technology with different pitches representative of bevel gear trains [23]. As previously mentioned, a cable is extremely smooth and Roto-Lok is no exception. Backlash is eliminated by preloading the cables. This technology is mature and has been applied to precision pointing devices. The rotational stiffness of the system is maximized by locating the drive cylinder very close to the output cylinder and maximizing the contact angle of the cables with a "figure-eight" design.

Howne Roto-Lok technology has several disadvantages; primarily its limitedon and large size. The ratio of speed reduction is proportional to its r for a single stage resulting in a substantial volume. The spring increases its friction but prevents the cables from slipping; howeverriction losses are less than comparable gears and belts. The torsion-up and bending in the input cylinder is another weakness. Given tt requirements, the Roto-Lok can be very effective.

Gea are an effective load-carrying member, but also a source of backlasline pitch gear will be quieter with less backlash; unfortunately its rati capacity will decrease. The traction drive is the result of having nite amount of teeth on a bevel gear. By preloading the tractio, the forces are transferred using the phenomenon called creep.

Zerash, high torsion stiffness, smooth operation and a compact size make thsmission desirable. The tallest pole it must hurdle is its acceptal The technology is not yet mature and needs to be developed. The amooreload required to make the transmission operational is staggerpupled with rollers that must be critically aligned, the tractior (including rolling element) is susceptible to fatigue.

Duesional slippage, a position sensor should be mounted to its motion for closed-loop control. The traction drive can be designed to operate:e without lubrication. Positioning accuracy can be thrown off by thermansion. Initial tests from ongoing work at Lewis Research Center a life of about 10 hours. The development of this transmission is:ess and the results to date are very promising.

The:oup of speed reducers is cycloidal drives. They are in their early st utilization on industrial robots. They are characterized by having single stage reduction ratio, zero backlash and high torsional stiffnesir small size can be coupled directly to the motor. Power is transmita rolling action, minimizing wear and frictional heat, rather than slintact. The transmission will carry a greater load for its size due to th stronger cycloidal-shaped teeth. Its low inertia means a faster ation. However, cycloidal drives have disadvantqages. They are very heahave possible speed variations. Their efficiency drops to 50% unloadeposed to 95% fully loaded.

Compound

A cc joint or a differential is a clever mechanism to create a joint with con axes. These joints are ideal for a robotic wrist and possibly for the:r or a 2-DOF elbow. This is not a new device; it has been in use sinc manipulators for the nuclear industry. This compound joint can be bmpactly but not with distributed actuators. The motors could be adjacthe joint or be placed in the base.

In t common compound joint (Fig. 16), there are two inputs with two outputs.:s 1 and 2 could be a bevel gear, a traction cone or cable driven. nputs No. 1 and No. 2 rotate in the same direction, it results in a benn at output "A". When input No. 1 turns opposite to input No. 2, tit rolls in output "B". The differential with bevel gears can be prelo minimize backlash. A traction input has no backlash but

h loading. Alignment is critical and is susceptible to wear and
e same input to the differential can be accomplished with pulleys
Precise movement can be controlled with an antagonistic pair of
With some type of tension on the cable, the same pulley can be
h a single actuator.

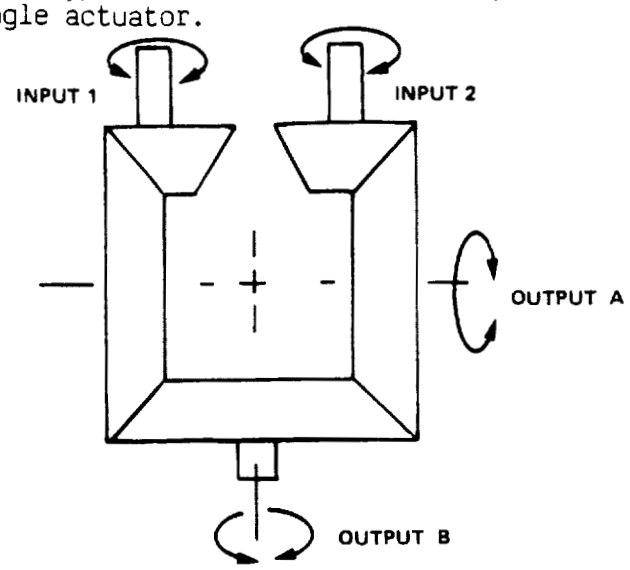


Figure 16 Compound Joint

ne physical configuration of the joint, the bend output has a
nt of travel. There is no reason a compound joint could not be
n a speed reducer. A good example is the Man-Equivalent Tele-
ot (METR) module being developed at ORNL [24]. Each motor goes
eed reducer before input to the traction differential. Finally,
joint is more complex, but exhibits a kinematic advantage by
ntersecting axes.

taken a look at the state of the art in robotic actuators. In
we feel that the motor for a robotic joint in space will be a
gnet DC motor. Having settled on a motor, the options for the
e in the transmission. We have reviewed ten candidate transmis-
s has its own merits as well as disadvantages. In determining the
smission for a particular joint, there is a methodology to be
hoosing the hardware. This was shown in Figure 15 and takes into
n different parameters such as performance, environment, and
e parameters were further discussed during the design consider-
also took a look at a compound joint/differential. Again, there
es and disadvantages. Finally, there is no perfect transmission.
ve their own place and application. It is up to the manipulator
optimize his task by utilizing the desired mechanism and
to satisfy the desired requirement.

References

1. P. Brunson, W. Chun, P. Cogeos: "Next-Generation Space Manipulator." Proceedings of the Conference on Artificial Intelligence for Space Applications, Huntsville, Alabama, November 13-14, 1986.

2. Direct Drive Engineering Handbook. Magnetic Technology, Canoga Park, CA.

3. K. Dekker: "New Materials for Improved Motor Designs." Motion, May/June 1986, pp. 6-13.

4. Intelligent Task Automation (ITA). AFWAL-TR-85-4062, Phase I, Vol. III, Martin Marietta, Denver Division, January 1985.

5. Mechanical Drives Reference Issue. Machine Design, October 16, 1986.

6. A. Critchlow: Introduction to Robotics. Macmillan Publishing Co.

7. W. Holzbock: Robotic Technology: Principles and Practice. Van Nostrand Reinhold Company, Inc.

8. B. Powell: "Direct Drives for Precision Robots." Machine Design, March 20, 1986.

9. S. Loewenthal, D. Rohn, B. Steinetz: "Application of Traction Drives as Servo Mechanisms." 19th Aerospace Mechanism Symposium, May 1985.

10. J. Carlson: "Harmonic Drives for Servomechanisms." Machine Design, January 10, 1985.

11. Roto-Lok Rotary Design System. TRAX Instruments, Albuquerque, New Mexico.

12. J. Coy: Geared Power Transmission Technology. NASA Document N83-20122.

13. D. Bak: "Gearset Stops, Reverses with Zero Backlash." Design News, September 23, 1985, pp. 94-97.

14. E. Lindsley: "Impossible Squirm Drive." Popular Science, November 1984, pp. 76-78.

15. Personal correspondence with Victory Enterprises, Austin, Texas.

16. W. Seering, V. Scheinman: "Mechanical Design of an Industrial Robot." Handbook of Industrial Robotics, Simon Nof (Editor), John Wiley and Sons.

17. H. Warnecke, R. Schraft, M. Wanner: "Mechanical Design of the Robot System." Handbook of Industrial Robotics, Simon Nof (Editor), John Wiley and Sons.

18. P. Nguyen, R. Ravindran, R. Carr, D. Gossain, K. Doetsch: "Structural Flexibility of the Shuttle Remote Manipulator System Mechanical Arm." Guidance and Control Conference, San Diego, CA, August 1982.

ıt)

ıt Manipulator Arm (PFMA). MCR-77-201, Final Report, Martin Denver Division, April 1977.

Manipulator System Design and Concept for Zero-g Simulation. , Contract NAS9-13027, Final Report, Martin Marietta, Denver June 1973.

. J. Thompson, J. Farrell: "Design and Control of Modular, ılly Redundant Manipulators." Second AIAA/NASA/USAF Symposium :ion, Robotics and Advanced Computing for the National Space Iarch 1987.

ı: Dual Drive Actuators. NASA Document N82-23352.

"Mechanical Power Transmissions: Fighting Back Electrical Machine Design, January 22 1987, pp. 76-80.

W. Hamel: "Application of a Traction-Drive Seven-Degrees-of- ılerobot to Space Manipulation." 10th Annual AAS Guidance and ınference, Denver, CO, February 1987.

CABLE APPLICATIONS IN ROBOT COMPLIANT DEVICES

BY

JAMES J. KERLEY

Code 754.1
Goddard Space Flight Center
Greenbelt, MD

CABLE APPLICATIONS IN ROBOT COMPLIANT DEVICES

Numerous different mechanical/electromechanical compliant devices have been designed in the course of robot developments. These hardware devices, coupled with the software program, greatly influence the overall effectiveness of the robot to acquire the work object and perform simple mechanical tasks. This report describes an engineering model compliant device utilizing cable, as part of the mechanical attachment of a tool to a simulated robot arm. This unique cable system offers a simple inexpensive method to gain a high degree of compliance and permit the acquisition and mate-up of robot held tools with the work object in the presence of misalignment and vibration. Besides illustrating its ability to work with angular and linear misalignments, the report also shows how the compliant cable system can function when both the robot and work object are vibrating. In addition, principles are presented for other larger cable systems, providing much higher force capability for heavier work objects (payloads). This is also true of extreme vibratory motions which can be overcome through the use of a compound compliant cable system shown as a double universal joint.

Another feature illustrated with the engineering model is the principle that the compliant cable device can be adjusted in place to respond in a linear fashion or in an extremely nonlinear fashion. Further, the system can act with little damping or with heavy damping beyond critical damping as desired.

The engineering model compliant device has four Linear Variable Differential Transformers (LVDTs) mounted on it to sense deflection. The output of these sensors, when connected to an oscilloscope, permits remote

steering of the robot held device during mate-up with the work object. Further, it is shown that the LVDT signals can be changed into a digital format and handled by either a digital or an analog system:

Figure 1 illustrates the action of the compliant device. The robot is mounted to the lower two bolts and the work object is controlled by the action of a tool mounted to the upper two bolts. Thus the upper two bolts could be mounted to a screw driver, a clamp, a socket wrench, etc. Because the compliant device of Figure 1 can easily move in the six degrees of freedom, it is possible to adjust to the surface of the work object and perform its function. The bending motions of the cable to meet these requirements will be illustrated in the figures that follow.

Figure 2 illustrates the side motion of the compliant cable device. The cable has not only moved in the horizontal plane toward the thumb in the picture but the cable is also rotated because the force is not applied to the center of restraint of the cable system.

Figure 3 is an example of the compliant device under compression. Notice that all of the cables are bending down but none of them are pivoting about the swage points.

Figure 4 illustrates the motion of the compliant device that is activated by a combination of rotational and compression forces.

Figure 5 shows the same compliant device subject to tension forces. The system is so designed that the resistance to compression and tension exhibit the same force deflection curves.

Figure 6 is a close-up macro-picture of the cable in shear. This particular cable is 7 x 7 x 3 IWRC regular lay. It is quite flexible when compared to the common cable used for lifting and fits into small places that regular cable cannot.

Figure 7 is ure of the bending of 7 x 19 IWRC right regular lay preformed cable severe loading conditions. Notice that the cable handles the loae well with no yielding or falling apart of the cable stranding.

Figure 8 is aresis curve of a compliant device subject to an ever increasing load.ice the damping which causes an offset in the zero point. Also nothat the stiffness increases as the load increases. This feature of compliant device allows a robot to approach a work object softly, buen stiffness is needed for torque or motion, it is there since the g constant increases with deflection. It is this ability of the cant device to get stiffer and stiffer under a load that allows it to tery high overloads.

Figures 9a and 9how a complete hysteresis motion from a strong force on right, 9a, to a ng force on the left, 9b. The little squares on the compliant device ar inches each. This device can adjust itself to ±1/2 inches in every dition. An important factor about these compliant devices is that they able to adjust themselves to a motion of 1/2 inches in any plane with equrestraint in all three planes.

Figure 10a and 10b re two pictures of the same configuration where one is stiff, 10a, and oth is quite compliant, 10b. The stiffness curves are shown on Figure 17. rge robots can be designed to adjust the compliant attachment in space. us the compliant device can be very compliant when it makes contact but chages its stiffness before it starts to work.

Figure 11 is a robot demonstrator to illustrate the points previously discussed. Through various controls on the far side of the demonstrator, the compliant device (so marked and previously described) can move vertically, laterally, in and out and rotate in one plane as indicated on

the picture. The cylinder in the foreground is equipped with a bolt and nut facing the compliant device. The compliant device can access the nut and rotate it off the stud. It can further contact and capture the nut, even though the compliant device is misaligned with the bolt and nut assembly. The handle to the right of the aluminum cylinder is used to move the nut back and forth while the socket wrench on the robot's compliant device captures the nut and turns it. This ability of the compliant device to be soft for adjustment and stiff for turning has been previously discussed. The ability of the socket on the end of the compliant device to capture a moving nut is most useful. It should be repeated here that both the robot and work object could be moving in different modes, yet the compliant device will allow the socket to capture the nut.

Figure 12 shows the same robot demonstrator from the top. The nut is clearly shown. The compliant device is shown along with the socket wrench on the end of the compliant device. The rotating controls (operated by hand) are shown on the right. They give the compliant device motion in all three planes while rotating. It should be mentioned here that all of these motions could be controlled with servos if needed.

Figure 13 shows the nut at an angle of 7 1/2° from the line of the robot and the compliant cable device. It is possible with this compliant device to adjust itself to this angle while it rotates and turns the nut up on the bolt.

Figure 14 shows how far a socket wrench can be from the center line of the nut and still grasp the nut and turn it. Figure 14 also shows the LVDT's mounted inside the compliant device to measure the deflections.

Figure 15 shows a close-up of the socket wrench as it is in the process of latching on to the nut (previously shown on Figure 14). Note that the compliant device rotates and translates into a position where it can slide over the nut and turn it.

Figure 16 shows a double compliant device similar to a universal joint in a car. With this device it is possible to meet very large displacements and still not lose the power to operate.

Figure 17 shows the stiff curve of Figure 10a and the very compliant device of Figure 10b. Note the large amount of damping with the soft device and the small amount of damping with the stiff device. As previously discussed, it is possible to adjust the compliant device while operating and change from an extremely flexible device that will adapt itself to large alignments and prevent hard contacts during coupling. When the robot is firmly attached to the work object, the compliant device can be stiffened and respond almost the same as a solid and rigid bar.

SUMMARY

Robotic systems need compliance to connect the robot to the work object. The cable system illustrated here offers compliance for mating but can be changed in space to become quite stiff. Thus the same system can do both tasks, even in environments where the work object or robot are moving at different frequencies and different amplitudes. The adjustment can be made in all six degrees of freedom, translated in or rotated in any plane and still make a good contact and control.

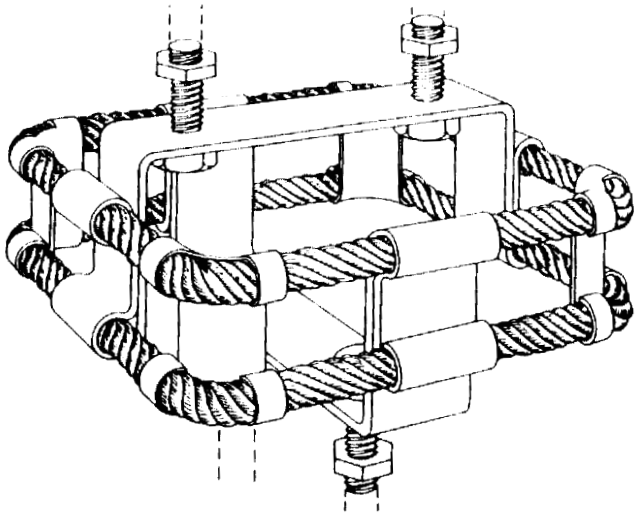


FIG.1 TYPICAL COMPLIANT DEVICE

FIG.2 SIDE MOTION

FIG.3 COMPRESSION

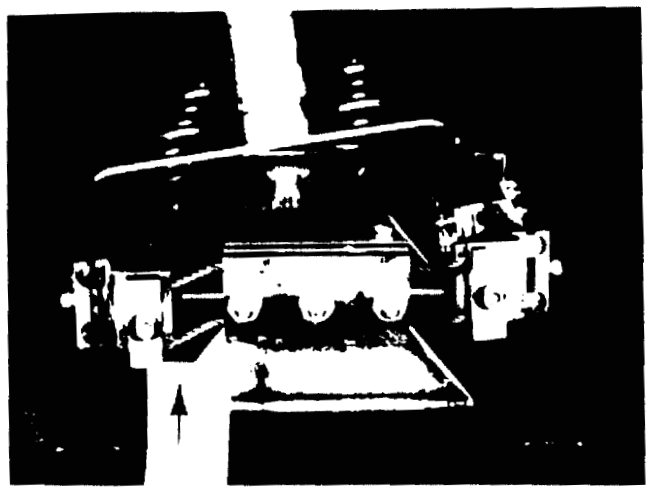


FIG.4 BENDING

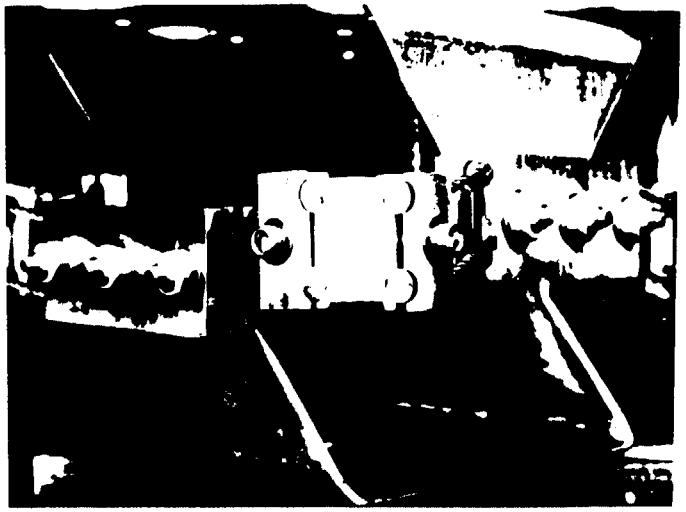


FIG.5 TENSION LOAD

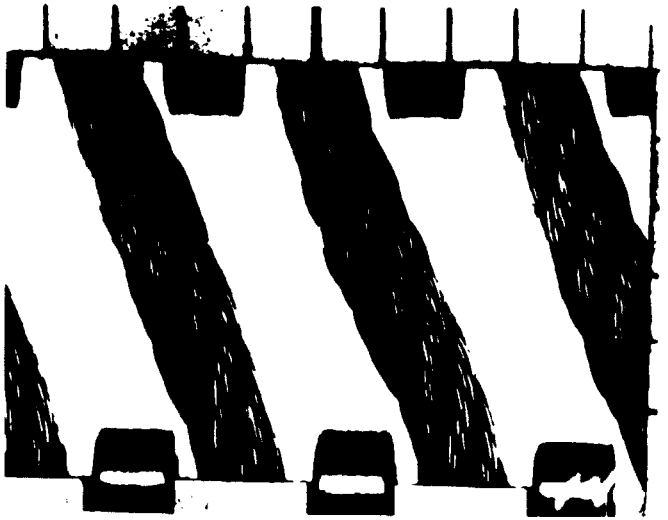
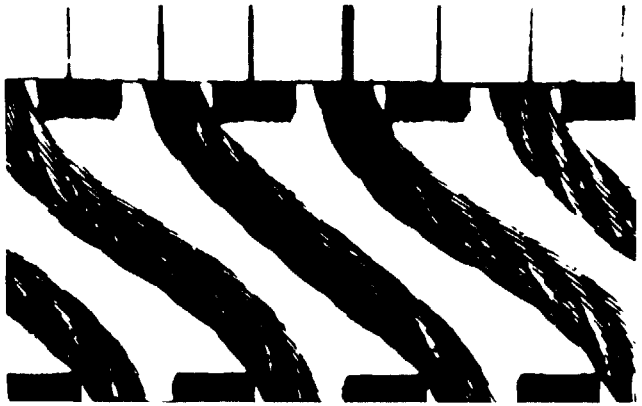
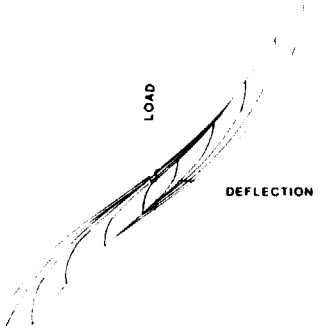


FIG.6 7X7X3 CABLE BENDING





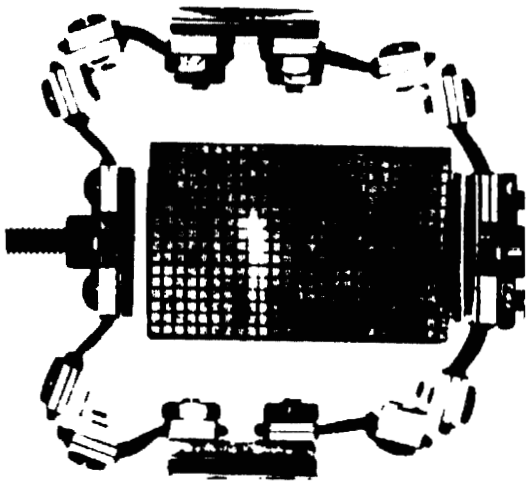


FIG.9-A PULL TO RIGHT

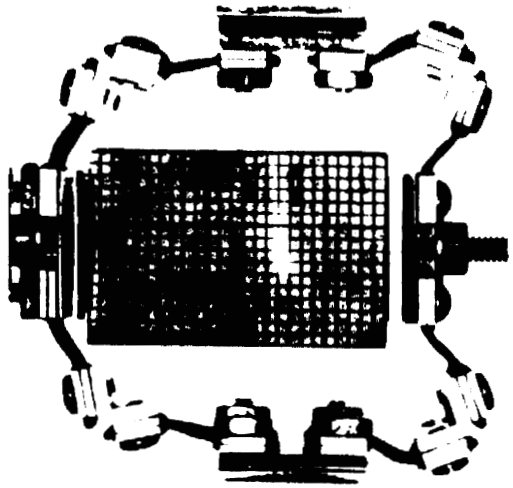


FIG.9-B PULL TO LEFT

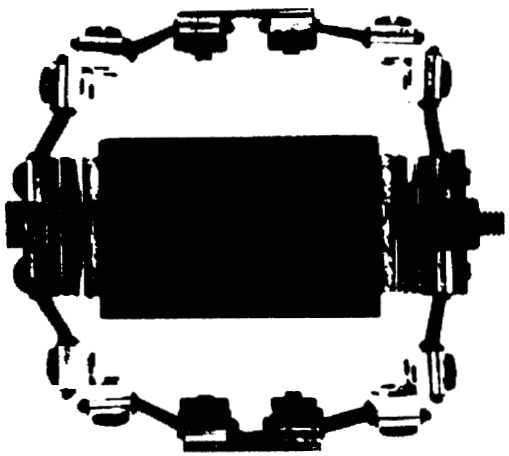


FIG.10-A STIFF CONFIGURATION PULL TO LEFT

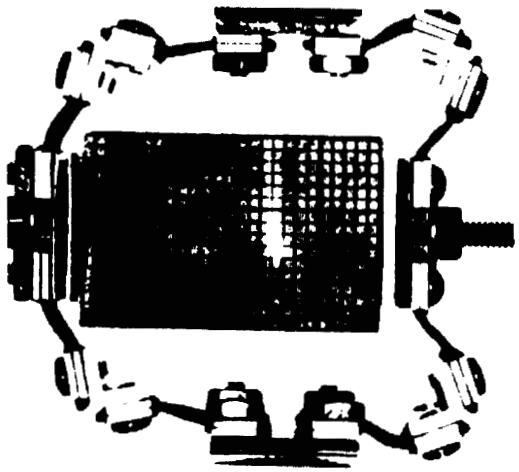


FIG.10-B PLIANT CONFIGURATION PULL TO LEFT

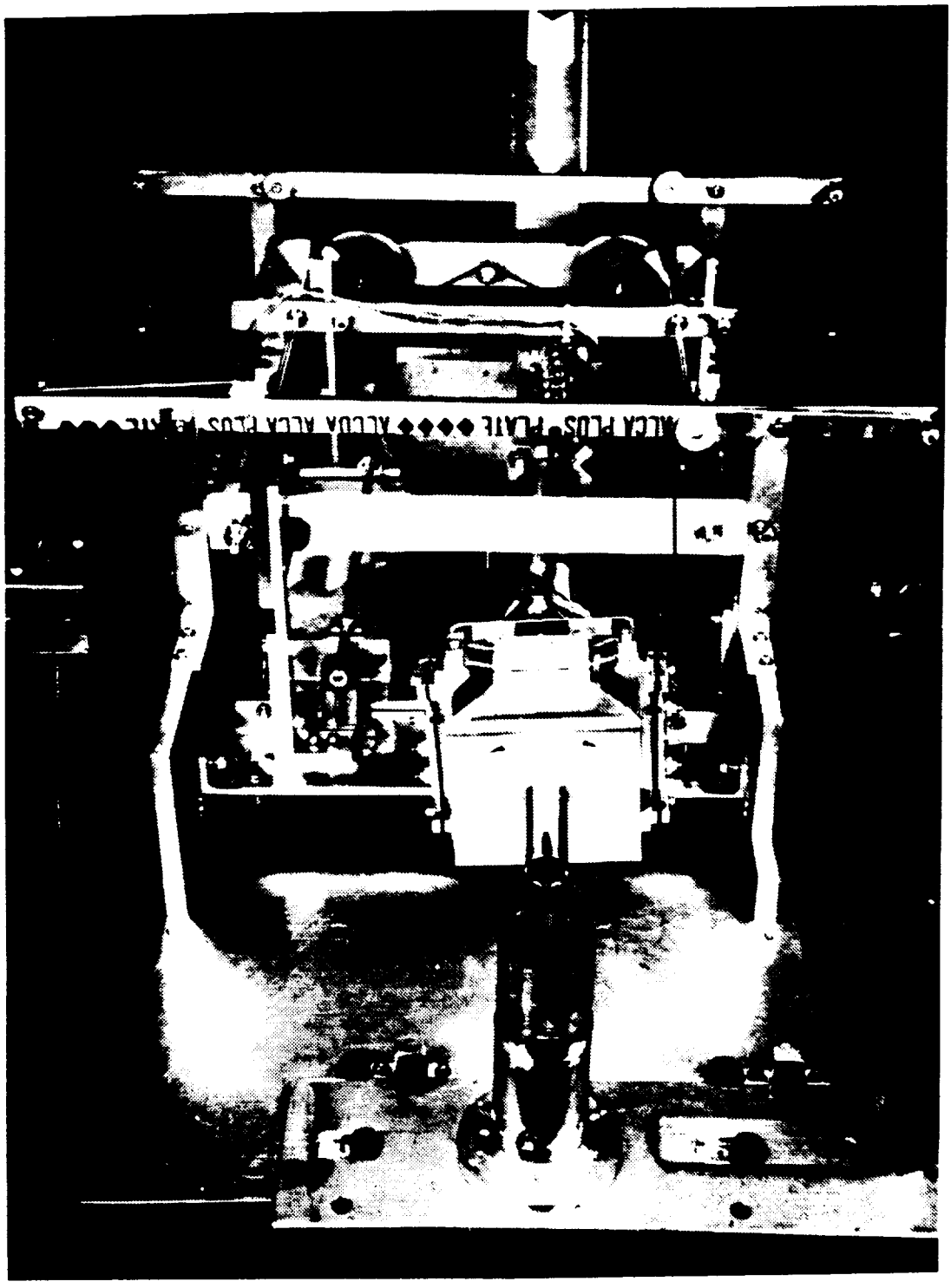


FIG. 11 COMPLIANT TESTING & CALIBRATION MACHINE

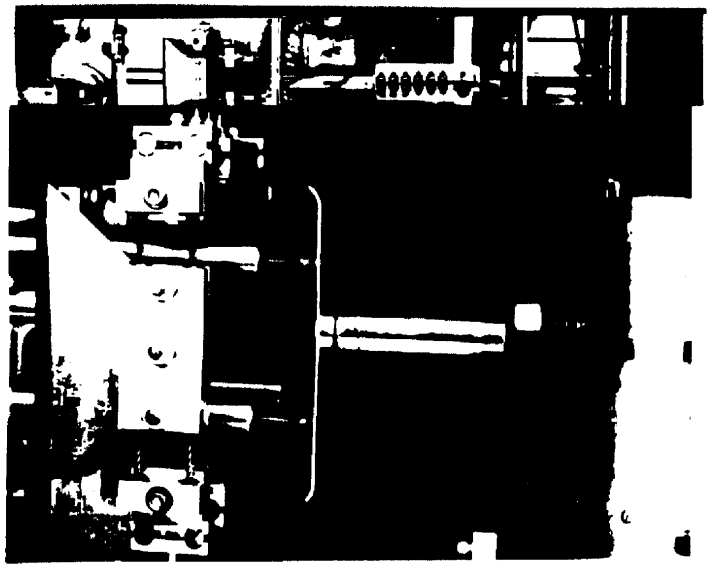


FIG. 14 COMPLIANT DEVICE APPROACHES NUT

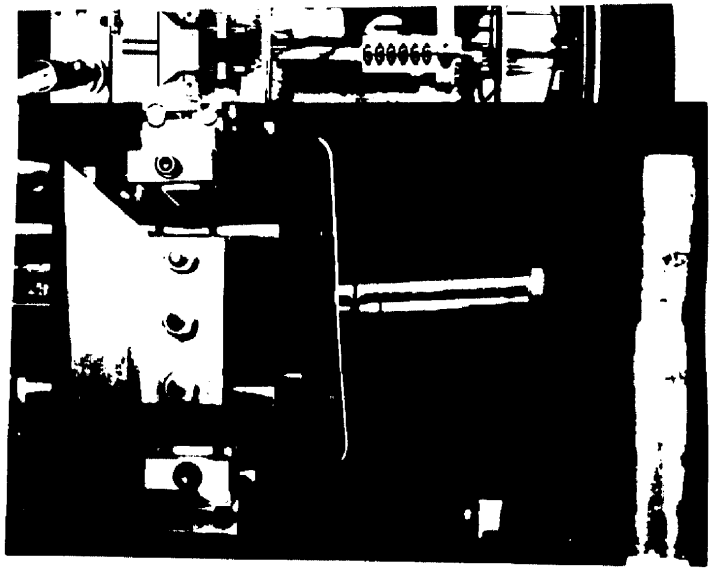


FIG.15 COMPLIANT DEVICE LATCHES ON

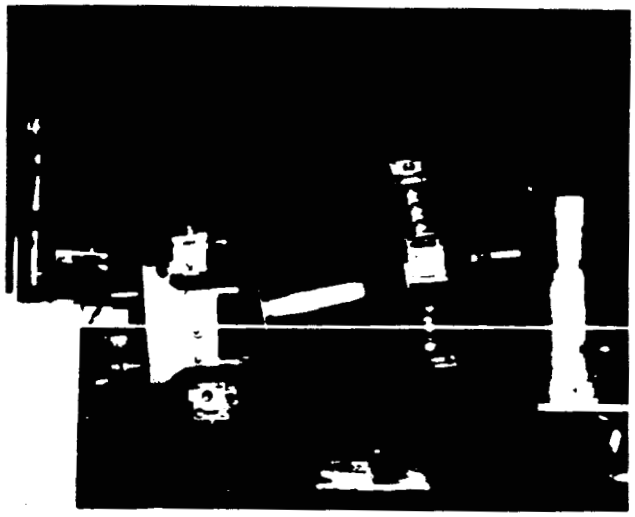


FIG. 16 TWO SECTION UNIVERSAL ACTION

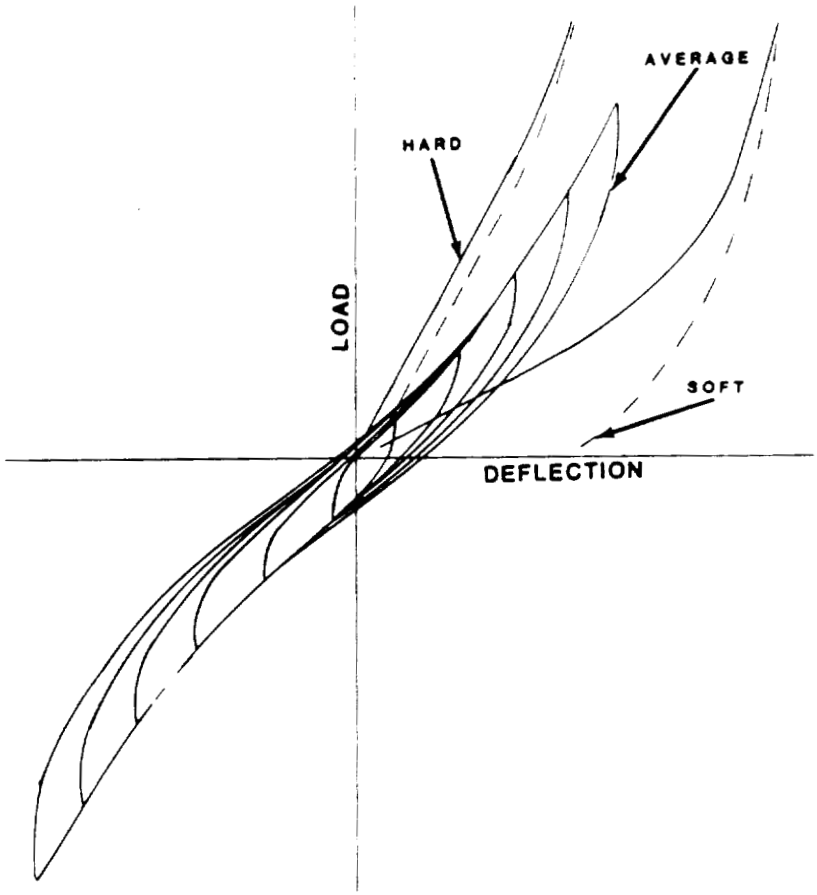


FIG. 17 HYSTERESIS CURVES FOR FIG. 10-A & FIG. 10-B

1987 GODDARD CONFERENCE ON SPACE APPLICATIONS OF
ARTIFICIAL INTELLIGENCE (AI) AND ROBOTICS

Computer Hardware and Software for Robotic Control: The Kennedy Space Center (KSC) Robotics Applications Development Laboratory (RADL)

Virgil Leon Davis

May 14, 1987

N89-10103

ABSTRACT

KSC has implemented an **integrated system** that coordinates state-of-the-art robotic subsystems. It is a sensor based real-time robotic control system performing operations beyond the capability of an off-the-shelf robot. The integrated system provides real-time **closed loop adaptive path control** of position and orientation of all six axes of a large robot; enables the implementation of a highly configurable, expandable **testbed for sensor system development**; and makes several smart distributed control subsystems (robot arm controller, process controller, graphics display and vision tracking) appear as intelligent peripherals to a supervisory computer coordinating the overall system.

INTRODUCTION

Robotics technology is a rapidly advancing field moving from applications on repetitive manufacturing processes toward applications of more variable and complex tasks. Current directions of NASA design for the Space Station and other future spacecraft is moving toward the use of robotics for operational, maintenance and repair functions while the systems are in orbit. These spacecraft systems will eventually require processing through KSC for launch and refurbishment.

In the future, KSC will be called on to design ground processing facilities for new generation launch vehicles such as the Heavy Lift Launch Vehicle and the Second Generation Shuttle. The design of these facilities should take advantage of state-of-the-art robotics technology to provide the most efficient and effective vehicle processing.

In addition to these future needs for robotics technology expertise, it is readily apparent that robotics technology could also have near-term applications to some of the existing hazardous and repetitive Shuttle and payload processing activities at KSC.

BACKGROUND

Launch site applications of Robotics to hazardous and repetitive Shuttle processing activities will offer some unique opportunities at KSC. Commercially available robots traditionally have not allowed an easy and effective means to integrate sensors with robots in the formation of flexible control systems. Without this capability, it is very difficult to develop a system in which **robot motion** can be **controlled adaptively** in **real-time**. This real-time adaptive control is the necessary **tool** for performing **tracking** of a **Shuttle vehicle** stacked at the launch pad while it is **rocking in the wind**, in order to dock and insert umbilicals (consisting of a ganged connection of electrical and cryogenic/hypergolic fluid lines) without damage to the vehicle and without hazardous leaks.

Present "T-0" Umbilicals have to be connected during excursions caused by firing of the main engines in case of an abort (which has occurred twice already) prior to ignition of the Solid Rocket Boosters

(SRB). Since it presently takes from 14 to 34 hours to reconnect various size umbilicals, there is not adequate time to safe the vehicle by draining off hazardous fuels, unless the umbilical remains connected until the Shuttle starts climbing skyward. If disconnection of these large mechanisms is done improperly, damage to Shuttle tiles or structural members could result. An orderly/controlled disconnect just prior to launch, rather than during launch, with the capability to rapidly and precisely reconnect, is the desirable approach KSC is investigating for the design of future launch vehicles. Until now such a design has been technically unfeasible, but with the advent of "peg-in-the-hole" robotics technology, high speed pipelined vision processors and real-time software control algorithms; the integration of these technologies should enable this **30 year old goal** to be accomplished.

PURPOSE

In addition to remote mate/demate of umbilical mechanisms, there are other hazardous, time consuming, labor intensive ground processing functions at KSC that could benefit from cost savings brought about by enhanced safety, productivity and efficiency through the utilization of advanced robotics technology. Therfore, a Robotics Development Team was established at KSC to determine the most feasible approach to "capture" the technology and to provide for implementation of a highly configurable, expandable, testbed capability to **perform robotics research and development.**

The team's initial objective was to develop a robotics laboratory at KSC that would provide a facility for training engineers in the unique characteristics and many disciplines involved in robotics technology. It was also to provide a facility where testing of robotics technology can take place to develop the feasibility of applying advanced technological solutions to current Shuttle/payload ground processing activities.

The ultimate objective of this research will be to extend the lessons learned and techniques/systems developed to support existing ground systems, and to further the development of similar systems for future ground servicing of advanced launch-vehicles/payloads. Some of these ground operational enhancements could also be applied to space operational systems.

Our approach was to develop, procure and install an applications development laboratory in which robotics hardware, actuators, end-effectors, algorithms, software, sensors and control systems will undergo conceptualization, development, evaluation, and checkout using a large scale test article.

For these reasons, KSC specified a **Robotic Development Prototype System** with the requirements of:

(1) providing real-time closed loop adaptive path control of position and orientation of all six axes of a large heavy lift (90 kilogram) robot,

(2) providing a sensor based testbed,

(3) coordinating and integrating state-of-the-art robotic subsystems through the use of a reconfigurable/expandable control and monitor system, and

(4) allowing operations beyond the capability of an off-the-shelf robot through a universal development system for varied applications.

RESOLUTION METHODOLOGY

The Robotic Development Prototype System contract was performed by ASEA Robotics, Inc. (New Berlin, Wisconsin and White Plains, New York) in conjunction with Adaptive Automation, Inc. (South Windsor, Connecticut). These companies had previously worked together to provide some unique systems for closed loop robotic control and sensor system integration. The delivery, installation, service and acceptance testing of the robotic equipment was managed by ASEA. Adaptive Automation performed system integration design and software development. They did an excellent job fulfilling specification requirements, designing the system to:

(1) exceed performance requirements,

(2) ensure that it would not become outdated by virtue of obsolete technology by allowing future performance capabilities to be added to the system as new technology becomes available; and

(3) use structured software modular techniques, allowing efficient and easy integration of new sensor technology.

In order to aid the systems developer in the formulation of his proposal, the KSC specification not only pointed out the type of robotic equipment NASA intended to procure, but informed him of the overall development plan for the use of the prototype equipment. The plan (Refer to Figure 1) was to **procure "off-the-shelf"** state-of-the-art robotic hardware and "intelligent" feedback control systems and to marry this hardware and software **with KSC developed work cells** incorporating sonic, infrared and tactile feedback sensors, optical transmission devices, hypergolic and cryogenic fluid couplings, and various end-effector gripper devices. It was later decided that vision control for "lines" management would not be done with an object recognition system, but would use standard KSC camera systems to enable an operator to monitor and safe the system in case of entanglement of cryogenic or electrical lines.

As delivered, Item I hardware (**Industrial Robot Arm Control**) consists of a heavy lift, servo controlled robot arm mounted on a 30 foot track, an arm controller, a teach pendant, special maintenance tools, and grippers. An identical set of Item I equipment, minus the track, was delivered to the subcontractor, Adaptive Automation, as a rental unit to allow them to perform software developmental integration

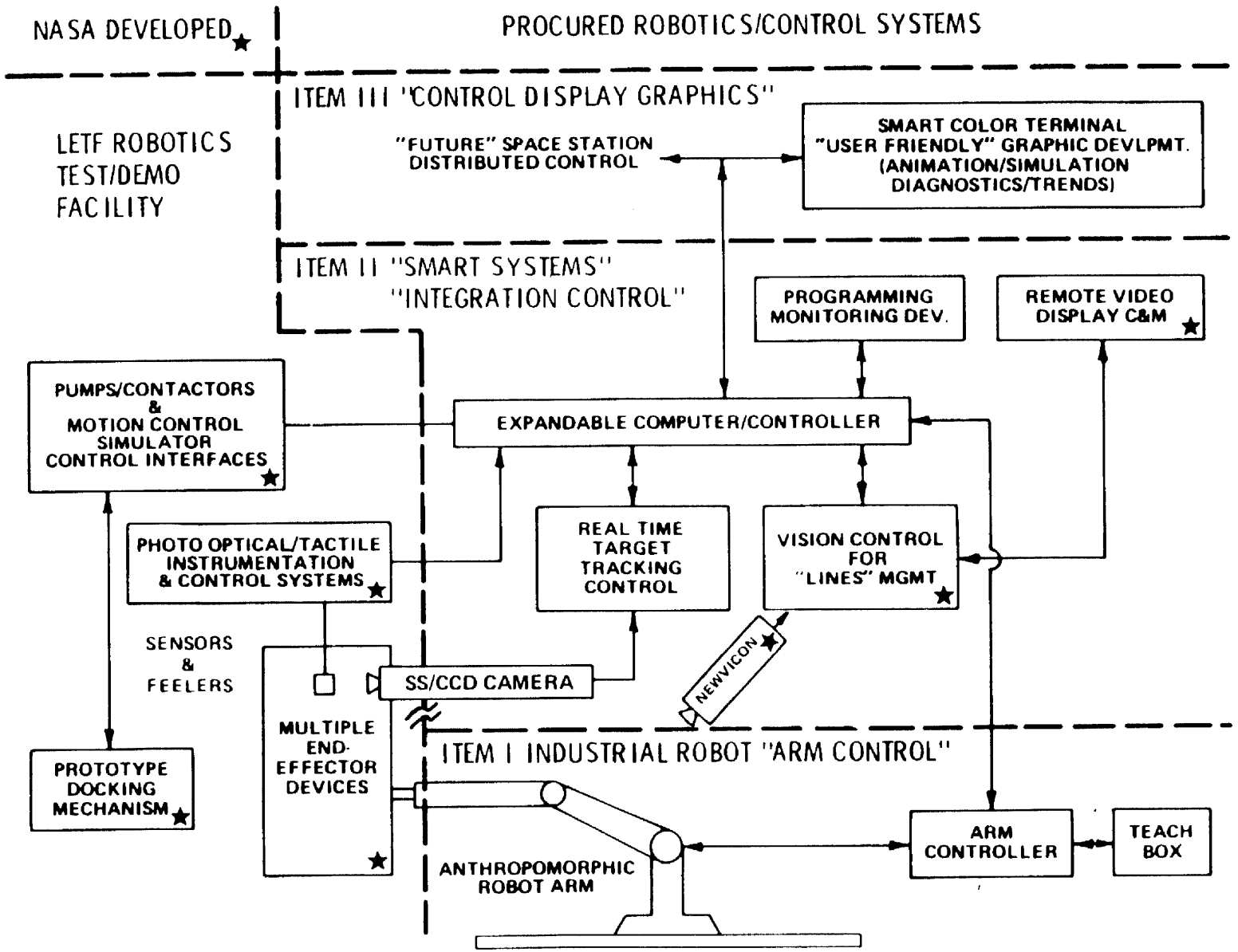
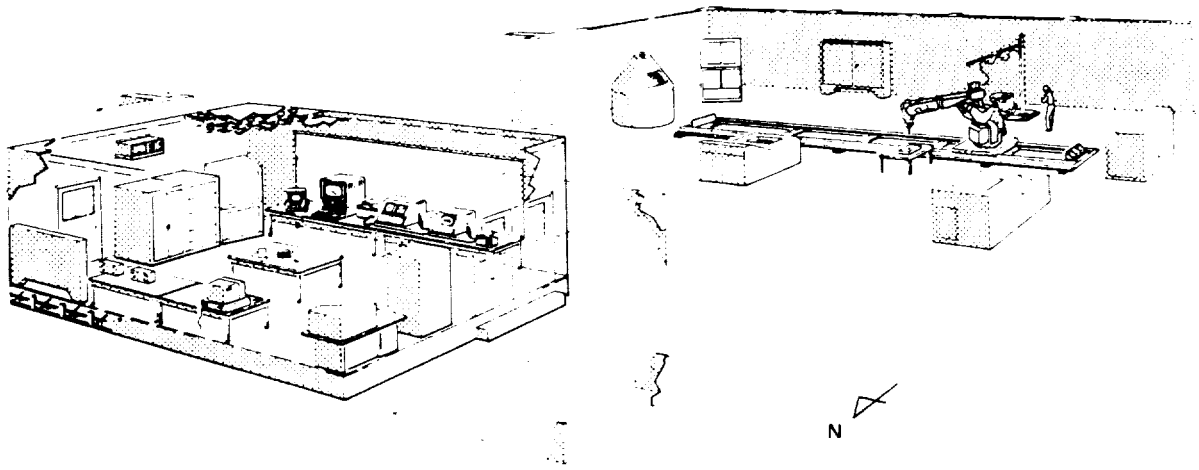


FIGURE 1. SYSTEM CONCEPT

with Item II & III equipment while simultaneously allowing NASA to gain valuable experience with the robot prior to delivery of the overall system. Item II hardware (**Smart Systems Integration**) consists of a solid state camera, a vision processor, a programmable process controller, software maintenance terminals, and a MicroVAX Supermicrocomputer. Item III hardware (**Control Display Graphics**) consists of a smart color terminal, an alarm/report message printer, and a video hardcopy color printer.

All equipment was installed at KSC prior to acceptance testing. Training on Item I equipment took place at the factory in New Berlin in December 1985. Also that month, the robot arm, track and robot controller were delivered to the Kennedy Space Center in Florida, but not installed into a high bay of the Launch Equipment Test Facility (LETF) until January 1986. Hazardous work on the Space Telescope Transporter preempted the high bay delaying acceptance testing and preliminary utilization of the stand-alone robot for several months. The Smart Systems Integration computer/controller and the Control Display Graphics equipment was delivered in September, 1986. During September and October, Items II & III equipment underwent installation into a control room built by NASA, acceptance testing was performed on the total integrated control systems, and training of NASA engineers and support contractor personnel was completed.

The area in the high bay of the LETF where the Robotic Development Prototype System was installed is now know as the **Robotics Applications Development Laboratory** (**RADL**). Refer to Figure 2.



Robotics Applications Development Laboratory (RADL)

Figure 2

The RADL is unique in that:

(1) a large robot travels on a track enabling it to access several different work cell applications. The system is **highly reconfigurable to adapt to various prototype configurations**, making it a general purpose, multi-station, research and development testbed;

(2) the robot is **integrated** through a computer and software system **to several smart distributed control subsystems**:

(A) vision controller for tracking,
(B) process controller for work cell integration and
(C) a smart graphics display terminal for coordination of the overall network; and

(3) the laboratory **permits sophisticated control algorithms and signal processing techniques** to be applied to sensor information processing, allowing for applications that currently can not be automated without the use of advanced sensor systems.

The initial thrust of the RADL will be to develop the systems and techniques required for automated loading and unloading of hypergolics for space vehicles and payloads during prelaunch ground operations. Future tasks undertaken by the RADL will be to extend these automated techniques to other fluids (such as cryogenic) as well as electrical power, fiber optic communication, and data system mate/demate functions. As the expertise of the robotics engineers increases, and as application requirements dictate, the capabilities of the laboratory will be increased to include equipment for three dimensional scanning, higher order image processing, artificial intelligence, sonic, laser and other ranging systems, tactile systems, and mobility systems.

COMPUTER HARDWARE AND SOFTWARE DESIGN OVERVIEW

ASEA Robotics Inc. (ARI) and Adaptive Automation Inc. (AAI) were very responsive to the NASA requirements of providing "real-time adaptive servo control & feedback mechanism integration." We are not discussing "adaptivity" which concerns the control dynamics of a robot arm relative to different weights being handled. Our use of "adaptive control" implies the ability to "adapt" to real world changes as determined by sensory devices on and around the robot. The delivered system provides a **set of hardware and software building blocks** as a foundation for a general purpose sensor development testbed for robot control. The computers are **expandable** to allow for the future needs for infrared photo-optical, vision or tactile feedback devices. For instance, the MicroVAX provides the computing power required for the sensor processing and the interfacing hardware expandability necessary for future requirements; the Programmable Process Controller provides the present need for 248 analog and digital I/O signals and has I/O expansion capacity of over 1,000 signals; and the Vision System contains state-of-the-art computerized tracking system hardware, readily expandable for future development needs.

In addition to hardware, AAI provided software modules to support a base-line capability from which the system could expand. The objective was to provide an open, **flexible** and expandable system, but one which was programmable at a high level. System operational **software** was provided in the form of **libraries of subroutines/modules** which can be used by a NASA application programmer to allow "high level" programming of the system without requiring the programmer to be familiar with the low level functioning of the communication procedures between the various subsystems.

This **modular hardware/software design** approach provides: ease of performance enhancement, alteration with minimal impact, and stand-alone or integrated mode functions.

INDUSTRIAL ROBOT ARM CONTROL

To acomplish early development tasks, an anthropomorphic robot arm having electric motor drives was specified. The decision was made to go with electric motor drives to avoid potential leaks of hydraulic fluids which could cause damage or replacement to Shuttle Orbiter tiles or cause contamination by venting oil vapors. The arm was specified to be a servo-controlled mechanism since it had to be an integral part of a closed loop tracking system. We also wanted to take advantage of some common characteristics of **servo-controlled robots** which include: smooth motions, controlled movement of heavy loads, flexibility of manipulation and accurate/repeatable end-of-arm positioning.

ANTHROPOMORPHIC ROBOT

The robot delivered was an **IRB-90/2** manufactured by ARI in the United States. It is capable of lifting 90 kilograms and holding it 3,000 mm from its base having a repeatability of 1 mm under constant operational conditions at maximum reach and load over a large working range. All arm joints (axes) are actuated by direct current servo motor drives with closed loop feedback control through resolvers and tachogenerators. The robot is moved using simple hand motions, plain dialogue and self-instructing commands through a joystick mounted on a programmable function panel (teach pendant). The processor controls are a Motorola 68000 based system incorporating both American and Swedish technology. An extended 21K RAM memory is backed up by floppy disk and a battery providing 400 hours of memory storage. The robot controller comes with a standard adaptive control function which provides control (in local tool reference frame) of tool position only (orientation can not be controlled). The adaptive control can only operate with three analog or digital signals simultaneously. This is not enough capability for 6 dimensional (3 directions and 3 orientations) adaptive control of a sensory development testbed. However, the robot has the capability to have all 6 axes controlled from an external computer. Therefore, the robot controller's optional "external computer link software" was used to implement the task of communicating with the MicroVAX to provide the extra 3 degrees of freedom necessary to incorporate advanced docking stratagies being developed by NASA at the RADL (Refer to Figure 3).

Figure 3

The use of an **off-the-shelf** robot/controller with an integrated computer control system provides several advantages:

(1) **Development time and money is reduced** by not having to specify and build a specially designed robot to act as a tracking mechanism.

(2) The robot controller can perform kinematic/dynamic matrix transformations using its internal processor capability and thereby **free up the MicroVAX computer for supervisory tasks.**

(3) It can be used as a **"stand alone"** system **or** integrated with other computerized control systems in a **distributed network.**

ROBOT TRACK

In order to make the laboratory a **multi-station developmental testbed**, the robot was placed on a track. The track was subcontracted to ESAB North America Robotic Welding Division in Fort Collins, Colorado. The RADL installation was the first use of an IRB/90 on a track and the installation was completed without degrading the robot's repeatability performance. The track uses an electric motor acting as a seventh axis enabling the robot to service various workcells. Several experiments can be located along and around the 9 meter (30 foot) track. Sensors will be used along the track as inputs to the integrated control sytem to define areas where obstacles must be avoided. Also, if there is a problem with one experiment, the robot can simply be moved along its track to another work station. This flexibility increases its efficency and eliminates the necessity of purchasing additional robots for each developmental project. Since the smart systems computers enable adaptive control of position, velocity and orientation of all six axes, the robot can be made to simulate a spherical or cartesian robot. NASA can determine tolerances required, robot characteristics, and control system strategies necessary for a prototype system prior to releasing a specification for an aerospace application, therefore, saving time and money.

SMART SYSTEMS INTEGRATION

The integrated computer hardware and software subsystems were specified as 2 basic blocks: a "Computer/Controller" (or a combination) and a "Real-time Target Tracking Controller." They were to allow servo control, tracking, and feedback mechanisms integration providing the capability for:

(1) **supervisory coordination** of various "smart control systems,"

(2) **I/O hardware interfaces** for integration of work cells that NASA will implement to control peculiar tasks such as prototype docking motion control simulation mechanisms, and

(3) "Adaptive Path Control" of docking mechanisms through **"real-time visual feedback."**

The Computer/Controller function was satisfied by 2 separate computer systems operating in an integrated manner. The systems are a Digital Equipment Corporation (DEC) MicroVAX II supermicrocomputer and an ASEA MasterPiece 280 Programmable Process Controller. The target tracking was supported by the MicroVAX interfacing with a hybrid system made up from a Motorola 68010 computer controlling DATACUBE vision processor hardware. The delivered system provided a baseline capability to demonstrate the functioning of target tracking. NASA is currently upgrading the hardware and software for advanced tracking development.

SUPERVISORY COMPUTER

The supervisory computer is the heart of the RADL system integrating the various "smart" subsystems, allowing them to talk to one another and making them appear transparent to the user. AAI chose to implement the supervisory computer using a **DEC MicroVAX II computer** configured with a 70 Mb hard disk, 2 Mb of RAM memory, a 95 Mb tape unit and a Q-bus with 9 serial ports and one parallel port. The system was configured with the Micro VMS operating system because it provided the best combination of supporting a multiple user and a multiple process environment, while providing relatively good real time response. All software was developed in the VAX C programming language because it supports a structured, high-level programming environment while providing low-level "bit manipulation" necessary for control. This hardware/software combination offers a wide range of potential product enhancements, meets stringent throughput requirements and provides a system that can be easily documented and maintained.

SYSTEMS INTEGRATION SOFTWARE

Software for operation, demonstration and acceptance testing of the integrated systems was required to be delivered to the Government for all furnished subsystems in the form of modular subroutine libraries. Software was required to be easily programmable and to be developed in a top-down, structured manner with sufficient annotation to allow clear understanding of its operation. Diagnostic software programs were required to verify operational status of the communication links to the various subsystems, to enable debugging and to allow troubleshooting of the integrated systems. AAI fulfilled those requirements by providing 9 major computer system software functional modules:

(1) **Operator interface modules** provide easy of use menu driven displays that allow command visibility, descriptive terminology and operator prompts. A status window, located in the lower portion of the screen, displays any messages in understandable phrases.

(2) **Configuration file processing modules** contain parameters that need to be changed often by the operator. These modules reside in the MicroVAX and allow both flexibility and ease of operation. There are several parameter files including robot, vision, closed loop control, programmable controller, and graphic display parameters. They are extremely "user friendly" "text" files and can be read, printed, rearranged and easily modified.

(3) **Robot communication module software** provides sophisticated real-time target tracking robot position motion command functions that allow for direct control of all 6 axes of the robot arm's velocity, orientation and position. To ensure that the robot controller is never waiting for a motion command from the MicroVAX, a second command is transferred to the robot controller before the motion of the current command has been completed. A specially developed math library allows sensor positional information to be transformed into a "quaternion" representation for use by the ASEA robot controller. Also provided are robot communication functions for operator tasks such as upload of robot programs from the robot controller, download of robot programs from the computer, change of the current robot mode, synchronization of the robot and monitoring of robot status.

(4) **Vision system communication modules** support a master/slave relationship with the MicroVAX being the master and the vision subsystem the slave. These modules maximize the throughput rate by minimizing the length of commands and responses, ensure data integrity through parity and checksum techniques and allow for expansion of vision functions.

(5) **Programmable process controller communication modules** allow individual data items as well as groups (data sets) of functionally similar items to be transferred to/from the MicroVAX. The communication protocol is an ASCII protocol designed by ASEA.

(6) **Simulation modules** provide performance data (obtained during any target tracking experiment) which can be archived onto the MicroVAX disk and transferred ("played back") to the color graphics display subsystem through the programmable controller subsystem. The robot's position and current tracking error can be examined in more detail and viewed repeatedly in "simulation" mode.

(7) **Exception handling modules** enable the operator to immediately determine the cause of an exception and to take the appropriate action. All error conditions are displayed in the system status window through the use of simple, readable messages. Exception handling modules are separated by their causes and by their level of severity to ensure that errors are detected, correctly classified, and properly handled.

(8) **Diagnostic modules** aid the operator in identifying hardware problems and in monitoring system performance. An extensive set of diagnostic routines have been written to examine all communication between the MicroVAX and the other system components and to store normal/abnormal performance data for display.

(9) **Closed-loop control modules** provide real time 2-D tracking control of the robot arm using coordination between the MicroVAX, vision subsystem, and robot controller. The vision subsystem calculates target error information every 33 milliseconds, the MicroVAX closes the loop using PID (proportional, integral, and derivative) feedback and sends a new motion command to the robot approximately every 90 milliseconds. Algorithms in 6-D are presently being formulated to perform dynamic umbilical mate/demate.

PROGRAMMABLE PROCESS CONTROLLER

The NASA specification identified the need for a **dedicated control processor** with flexible programming and ease of expansion. It even identified the amount and type of I/O support required. AAI determined that a Programmable Process Controller (PPC) would provide a cost effective solution to work cell integration while offloading the supervisory computer for more time critical tasks such as interfacing the robot and vision control systems. Sensors need to be interfaced directly to the MicroVAX's Q-bus only in time critical situations. The PPC can incur all the overhead involved in processing input/output (I/O) signals and can transfer only exception data or requested application display information to the MicroVAX.

AAI selected a **MasterPiece 280 PPC** manufactured by ASEA Industrial Systems (AIS). It is a Motorola 68000 based system (similar to a smart Programmable Logic Controller) providing logic control, process control, data handling, and PID functional capabilities. Work cells will be interfaced to the robotic systems through the PPC to provide closed loop control of each test apparatus. Overall systems display information will be processed by the PPC to a "slave" color graphics display system. The distributed control approach is evident in that the PPC will do what it can do best (process all input data and control routines efficiently within 50 milliseconds) and the graphics system will do what it can do best (process and display real-time performance data to a color screen), each system sharing duties and offloading processor functions from one another.

Programming of the MasterPiece is done using function blocks (a "higher level" programming method than relay ladder logic). Programming is accomplished by a **MasterAid 214** system which is a portable Motorola 68000 based system that has its own display screen, keyboard, and floppy disk drive. The MasterAid and associated printer is used during program execution to provide real-time display of the internal programs and for troubleshooting. It is used during program development for off-line design and debug.

PROGRAMMABLE PROCESS CONTROLLER SOFTWARE

The only **application** program requirements specified by NASA, were to receive data from the MicroVAX and transfer it to the color graphics display system. The application software developed to support the RADL color graphics display was separated into functional modules, generated on the MasterAid, implemented on the MasterPiece PPC and displayed on the color graphics CRT. Data sets were defined to group similar types of information onto one screen for quick access, straightforward clarification, and for an overview of the latest configuration of control parameters. The programs provide the following displays:

(1) A **tracking grid display** reads previously recorded tracking error information and robot arm positions, and dynamically replays this data onto a multi-colored grid to depict tracking error. Scaling is

variable and grid areas are dynamically highlighted to illustrate the difference between camera position and target location.

(2) A **robot status display** provides a graphic representation of the robot's current position on the track. It also provides robot controller status information such as the positions of the robot axes, the robot's operating mode, robot programming information and diagnostic data.

(3) **Data set displays** provide information concerning the system's serial **communication parameters** between the MicroVAX, robot, vision and programmable process controller. The data set entries contain such information as baud rate, number of data bits, and port number.

(4) Other **data set displays** provide information concerning current **closed loop control parameters** of both the robot and vision systems. These data sets contains entries such as PID constants, robot scale factors, camera position, and the time period for the robot arm to move in its approach to the target.

NASA is developing more application programs to integrate new test cells as they come on-line.

REAL-TIME TARGET TRACKING CONTROLLER

The NASA specification emphasized the importance of this subsystem to provide rapid and precise control of the robot arm. It was required that the system be a real-time servo loop consisting of a small solid state camera, mounted on the robot's end-effector, which views a docking target and uses centroid error signals to process command signals to servo controls in order to make the end of the arm track a moving mechanism. The specification stated, "During docking tasks, the arm will be commanded to near full-extension and tracking-control processing will be initiated. Therefore, the servo loop will mostly involve wrist movements but may involve minor elbow movements. After identification of the target, the vision control system will only involve processing of target tracking errors. These simplifications, together with simplified centroid or **equivalent target location** calculations, will eliminate much of the arithmetic and the discrimination operations which slow down most "vision-control" systems, to enable it to provide "real-time" position control. After the end-effector is "locked" onto the target, distances and angles will be determined by either the vision control subsystem or later augmented by NASA developed photo-optical, laser or tactile devices; and integrated with the Target Tracking Controller and the Arm Controller, through the Computer/Controller, to initiate final insertion sequences." Tracking performance tolerances were not specified since NASA intends to develop docking mechanism requirements. The solution was to obtain a low cost system that provided the **necessary tools** to determine the distance, tolerance and compliance capabilites required for the design of remote umbilical mechanisms. NASA also plans to use the system to provide a **technology base** for future developement of advanced tracking control capabilities for other applications.

Again, AAI satisfied requirements quite well by implementing the vision system with the highest power vision processing boards available at the time, interfacing them with a high performance computer, and developing general purpose modular software to support a real-time system while allowing the flexibility to support a wide range of future vision applications.

TRACKING SYSTEM COMPUTER

AAI selected the **Motorola System 1000** as the vision system computer interfacing a DATACUBE image processing board set through a VME bus. The System 1000 is configured with a Motorola 68010 10 MHz processor, a 15 Mb hard disk, a 512 Kb RAM memory, a 655 Kb floppy diskette, 3 serial ports, and 1 parallel port. The vision system is supported by the Motorola VERSAdos operating system: a real-time, multiuser, multitasking operating system with features necessary for the support of the image processing boards. These features provide servicing of directly connected interrupts, intertask communication, system utilities, memory allocation and task management services.

IMAGE PROCESSING HARDWARE

The RADL vision system uses four boards selected from the **DATACUBE MaxVideo** line of image processing products. They use a pipeline design approach providing a high performance image processing capability with the flexibility to accommodate more modules without impacting the capability to process images at the scan rate of the camera (30 images per second). The initial image processing configuration uses the following hardware/firmware boards:

(1) DIGIMAX - An **image digitizing board** performing A/D and D/A conversions at a 7.16 MHz rate from a standard RS-170 video signal. The analog input signal is software filter selectable and conditioned with programmable gain and offset circuitry. It provides graphics overlays, dynamic input multiplexing and transparent switching of Input and Output Look Up Tables.

(2) VFIR - A **pipelined linear signal processing board** for time critical processing at 144 million arithmetic operations per second. It performs a 3 X 3 convolution operator to the image to enhance its edges. A full frame of video data is processed in much less time than the 1/30th of a second it takes the camera to scan the image.

(3) FEATUREMAX - A **feature extraction board** that counts the number of occurrences of many different events and stores their x and y coordinates in a 64 Kb block of memory. The board also provides histogram recording of the locations of up to 16 K features. The feature extraction board receives the enhanced edge image and records the coordinates of every point in the image that has a value higher than a preset threshold value. The Motorola computer uses this data to calculate centroids by vector summing the xy coordinate pairs and dividing by the total number of pairs.

(4) FRAMESTORE - An **image storage board** containing 3 (384H x 512V pixel) frame storage buffers to hold digitized video images. It is used to provide a window (mask) which is gated with the data output from the pipelined linear signal processing board to reduce the "area of interest" processed by the feature extraction board. It also draws a cross hair on the operator's TV monitor to allow him to view what the vision system is calculating as the center of the target.

TRACKING SYSTEM SOFTWARE

Firmware on the image processing boards was integrated with Motorola based software modules developed to control and monitor target tracking tasks. Their modularity allows them to be used later in different combinations for future image processing tasks. The tracking system modules provide the following functions:

(1) **System initiation modules** allocate shared memory blocks for intertask communication, allocates system queues to allow transfer of messages between tasks and loads/starts the other vision system tasks.

(2) **Command processing modules** examine commands received from the operator's terminal which can set vision system parameters and stop the target tracking task. They are essentially message processor modules invoking routines to initialize, request status, set/request parameters, start and stop target tracking and set the area of interest window.

(3) **Vision system communication modules** implement system protocol with the MicroVAX. AAI developed a protocol in which one MicroVAX command generates continuous vision system responses. This mode transfers target tracking coordinate information from the vision system to the supervisory computer. Termination can be by either the MicroVAX, the operator or a vision system error. An additional "window" command allows the MicroVAX to dynamically control the size of the camera view, as the distance from the camera to the target changes.

(4) **Target tracking modules** compute the spatial derivatives of the image, determine the location of significant edges and determine the centroid of the edges.

SOLID STATE CAMERA

A Charged Coupled Device (CCD) camera was provided with software offsets to enable remounting on various end-effector devices. Since target edge data is used to determine the location of the target, the camera is equipped with an auto-iris lens to provide compensation for variations in lighting. The vision system provides target location information in a plane perpendicular to the line of sight of the camera.

CONTROL DISPLAY GRAPHICS

The RADL control room houses all computer control equipment with the control and monitor (C&M) devices positioned along a 5 meter picture

window (Reference Figure 2) overlooking the robotic test area. Devices available there are the MasterAid C&M CRT for the PPC, a TV monitor for display of camera and tracking system data, a DEC VT220 terminal for online C&M of the MicroVAX and tracking vision systems, and a smart color terminal. NASA has added a DEC VT240 terminal for offline programming of the vision system and is installing control panels for work cell integration and a remote video display C&M panel. The video panel will provide black and white video displays and joystick control of 4 high-contrast cameras placed around the outside of the robot test area. A color stereo camera mounted on the shoulder of the robot will send color data to a 3-D monitor providing a display image for depth perception. The various CRT terminals provide a **"bird's-eye view"** for programming and troubleshooting of both the supervisory MicroVAX computer, the Motorola vision computer and the programmable process controller. The smart color terminal is a focal point for demonstration purposes providing a **"big-picture" display** of the overall process.

SMART COLOR TERMINAL

The smart color terminal is an interactive, high speed, color graphics CRT which provides operating personnel with **real-time status of the processes** under their control. It permits operator interaction in a timely and responsive manner through displays which include: process graphics with color coded status/control parameters, process diagnostics using color and blink for ease of interpretation, emergency and alarm conditions for fast corrective action, and exception data for real-time statistical analysis. The need for this device is to reduce the display software "intensity" of the various work cells and subsystems being controlled.

The color graphics display system chosen was the **MasterView 820** which is designed to interface with the MasterPiece PPC system. It is configured with a Motorola 68000 based processor, memory, floppy disk, 19" color display unit, color "frame-grabber" printer, battery backup and keyboards for operation and display generation. The MasterView system is specifically designed to provide user friendly graphics development for overall systems status, exception data, diagnostics, simulation and trend displays.

All software required to build and configure **user displays** is included with the system. The system includes a packet of standard displays which can be easily configured by the user: 6 overview displays containing 10 groups each with 10 objects, 60 group displays with 10 to 100 objects each, 7 types of object displays, 20 trend displays, 10 to 20 application specific displays and event and alarm lists. Special displays can be rapidly set up from a choice of preprogrammed items (pumps, valves, special symbols, etc.) or can be "drawn" by a person with little programming knowledge via the system's line drawing and text creation capabilities. AAI provided a tracking error demonstration program and other application displays (refer to PROGRAMMABLE PROCESS CONTROLLER SOFTWARE). NASA is providing display graphics in accordance with future work cell development.

ONGOING RESEARCH IN THE RADL

The integrated RADL system is **currently providing an easy to use testbed for NASA sensor integration experiments** and successfully fulfilling its initial target tracking requirement (Refer to Figure 4). Advanced target tracking development is in progress concerning the mating of umbilicals used during space vehicle launch. Programmatic studies are underway to use the laboratory's capabilities to **enhance the safety, productivity and efficiency of KSC facilities** for Shuttle and Space Station ground processing operations.

VISION SYSTEM UPGRADES

For the delivered system, the robot must be positioned such that the target is entirely within the field of view for the tracking function to perform, target identification or object recognition is not performed, and orientation control is not provided; but the capability is available within the integrated systems. **Future system expansion** to provide these capabilities is presently in progress. Newly developed image processing boards (for implementing real time large kernal operations) and enhanced software (for more robust, noise free, reliable edge detection) are being installed. At the same time, a faster processor (Motorola 68020) and a new VME backplane is being installed **to accommodate the latest special purpose hardware.** These new boards will provide real-time determination of the centroid of multiple targets and will allow discrimination between many different targets. Software is currently being developed to utilize 4 dots to determine position, distance and orientation. This will enable **upgrade** from 2-D **to 6-D tracking** control.

EXISTING KSC APPLICATIONS

Two robots have been developed at KSC: a small pneumatic control robot to test Electronic Security System cards and an Electrostatic Robotic Test Cell (ERTC) to measure electrostatic charge retention on nonconductive materials. The ERTC was installed in an environmental test chamber at KSC and has increased measurement repeatability, accuracy and productivity in a program inspecting thousands of material samples.

Robotic work cell development applications at KSC are currently focused on tracking and docking development, remote umbilical plate mate/demate, large connector/QD development, hazardous panel operations, and end-effector/gripper development. A graduate student is working with NASA contractor personnel on the development of **orientation control algorithms** utilizing vision data based on changes to the shape of a circle. Florida Institute of Technology is performing **end-effector research** based on previous NASA concepts developed at Langley Research Center and at Marshall Space Flight Center, as well as some innovative concepts of their own. Automated Dynamics Corporation has been given a Small Business Innovation Contract to develop a computer controlled "Universal End-effector with Torque Feedback" for the operation of hand

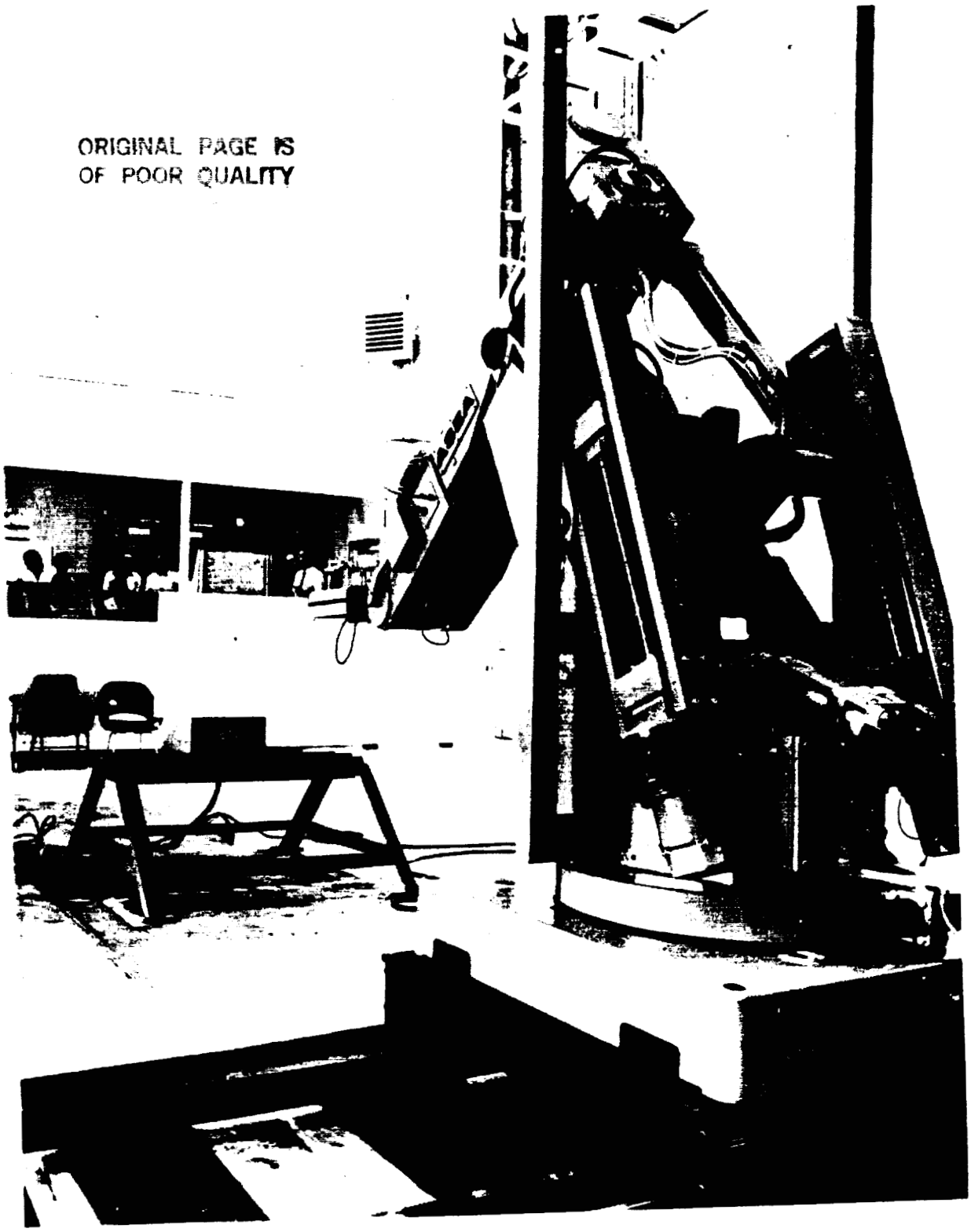


Figure 4

valves in hazardous environments. NASA is working with scientists at the Controlled Ecological Life Support System facility at KSC to **develop robotic techniques** for Plant Growth Chamber automation which may eventually aid extraterrestrial crop production.

ROBOTIC APPLICATIONS UNDER REVIEW

Studies on hazardous, time critical and labor intensive problems peculiar to KSC are being conducted for several applications. **Automation and robotics studies** are being performed on Space Station ground processing facilities. The use of **mobile robotics** for security, fire fighting and hazardous spill operations is being investigated. Robotic techniques to improve "Shuttle Orbiter **payload inspection** and closeout verification" (operations involving possible damage to payloads with expensive "return from Pad" consequences) are being investigated. Non-destructive test sensors, vision systems and various kinds of distance ranging sensor systems can be integrated with the RADL systems to develop the **prototype concepts for integrating robot parameters with large data based graphics** and **artificial intelligence** (AI) **software** systems. For instance, the RADL robot can position a sensor with precise accuracy, report that position and orientation, provide distance sensory data and integrate machine vision "electronic photographs" with graphics and AI software to furnish computer printouts providing automatic sizing and highlighting of exception data. This type of system is being proposed for a number of possible projects such as **nondestructive testing** for Solid Rocket Booster joint and seal verification, Shuttle Orbiter radiator **damage inspection**, Orbiter tile damage/debonding assessment and Orbiter **contour measurements**. The manual methods employed presently in these operations are very labor intensive and produce expensive serial-time flow problems.

SIGNIFICANCE

Implementation of the computer hardware and software systems in the Robotic Applications Development Laboratory system at KSC is for the **development and application of advanced robotic control technology.**

KSC not only launches spacecraft, but services these spacecraft on the ground: designing the support equipment, launch accessories and computer hardware/software for ground spacecraft servicing. Several of the technologies undergoing development in the RADL have similarities to **autonomous control, docking** and **refueling** tasks being developed for Space Station and satellite servicing applications.

Large operational cost savings are possible through the integration of advanced technologies for ground processing operations such as Orbiter tile and radiator damage assessment (as described above in ROBOTIC APPLICATIONS UNDER REVIEW). The **RADL** is an **ideal test-bed** where the government can work with private and aerospace contractors to establish the feasibility of these cost saving approaches.

TELEOPERATED POSITION CONTROL OF A PUMA ROBOT

Edmund Austin and Chung P. Fong, Ph.D.

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Abstract

A laboratory distributed computer control teleoperator system is developed to support NASA's future space telerobotic operation. This teleoperator system uses a universal force-reflecting hand controller in the local site as the operator's input device. In the remote site, a PUMA controller receives the Cartesian position commands and implements PID control laws to position the PUMA robot. The local site uses two microprocessors while the remote site uses three. The processors communicate with each other through shared memory. The PUMA robot controller was interfaced through custom made electronics to bypass VAL.

In this paper, the development status of this teleoperator system is reported. The execution time of each processor is analyzed, and the overall system throughput rate is reported. Methods to improve the efficiency and performance are discussed.

1.0 INTRODUCTION

It is hoped that in the future mankind will inhabit space on a permanent basis. Whether it be military crews operating space based offensive/defensive facilities or civilians living in space colonies, the habitation of space will require large amounts of construction and repair in space. For such tasks, it is preferable not to use astronauts since radiation hazards would limit the amount of time one person could be allotted EVAs (Extra Vehicular Activity). Also, there are some tasks where safety considerations would preclude using a man at all. In these situations it would be preferable to use some type of robotic device to achieve one's objectives.

Basically there are two types of robotic devices to use: autonomous and teleoperated. Autonomous systems require no human assistance to accomplish their task. After being informed of the task to perform, the autonomous system executes the task either from some preset repertoire of tasks or uses some type of artificial intelligence to determine how to tackle the problem. This requires that you have one or more of the following: 1) a very large set of pre-defined tasks to cover any and all eventualities, 2) a very good model of the environment, 3) a very powerful artificial intelligence computing capability, or 4) a good method of incorporating sensor data. This is a partial list of autonomous system requirements and still they may be difficult to meet.

An alternative is a teleoperated system. A teleoperated system is basically a robotic device that is remotely controlled by a human operator. So we will depend upon a human mind to determine how to tackle a task. Still, for a human operator to properly direct a robot he will need sensors not only to provide him with a view of what the robot is doing, but also a feel for the forces the robot is both exerting and experiencing.

In order to study some of these teleoperation issues we have constructed a teleoperation system consisting of a force reflecting hand controller, a Unimation 560 robot, and five National Semiconductor microprocessors. The microprocessors are needed to perform kinematic transformation and data communication since the FRHC (Force Reflecting Hand Controller) and robot are kinematically dissimilar and physically separated. With this system we will attempt to determine what defines "good" teleoperation and how to improve it.

2.0 SYSTEM ARCHITECTURE

The computing hardware of our system consists of five National Semiconductor 32016 CPU development boards with N.S. 16081 floating point unit and an 10 megahertz clock (except the FRHC control CPU which has a 6 megahertz clock), two BLC-519 I/O boards, two 128 kilobyte RAM boards, and some ancillary electronics to interface with the FRHC (Force Reflecting Hand Controller) and the Unimation PUMA 560 robot.

All of the computing hardware is contained in two Multibus Chassis. See Figure 1. More specifically, the various components perform the following tasks:

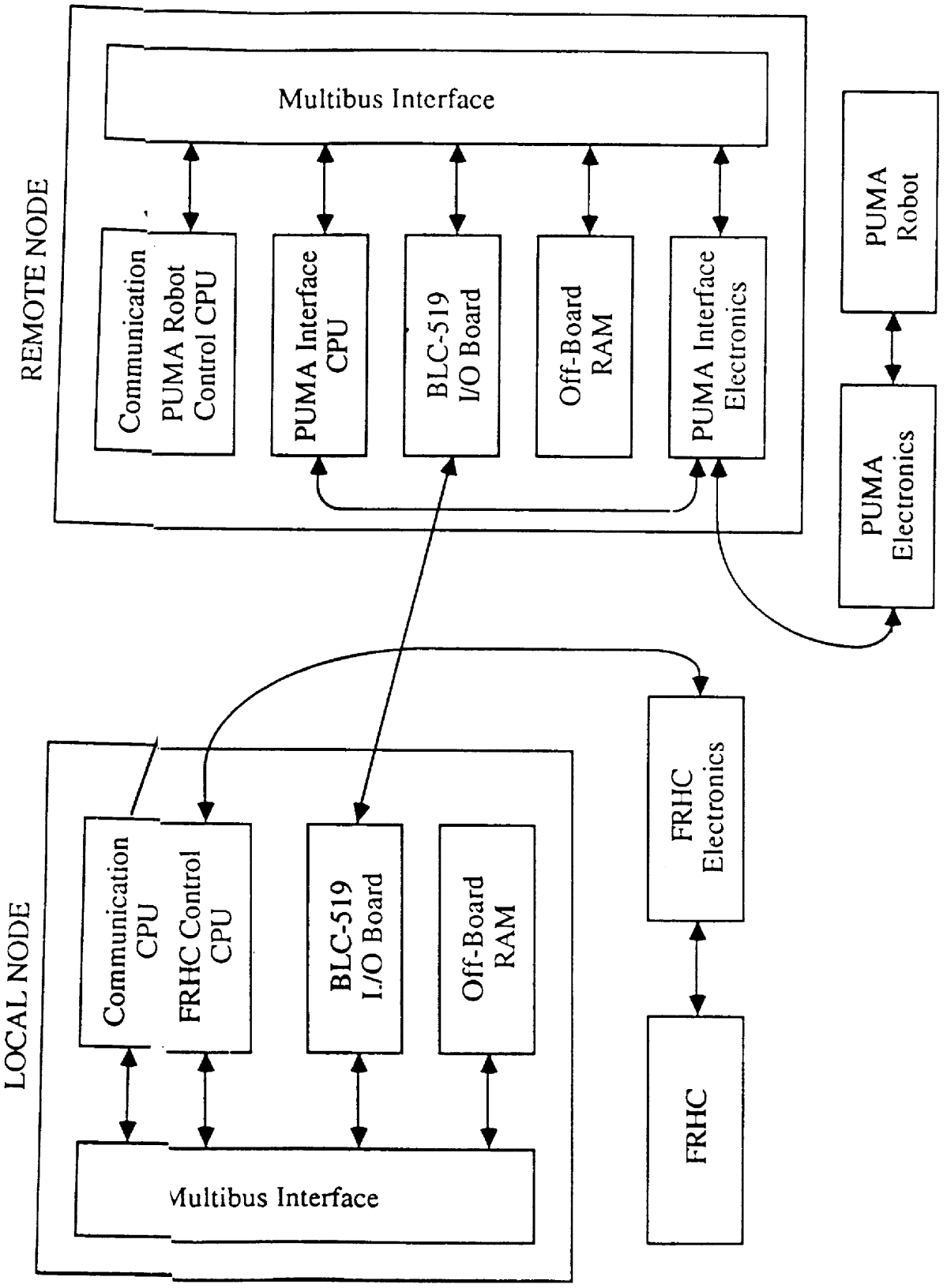


Figure 1. System Architecture

1) Two communications CPUs, one located in each chassis (node), that decide what data to send, retrieve that data from off board RAM, assemble that data into a buffer and send it over a parallel link. Conversely, when a communication CPU is receiving, it determines what data it is receiving, and places that data in the appropriate locations in off board RAM. The local communication CPU also contains the menu through which system parameters can be altered.

2) One FRHC control CPU that interprets the encoder values of the FRHC and converts them into joint angles, from these joint angles the Cartesian position and orientation of the end of the FRHC is determined (i.e. the FRHC T6 matrix), for position control. For rate control, the deviation of the FRHC from some neutral position is used to generate rate commands. This CPU also calculates the force feedback to backdrive the FRHC, whether we are in the rate or position mode.

3) Two BLC-519 I/O boards that have the 8 bit parallel I/O ports that the communications CPUs actually use to send and receive data between the remote and local nodes.

4) Two BLC-0128A 128 kilobyte off board RAM boards, that are used to hold all information that is shared between processors within a node and to hold all information shared between nodes. Each specific piece of stored information is held at a specific address known to all the CPUs.

5) The PUMA robot control CPU does the inverse kinematics and the forward kinematics of the robot, along with compensation for an end effector, and workspace transformations.

6) Actual interfacing with the robot is accomplished by the PUMA interface CPU. It sends joint angle commands to the robot, reads the current robot joint angles, calibrates the robot, and performs the setpoint interpolation.

7) The PUMA electronics provides the servo power to the robot and has six Motorola 6503 joint microprocessors that actually perform the low level robot servo control.

8) FRHC electronics allows us to read the potentiometers of the FRHC's six joints and to supply current to the motors attached to the six joints for force feedback.

9) The PUMA interface electronics facilitates direct communication with the six 6503 joint microprocessors.

PUMA Interface Electronics

Our interface design approach is to by-pass Unimation's LSI-11 resident VAL-II and to simulate Unimation's interface to the DRV11 in the PUMA arm controller by using two Intel 8255 Programmable Peripheral Interface(PPI) adapters. Port A, B, and C on the PPI are used for data input, output and handshake, respectively. Mode 1 is selected as the port mode. To read or write the data/commands from/to the joint processors, a request

signal hasent through handshake line. The data sampling rate and the position commandng rate is hence dependent on the high level processor cycle rate.

The l servo commands issued from high level processor are decomposed into digital semands acceptable by PUMA joint processors. The low level servo commands ardy the following four routines coded in "C":

a) read_int, command, data) - reads encoder/current value, depending on the commm the specified joint.

b) read_ommand, data) - reads an array of encoder/current values from all six joints

c) write_int, command, data) - write set points/motor currents, depending on the co, to the specified joint.

d) write_ommand, data) - write an array of set points/motor currents, to all six joints neously.

Embe the same interface software routine, those four routines interpret the commands form proper handshakes to read/write the data/digital servo commands to/from thA interface.

3.0 SYSTETROL MODES

Control Moration

The eoperation system is constituted of three control loops: the supervisory control loocal control loop and the remote control loop. The supervisory control loop includperator, the visual and audio feedback, and menu-driven commands. Through mection, the operator can switch the control mode, change the control parameters onitor the control status. The local control loop consists of the force reflecting hitroller and the local processors. The remote control loop is formed by the sensors anipulator and the remote processors. High level information, such as the Cartesiions/velocities and the control mode words, are exchanged between the local and the control loops.

In the control loop, the low level servo commands generated by the remote processors ad to the PUMA's joint processors through the interface board in the PUMA contnit. Since the PUMA joint processors control the joint motors either in "position or in "current mode," the servo commands generated by the processor are also bashe above two modes. In "position mode" the joint processor accepts encoder set and implements PID control. In "current mode" the motor torque commands acted by the joint processors. Currently, the PUMA interface processor takes advant the joint processor PID control capability and sends only encoder set points to therocessors. The direct joint motor control that sends current commands

to joint processors is not currently implemented because 1) the interrupt-driven interface, which facilitates the high bandwidth PD or PID servo control, is not installed; 2) the PUMA velocity needed for PD or PID control is also not available.

All of the following three major system control modes result in the position commands which servo the PUMA arm in its "position mode":

1) Position control mode. By reading the pots from the hand controller interface, the joint angles are computed. Forward kinematics of the hand controller is then performed to determine the end-point position and orientation in Cartesian coordinates, which is commonly referred to as the T6 matrix. Upon receiving the end-point positions of the hand controller through the parallel line, the remote processor performs the inverse kinematics of the PUMA to determine the desired joint positions. The desired joint set points in encoder values are then calculated and sent to the PUMA joint processor interface. The current PUMA joint position is also read by the remote processor. From this, robot forward kinematics are done to calculate the position and orientation of the robot's endpoint. This information is transmitted back to the local node. The position error between the hand controller position and PUMA position are then computed and used to back drive the hand controller.

2) Rate control mode. In rate control mode, the hand controller is utilized as a joystick-type input device, in which the spring effect is generated by the software. The joystick's displacement from its nominal position determines the rate. By integrating the rate, the local processor yields the aggregated positions which are then sent to the remote site. A small deadband was set around the nominal position of the hand controller to ensure no position output when the hand controller rests in the neighborhood of its nominal position. Since the feedback position is not processed in this mode, the throughput is slightly faster than in the position mode.

3) Mixed mode. In this mode, the translational axes operate in rate control mode while the rotational axes operate in position control mode. This mode would allow the operator to quickly move PUMA arm in the work space while maintaining the same orientation of the hand controller.

The above three control modes are actually commanding the PUMA arm's movement, and are therefore categorized as operation modes. The following are three other non-operation modes, in which the PUMA arm is "quiescent":

1) Start mode. This is the first mode entered upon turning on the power. In this mode, the initialization, health and status checking take place.

2) Index/menu mode. Since the work volume of the hand controller is much smaller than that of the PUMA arm, one of the means to augment the hand controller positioning capability is to use indexing. This mode allows the hand controller to move to a new position while the PUMA arm is frozen. The teleoperation resumes after a key on the keyboard or a button on the hand controller is hit. Another means of augmenting the

hand controller positioning capability is by using scaling factors. The movement of the PUMA arm can be amplified to a larger motion or confined to a smaller motion by the proper selection of scaling factors in the menu. The first one is for gross movement and the second one is for dexterous positioning. The scaling factors can be applied to both position control mode and rate control mode.

3) Quit mode. This mode is entered through menu selection. In this mode, procedures including robot servo power shut off, return of confirmation messages, etc., must be exercised prior to system shut- down.

Control Mode Transition

Since the robot is frozen during index mode, it is used as a natural gateway for mode transition. After startup, the system will default into the index/menu mode. From index/menu mode, the operator can transit into any of the following modes:

a) Quit mode - By menu selection.

b) Position mode - By depressing the index button on the handgrip of the hand controller.

c) Rate mode - By depressing the index button after selecting the rate mode from the menu.

d) Mixed mode - By depressing the index button after selecting the mixed mode from the menu. The mode transition diagram is shown in Fig. 2.

When exiting the index mode, the current PUMA position is stored and the translational bias between the new hand position and PUMA position is calculated. Provided that the indexing is also applicable to the rotation, as the operator desires, the rotational bias has to be calculated. The translational vector of the subsequent position commands are then added to the stored PUMA position to yield new PUMA position commands.

By using Denavit-Hartenberg notations, the translational bias is computed as follows:

$$({}^{a0}_{h0}T) = ({}^{a0}_{a6}T)\,({}^{a6}_{h6}T)\,({}^{ho}_{h6}T)^{-1} = \begin{bmatrix} \underline{R} & \underline{P} \\ \underline{0} & 1 \end{bmatrix} \tag{1}$$

where the $({}^{a0}_{h0}T)$ denotes the transformation from a0 (base of robot arm) to h0 (base of hand controller). Similarly, $({}^{a0}_{a6}T)$ is the current PUMA position (i.e., T6 matrix) w.r.t. its own base reference. $({}^{ho}_{h6}T)$ is the hand controller's new position w.r.t. its own base reference, and $({}^{a6}_{h6}T)$ is the coordinate transformation between PUMA arm's last joint and the hand controller's hand grip. The $\underline{R}$ matrix in $({}^{a0}_{h0}T)$ results in Eq. (1) being replaced by a 3x3 matrix, which is pre-determined to correct the coordinate disparity between hand controller base and robot base. The $\underline{P}$ vector in $({}^{a0}_{h0}T)$ represents the translational bias.

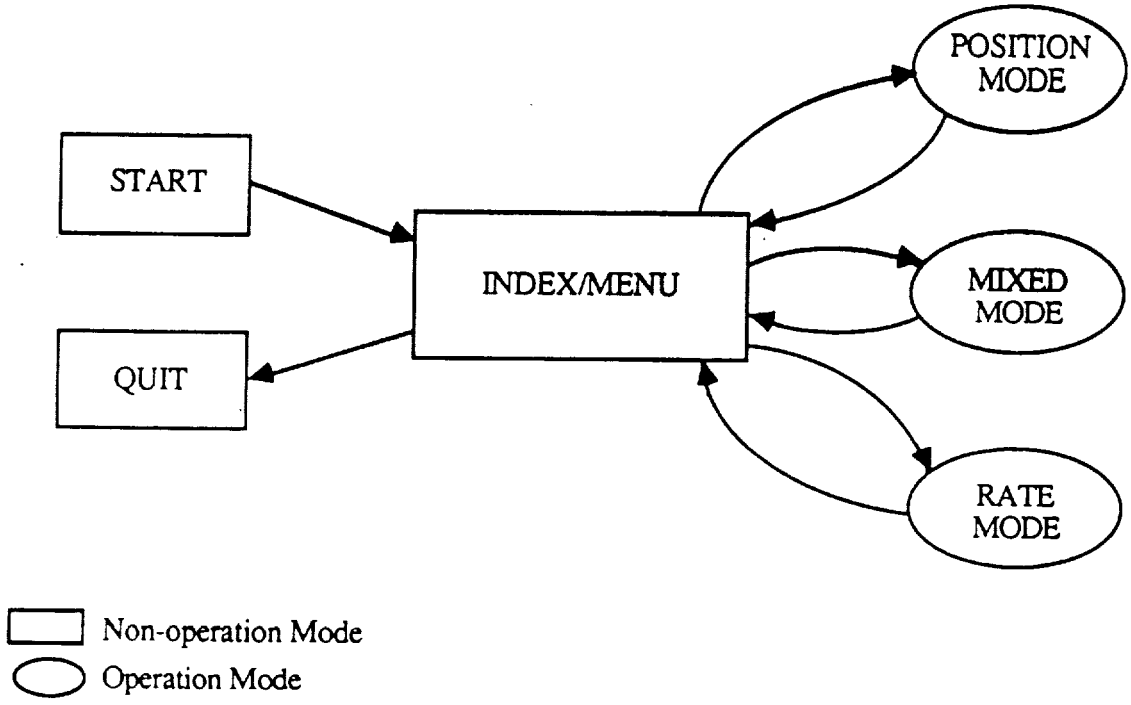


Figure 2. Mode Transition Diagram

The rotational bias is calculated as

$$({}^{a6}_{a6n}T) = ({}^{a0}_{a6}T)^{-1}({}^{a0}_{a6n}T) \tag{2}$$

where $({}^{a0}_{a6n}T)$ is determined by

$$({}^{a0}_{a6n}T) = ({}^{a0}_{h0}T)\,({}^{h0}_{h6}T)\,({}^{h6}_{a6}T) \tag{3}$$

and the $({}^{a0}_{h0}T)$ matrix is derived form Eq. (1). If the index on rotation is not desired, the $\underline{R}$ matrix in $({}^{a6}_{a6n}T)$ can be replaced by an identity matrix $\underline{I}$.

The subsequent position commands (T6) after indexing is then computed as follows:

$$({}^{a0}_{a6n}T) = ({}^{a0}_{h0}T)\,({}^{h0}_{h6}T)\,({}^{h6}_{a6}T)\,({}^{a6}_{a6n}T) \tag{4}$$

4.0 ROBOT KINEMATICS

In order to determine the position and orientation of the last link of the robot in Cartesian space, reference frames were assigned to each link of the robot, using Denavit-Hartenberg notation. See Figure 3. Six transformation matrices ${}^{i-1}_{i}T$ were then determined that relate the reference frame of link i to link $i-1$, which are a function of the joint angle of that link. Multiplying these six matrices, i.e.,

$${}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{5}T\,{}^{5}_{6}T = {}^{0}_{6}T \tag{5}$$

yields a ${}^{0}_{6}T$ matrix, a 4x4 matrix that contains both the orientation and position of the reference frame of the last link with respect to some arbitrary reference frame.

$${}^{0}_{6}T = \begin{bmatrix} R_{11} & R_{12} & R_{13} & P_x \\ R_{21} & R_{22} & R_{23} & P_y \\ R_{31} & R_{32} & R_{33} & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \underline{R} & \underline{P} \\ \underline{0} & 1 \end{bmatrix} \tag{6}$$

where $\underline{R}$ is the orientation matrix and $\underline{P}$ is the position vector. This generation of a ${}^{0}_{6}T$ given the joint angles of the robot is referred to as the forward kinematics calculation.

There is one problem with this formulation thus far and that is that the reference frame of the last link is actually embedded within the robot wrist while what is desired is the position and orientation of the tip of an end effector. See Figure 4. Therefore we must

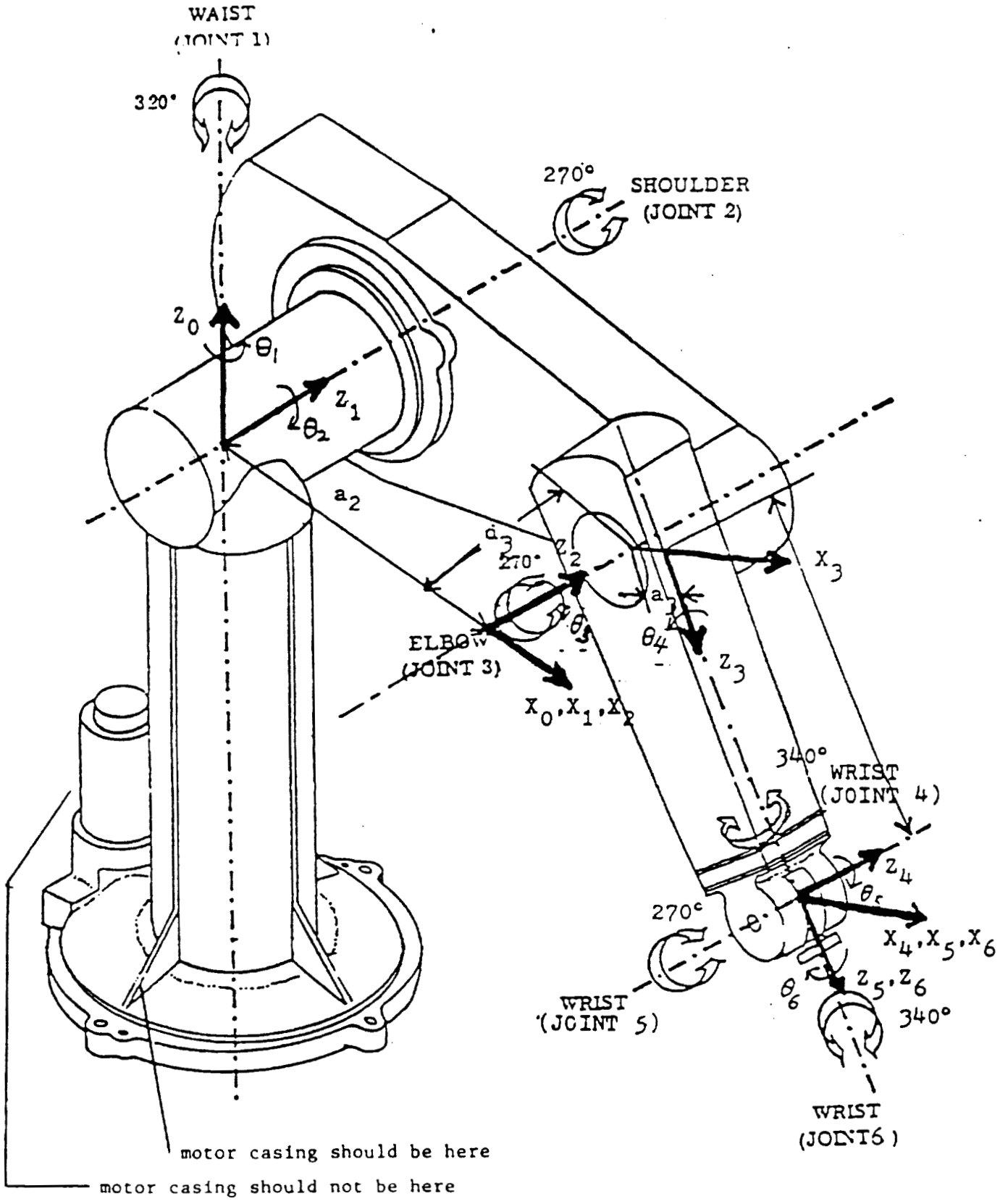


Figure 3. PUMA Robot Reference Frames

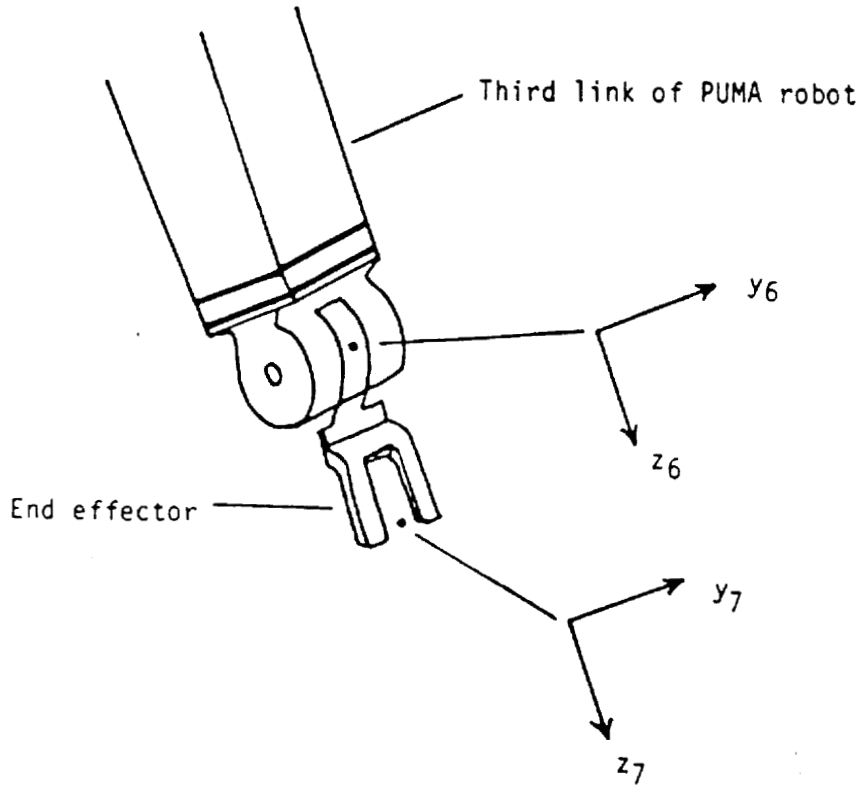


Figure 4. Robot T6 Frame Versus T7 Frame

multiply the ${}^{0}_{6}T$ matrix by a transformation that describes the position and orientation of the tip of the end effector with respect to the ${}^{0}_{6}T$ reference frame.

$$ {}^{0}_{7}T = {}^{0}_{6}T * E \quad \text{where E is also a 4x4 matrix.} \tag{7} $$

Now that we have described how to obtain the position and orientation of the robots end effector given the robots joint angles, the more difficult task is to determine the robot joint angles that will yield a specified position and orientation of the end effector tip. This is referred to as the inverse kinematics calculation. By successively premultiplying both sides of equation by the inverse of the leading term on the left hand side we obtain the following

$$ {}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{5}T\,{}^{5}_{6}T = {}^{0}_{1}T^{-1}({}^{0}_{6}T) \tag{8} $$

$$ {}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{5}T\,{}^{5}_{6}T = {}^{1}_{2}T^{-1}({}^{0}_{1}T^{-1})({}^{0}_{6}T) \tag{9} $$

$$ {}^{3}_{4}T\,{}^{4}_{5}T\,{}^{5}_{6}T = {}^{2}_{3}T^{-1}({}^{1}_{2}T^{-1})({}^{0}_{1}T^{-1})({}^{0}_{6}T) \tag{10} $$

$$ {}^{4}_{5}T\,{}^{5}_{6}T = {}^{3}_{4}T^{-1}({}^{2}_{3}T^{-1})({}^{1}_{2}T^{-1})({}^{0}_{1}T^{-1})({}^{0}_{6}T) \tag{11} $$

$$ {}^{5}_{6}T = {}^{4}_{5}T^{-1}({}^{3}_{4}T^{-1})({}^{2}_{3}T^{-1})({}^{1}_{2}T^{-1})({}^{0}_{1}T^{-1})({}^{0}_{6}T) \tag{12} $$

If we equate terms in equations (8) through (12) the angular values of the six PUMA joints can be found in terms of the Cartesian position and orientation [2]. Again, there is the problem that final solution obtained will be in terms of the components of the ${}^{0}_{6}T$ matrix, whereas what is actually specified is the ${}^{0}_{7}T$ matrix (i.e., the position and orientation of the tip of the end effector). From equation (7) we can obtain the intermediate ${}^{0}_{6}T$ matrix from the actual ${}^{0}_{7}T$ matrix by postmultiplying by the inverse of the end effector transformation so that

$$ {}^{0}_{6}T = {}^{0}_{7}T * E^{-1} \tag{13} $$

5.0 COMMUNICATIONS

While there are two distinct computing nodes, only one node, the local node, allows the user to interface with the system. It is through the local node that the user enters all appropriate data and receives any information from the system. Associated with the local communications processor is a menu. When the two nodes are communicating via the remote and local communication CPUs the user hits the "escape" key on the terminal attached to the local communication processor. This stops communication and enters the user into the index/menu mode. By now typing "m" on the keyboard, the user brings up the menu which has a hierarchical tree structure. Once inside the menu, the operator can call up sub-menus that change such system parameters as operational mode, motion scaling factor, robot end effector length, etc. All data to be shared by processors within a node as well as data to be used by processors within another node are stored in off

board RAM. When the user enters changes from the menu the appropriate data is also changed in off board RAM. Upon exiting the menu the user can re-start the inter node communication by simply hitting the "return" key on the terminal.

Since the local and remote node are located in two different chassis, a way was needed to allow the two nodes to communicate with one another. Another problem was that depending on the operational mode different data would be passed between the nodes. For example in position mode you would want the local node to send a position and orientation matrix, while in joint mode (not implemented yet) you would only want to send six joint angles.

Communication is achieved with two sets of boards, one set resides in each chassis. A communication board set consists of one 32016 CPU board and one BLC-519 I/O board. The BLC-519 has 9 eight bit parallel communication channels and uses the Intel 8255 chip for I/O. Each I/O board has three ports where each port contains three parallel communication channels. Currently, the I/O boards operate in mode 1, which simply means channel A of the port is used for local to remote node communication, channel B of the port is used for remote to local node communication, and the eight lines of channel C are used for hand-shaking.

Within the framework of the C programming language we set a pointer equal to the address of the appropriate channel of the appropriate port of the I/O board. Then by setting the value of whatever the pointer points to we can send a byte of data over the parallel communication link. Likewise, by reading the value of what the pointer points to, we can read what has been sent over the parallel link. Communication is synchronized by the use of a read acknowledge line and a write acknowledge line. If a sender is to send more than one byte of information, it waits for a read acknowledge signal from the receiving side before sending each subsequent byte. The read acknowledge is set by the receiver when its CPU board reads what has been sent to its I/O board. Similarly, the receiver CPU will not read what its I/O board has until it receives a write acknowledge. A write acknowledge is set whenever the sender places a byte of information on its parallel port.

Data sent from the local node to the remote node will always consist of a mode word, local status word, and a remote command word, each being two bytes in length. The bits of the mode word indicate what operational mode the system is currently in, whether indexing is on or not, and whether any parameters have been changed. Similarly the bits of the local status word show the local status, while the bits of the remote command word indicate the functions the remote side is to perform. See Figure 5 for a definition of the bits. Other data to be sent will consist of one or more of the following items FRHC T6 matrix, FRHC Cartesian velocity, FRHC joint angles, robot end effector length or FRHC frame vs. robot frame difference. The last two items will be sent only once after the value of either has been changed by the operator. By deciphering the mode word the local communications CPU determines which data to retrieve from shared RAM and send over the parallel communications link. For example if the mode is joint mode and the parameter change bit is set the local communications CPU will retrieve the mode word, local status word, remote command word, FRHC joint angles, robot end effector length,

- Mode Control Word

Bit	Control Mode
0	Position control
1	Rate control
2	Current control (future use)
3	Joint control (future use)
4	Index mode - On
5	Rotation indexing - On
6 to 14	Unused
15	Parameter change indicator

- FRHC Command Word

Bit	Command
0	Calibration cm'd
1	Servo power on/off cm'd

1c - Robot Command Word

Bit	Command
0	Calibration cm'd
1	Servo power on/off cm'd
2	Unused
3	Gripper open/close cm'd

- Robot Status Word

Bit	Meaning
0	Calibration status
1	Servo power on/off status
2	Joint limit violation in inverse kinematics solution
3	Gripper open or closed status
4	Robot processor (MACS) error
5	Robot UC error
6	Comm processor error
7	Sensor processor error

- FRHC Status Word

Bit	Meaning
0	Calibration status
1	Servo power on/off status
2	Unused for now
3	Unused for now
4	HACS error
5	MIS error
6	Comm processor error
7	Graphics processor error

Figure 5. Mode Status Command Word Bit Definition

and the FRHC vs. robot frame difference. These data items are assembled into a data buffer and sent byte by byte to the remote communications CPU. The first byte sent is a number telling how many bytes of information are contained in the data buffer. Held in the first two bytes of the data buffer is the mode word. Upon deciphering the mode word the receiving node then knows what data is held in the buffer. This data is then deposited in the appropriate places in shared RAM.

The format of the data held in the buffer is the actual binary pattern residing at the memory address that corresponds to the variable you have selected. For unsigned integer variables such as the mode and status word, it is the expected binary representation. However, for non-integer numbers such as the robot end effector or the T6 matrices, the memory location corresponding to a variable represents the non-integer variable as a 64 bit double precision number in the National Semiconductor Series 32000 floating point format. The 64 bit field contains such information as the sign of a number, the value of its exponent and the value of its mantissa. While non-integer information is originally held in a 64 bit field (double precision number) it is first converted to a 32 bit field (single precision number) before placing it in the data buffer. Again, the 32 bit field of the single precision number still contains such information as sign, mantissa, and exponent. The double precision to single precision conversion speeds overall parallel communication at the price of a slight reduction in the accuracy of the numbers transmitted.

In sending the literal contents of a variable's address in memory in floating point format we greatly increase the overall communication throughput, as compared to the option of sending numbers over in ASCII format.

Embedded within the communication software is also a provision for checking the health of each processor. Associated with each processor is a 32 bit long error word. Each bit in the error word corresponds to a specific problem in a specific processor. When a processor detects some problem within itself it then sets the appropriate bit with its error word. Of course these errors must be of a non-catastrophic nature, because if it were to cause a processor to "die" then that processor would be unable to set a bit in its error word. If the remote communications processor finds any of the remote node error words non-zero (i.e., some type of error) it sets the appropriate bit in the remote status word and ships any non-zero error words to the local node. By looking at the bits of the error word the operator can then tell what error has occurred in what processor and take any appropriate action.

6.0 SETPOINT INTERPOLATION

When both the local and remote nodes first became operational it was found that the robot motion, when following position commands generated by the FRHC, was unsatisfactorily jumpy. Upon comparing the cycle rates of the local and remote nodes it was found that the FRHC control CPU was running faster than the PUMA robot control CPU. Since the two processors were running asynchronously this meant that occasionally the robot control processor would miss a position command from the FRHC control CPU.

In an attemptooth out the position commands the robot tries to servo to, a spline fit was made seven if occasionally a point was missed the motion would be smooth and continuouead of smoothing out the FRHC Cartesian commands it was decided to smooth outesults of the robot's inverse kinematics (i.e., a joint space position composed the s six joint values) from an FRHC Cartesian command.

It was decto use a cubic parametric that is both smooth and continuous with the second deri of the curve at the spline knots (the original points we wish splined together) beingrarily set to zero. Referring to Figure 6 where $\underline{p1}$, $\underline{p2}$, and $\underline{p3}$ are the three joint spasition vectors we want splined together and $\underline{T1}$, $\underline{T2}$, and $\underline{T3}$ are the corresponding nt vectors, then a space curve through $\underline{p2}$ and $\underline{p3}$ is expressed by the equation

$$\underline{p}(u) = A + Bu + Cu^2 + Du^3 \tag{14}$$

where currently $u \leq 1$. From the boundary conditions we get the following relationships:

$$\underline{p}(0) = \underline{p2} \tag{15}$$

$$\underline{p}(1) = \underline{p3} \tag{16}$$

$$\left.\frac{dp}{du}\right|_{u=0} = \underline{T2} \tag{17}$$

$$\left.\frac{dp}{du}\right|_{u=1} = \underline{T3} \tag{18}$$

$$\underline{p}''(0) = \underline{p}''(1) = 0 \tag{19}$$

Using these rationships, we can easily determine that from equation (14)

$$A = \underline{p2} \tag{20}$$

$$B = \underline{T2} \tag{21}$$

$$C = 3 * (\underline{p3} - \underline{p2}) - 2 * \underline{T2} - \underline{T3} \tag{22}$$

$$D = 2 * (\underline{p2} - \underline{p3}) + \underline{T2} + \underline{T3} \tag{23}$$

Further we can determine that

$$T2[i] = \frac{(-6 * (p2[i] - p1[i] + 12 * (p3[i] - p1[i]) - 6 * (p3[i] - p2[i]))}{12} \tag{24}$$

$$T3[i] = \frac{(3 * (p2[i] - p1[i] - 6 * (p3[i] - p1[i]) + 21 * (p3[i] - p2[i]))}{12} \tag{25}$$

where $i = 0, 1, 2, 3, 4, 5$ are the six components of a robot joint space position, $\underline{p1}$ is the third most recent result of the robot inverse kinematics, $\underline{p2}$ is the second most recent result of the robot inverse kinematics and $\underline{p3}$ is the most recent result of the robot inverse kinematics.

Now, using equation (14) we generate seven spline fits that are equally spaced between $u = 0.5$ and $u = 1.5$. So, the spline fit starts midway between $\underline{p3}$ and $\underline{p2}$. It then follows a smooth curve through $\underline{p3}$ and ends at a predicted joint space point halfway past $\underline{p3}$.

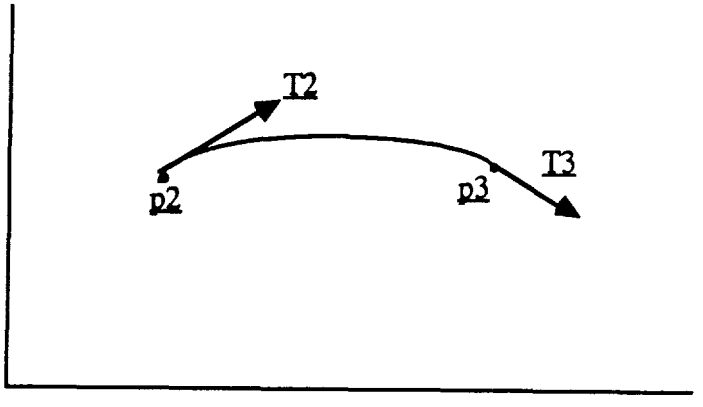


Figure 6. Two dimensional representation of six dimensional space curve

7.0 SYSTEM THROUGHPUT

There are primarily eight tasks that the PUMA robot control CPU performs in a serial fashion.

1) Retrieval of FRHC T6 from shared RAM: This process takes 0.000143 seconds which is equivalent to a rate of 6976.7 Hertz.

2) Workspace adjustment for use in forward kinematics: This process takes 0.001533 seconds which is equivalent to a rate of 6522 Hertz.

3) Forward kinematics: This process takes 0.006 seconds which is equivalent to a rate of 166.6 Hertz.

4) End effector compensation for forward kinematics: This process takes 0.00020 seconds which is equivalent to 5000 Hertz.

5) Placement of PUMA T6 in shared RAM: This process takes 0.0001366 seconds which is equivalent to a rate of 7317 Hertz.

6) Wordjustment for use in inverse kinematics: This process takes 0.00102 seconds equivalent to a rate of 980 Hertz.

7) End compensation for inverse kinematics: This process takes 0.0001925 seconds equivalent to a rate of 5194 Hertz.

8) Invenatics: This process takes 0.01071 seconds which is equivalent to a rate of 93.

Sum these times and including a few other smaller processes that are also part of the rolrol CPU computational workload yields a cycle time of 0.02267 seconds which is at to a rate of 44 Hertz.

Comion time to send a FRHC T6 matrix and receive a PUMA T6 matrix is 0.014 secich is equivalent to a rate of 71.4 Hertz.

The :ontrol CPU cycles at a rate of 25 Hertz in position mode and 27 Hertz in rate mle times of 0.04 seconds and 0.037 seconds respectively). This CPU had a 6 megaick rate.

Sum of these times in position mode, the minimum amount of time it takes to send and from the FRHC, have the command transmitted to the robot, send the robot back to the local node and generate position error based force feedback in the FR.0767 seconds, which is equivalent to a rate of 13 Hertz.

All oftware was coded in the high level programming language C, however, a rather inecross compiler (National Semiconductor's GCS) was used.

8.0 CONCNS AND FUTURE WORK

This resents the current framework of our teleoperator system development. Position of a PUMA robot bypassing VAL and using a distributed computing system aptisfactory, albeit with some limitations:

1) SinceMA joint processors' PID control parameters are proprietary information and oe easily accessed and altered, the control flexibility is hence somewhat handi

2) The Ioint rate information, which is crucial for bilateral servo control, is also not a

3) The hand controller electronics do not provide velocity information, and the velociation by software is not very accurate.

A new rsal Controller (UC) is under development to replace both of the hand controller ics and the entire PUMA controller in the near future [1]. This UC shall provide eass to the control parameters and easy adaptation of different control

methods. The limitations cited above that appeared in the current teleoperator setup shall be alleviated when the UC is implemented.

The goal of our teleoperator development is to realize a higher throughput bilateral servo control system. The following work is planned to reach this goal:

1) Direct current control, instead of position control, shall be implemented for force or force/position control of the robot arm.

2) Robot dynamics, in addition to kinematics, shall be implemented.

3) Information from robot force torque sensors shall be included in the calculation of hand controller force feedback.

4) To attempt an increase in system throughput, interrupt driving and synchronization of the distributed processor shall be explored.

5) Obtain a more efficient cross compiler.

9.0 REFERENCES

[1] Bejczy, A.K. and Z. Szakaly, "Universal Computer Control System for Space Telerobots," Proceedings of IEEE Conference on Robotics and Automation, Raleigh, NC, March 1987.

[2] Paul, R.P., Robot Manipulators - Mathematics Programming and Control, MIT Press, 1981.

[3] Denavit, J. and R.S. Hartenberg, "A Kinematic Notation for Lower Pair Mechanisms Based on Matrices," Vol. 22, Transactions of the ASME, 19855, pp. 215-221.

[4] Fong, C.P., A.K. Bejczy, and R. Dotson, "Distributed Microcomputer Control System for Advanced Teleoperators," Proceedings of IEEE International Conference on Robotics and Automation, San Francisco, CA, April 7-10, 1986.

[5] Salganicoff, M., E. Austin, and C.P. Fong, "Real-time Simulation of a Distributed Teleoperator System," Proceedings of IASTED (International Association for Science and Technology for Developement) Conference on Applied Simulation and Modeling, Vancouver, BC, June 1986, pp. 594-598.

[6] Pressman, R.S. and J.E. Williams, Numerical Control and Computer Aided Manufacturing, John Wiley and Sons, Inc., 1977.

PERFORMANCE IMPROVEMENT OF ROBOTS USING A LEARNING CONTROL SCHEME

Ramuhalli Krishna
Advanced Technology & Research, Inc.,
3933 Sandy Spring Road
Burtonsville, MD 20866

and

Pen-Tai Chiang and Jackson C.S. Yang
Robotics Laboratory
Department of Mechanical Engineering
University of Maryland
College Park, MD 20742

ABSTRACT

Many applications of robots require that the same task be repeated a number of times. In such applications, the errors associated with one cycle are also repeated every cycle of the operation. An off-line learning control scheme is used here to modify the command function which would result in smaller errors in the next operation. The learning scheme is based on a knowledge of the errors and error rates associated with each cycle. Necessary conditions for the iterative scheme to converge to zero errors are derived analytically considering a second order servosystem model. Computer simulations show that the errors are reduced at a faster rate if the error rate is included in the iteration scheme. The results also indicate that the scheme may increase the magnitude of errors if the rate information is not included in the iteration scheme. Modification of the command input using a phase and gain adjustment is also proposed to reduce the errors with one attempt. The scheme is then applied to a computer model of a robot system similar to PUMA 560. Improved performance of the robot is shown by considering various cases of trajectory tracing. The scheme is also applied to a real robot PUMA 560. The results show that the proposed scheme can be successfully used to improve the performance of actual robots within the limitations of the repeatability and noise characteristics of the robot.

1987 Goddard Conference on Space Applications of Artificial Intelligence (AI) and Robotics, May 13-14, 1987

1.0 INTRODUCTION

Several methods of performance improvement of robots have been attempted previously by many researchers. Some of these employ on-line adaptive control schemes[1,2] considering factors such as flexibility of the arms and variation of loads. Such control schemes are required for trajectories which are not defined a priori or require accurate trajectory control in the first attempt. For repetitive type of operations, as most commonly required in industrial applications, such as welding, cutting or sealing, a control scheme which can learn based on its previous performance appears to be more attractive because of the simplicity of the technique. A method to obtain a modified command input signal called Computed Repetitive Adjustment Technique (CREATE)[3] is embodied in an algorithm for correcting a robot's motion in successive testing of the same job. This repetitive testing is continued until the trajectory errors are within the acceptable bound before performing the actual work. The technique was later applied[4,5] to improve the performance of a mathematical model of a three-link robot arm.

A scheme very similar to CREATE was studied by Craig[6] for application by considering a linearized model of the robot. Conditions for the convergence of the scheme to yield minimum errors were obtained with the assumption that several critical parameters of the system are known. Arimoto[7] proposed a learning scheme based on measuring error rate only. He later extended[8,9] the control scheme to include the error and error rate information. However, the conditions[8] he arrived at are found to be unsatisfactory. Several other researchers[10,11,12] have attempted to obtain convergence conditions for the iteration scheme, but the analysis are generally inadequate. Togai[13,14] obtained interesting algorithms by using discrete analysis and by applying optimal control techniques. An algorithm based on optimal control technique is also given by Harokopos[15] for continuous systems. Bedewi[16] used CREATE technique to refine the performance of the robot starting with a dynamic inverse of the model of the system. In this paper, the learning control scheme proposed by Arimoto[8] will be considered. Analysis of the scheme will be attempted with a servosystem model. The scheme will then be applied to a mathematical model of a three-link robot to show the improvement in performance.

2.0 SECOND ORDER SERVOSYSTEM MODEL

2.1 Description of the Servosystem Model

A servo control design is given in this section which will be applied to design independent control of each joint of the robot. The second order servosystem model is also used in this section to test the proposed learning control scheme. The scheme is then applied to improve the performance of the robot model in section 3.

The dynamics of the servosystem can be represented, by ignoring the damping factor for the sake of simplicity, as

$$I\ddot{\theta} = T_C \qquad (1)$$

where θ is the angular position (a function of time, t)

T_C is the control torque

I is the moment of inertia of the servo, and

$\ddot{\theta}$ is the second derivative of θ with respect to time

It is necessary to feedback the angular position, θ , and its rate, $\dot{\theta}$, to obtain a stable position control system. An integral feedback of the position may also be used to obtain a higher speed of response. However, an integral feedback increases the order of the system and introduces oscillations in the system response. These characteristics are not desirable for the present application and hence only a proportional plus derivative controller is considered for stabilizing the servosystem as shown in Fig. 1. The characteristic equation of the closedloop system can be written as,

$$s^2 + (K_2/I)s + (K_1/I) = 0 \qquad (2)$$

This equation may be compared with the damped oscillatory system

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0 \qquad (3)$$

where ζ is the damping ratio and ω_n is the natural frequency of the system. It is desirable to have a high value of ω_n to get a large bandwidth of the system. The stability of the second order system is guaranteed as long as K_1 and K_2 remain positive. However, the feedback gain value K_1 can not be increased beyond a certain a limit (due to the practical limitations of the actuator) in an effort to increase the bandwidth of the system. Within these limits, the feedback gain values are obtained using the relations (obtained from Eqs. (2) and (3)) as,

$$K_1 = \omega_n^2 I \quad \text{and} \quad K_2 = 2\zeta\omega_n I \qquad (4)$$

The rate feedback gain, K_2, is assumed to achieve critical damping in the closedloop system. For such a case, the system will have the highest speed of response corresponding to the lowest settling time and for a given bandwidth of the system. Fig. 2 shows the transient response of the second order system for a step input and for the selected feedback gain values.

Case	(a)	$K_1 = 100$	$K_2 = 20$	$\omega_n = 10$ rad/sec
	(b)	$K_1 = 144$	$K_2 = 24$	$\omega_n = 12$ rad/sec
	(c)	$K_1 = 625$	$K_2 = 50$	$\omega_n = 25$ rad/sec

Case (c) shows the lowest rise time of approximately 0.2 sec. The same feedback gain values will be considered later to control the robot arm joints.

2.2 A Learning Control Scheme

The second order system considered may be represented as (from Fig. 2),

$$(Is^2 + K_2 S + K_1)\,\theta = \theta_C \qquad (5)$$

l, the command function, θ_C, is the same as the desired trajectory, θ_C, is

$$\dot{\theta}_d - \dot{\theta}_a) + b(\theta_d - \theta_a) \qquad (6)$$

re constants proportional to the error rate and the error, respectively. For
1 the command signal, θ_C , is modified as,

$$)_C + \theta_e \qquad (7)$$

the system to this modified command signal is determined. The new error
)btained using Eq. (6) The command signal is again modified according to
w command signal is used in Eq. (2.5) and the response of the system is
: process is repeated until the errors, θ_e, fall within an acceptable value.

ıdy is possible for the second order system considered here. Eq. (5)
second order system is written in a slightly different form for the kth

$$\theta_k = \theta_{ck} \qquad (8)$$

:eration the equation as

$$| + K_1\theta_{k+1} = \theta_{ck} + \theta_{ek} \qquad (9)$$

$$_k) + b(\theta_d - \theta_k) \qquad (10)$$

(8) from Eq. (9), we get

$$= \theta_{ek} \qquad (11)$$

$$)_k$$

$$a\dot{d}_k + bd_k \qquad (12)$$

$$(\tau)\, d\tau \qquad (13)$$

Then

$$J_{k+1} = \int_0^t \theta^2_{ek+1}(\tau)\, d\tau$$

$$= \int_0^t \left[\theta_{ek} - (a\dot{d}_k + bd_k)\right]^2 d\tau$$

Expanding and using equations (13) and (10), we get

$$J_k - J_{k+1} = \int_0^t \{(2aq - a^2 - 2b)\,\dot{d}_k^2 + (2bp - b^2)\,d_k^2\}\, d\tau + a\dot{d}_k^2(t)$$

$$+ (ap + bq - ab)\,d_k^2 + 2bd_k\dot{d}_k$$

$$\geq 0, \quad \text{for } a^2 > 2b$$

$$a < 3q/2$$

$$b < 2p \quad \text{and for small error rates} \qquad (14)$$

Eq.(14) gives sufficient conditions for the guaranteed convergence of the error to zero value with each step of the proposed iteration scheme.

Since only the command signals are being modified, the torque requirements on the actuators will not drastically change provided the initial errors are small. The feedback values selected for the application in this section are $K_1 = 100$ and $K_2 = 20$ and corresponds to a closedloop frequency of 10 rad/sec. The response of the servosystem to a command input,

$$\theta_C = \sin(\omega t)$$

is shown in Fig. 3 for $\omega = \pi$ rad/sec. The actual trajectory shows a phase lag and an amplitude modification. Application of the learning control scheme shows reduced errors with the first iteration (Fig. 4) and the third iteration (Fig. 5) and for the parameters a= 0.1 and b= 0.9. The figures compare average errors associated with each iteration. The average

as

$$AE = \frac{\sum_{i=1}^{NP} | \theta_{ei} |}{NP} \qquad (15)$$

mber of points on the trajectory. Higher values of ω were considered next
ance of the iteration scheme was studied. Fig. 6 shows the average error
each iteration for various values of "a" and "b" and for ω = 2 π rad/sec.
rly show that for the case in which a = 0, the errors tend to increase after a
n accordance with the derived conditions (Eq. (14)). Similar results are
for ω =4 π rad/sec. In general, the results show that the rate of
reases with increasing values of "a" up to a certain value. For the present
best results are obtained with a= 0.2 and b= 0.8 at all values of ω. The
ing error rate in the modification procedure is obvious from Figs. 6 and 7.
bserved that the conditions given by Eq. (14) are rather conservative

1 Phase Adjustment Technique

he previous results that there is a unique command function, θ_C , for
l trajectory, θ_a , will be the same as the desired trajectory, at least for
l single-output systems. In the previous examples, inclusion of the error
helped a great deal in arriving at the unique command function with only a
The next obvious question is whether it is possible to obtain the ideal
on with a single iteration so that the resulting trajectory would follow the
y. An attempt has been made in this section to answer the question.

of the second order system shows that the output of the system follows the
ly, provided the frequencies associated with the function are well within the
th. As the input frequency approaches the system bandwidth, the output of
s marked deviation from the input signal. The input-output relationship is
se and the gain plots in which the phase difference and the ratio of the
: output to the input are plotted as a function of the input frequency.

he previous example, θ_d = sin(ωt), with ω = π rad/sec. The time
system is as shown in Fig. 3., for the first trial. A modification scheme
was employed with a=0.1 and b=0.9. The actual trajectories obtained
n and third iteration are shown in Figs. 4 and 5, respectively. From Fig. 3
ues of phase (-0.5969 rad) and gain (0.91) were obtained. The command
dified as

l sin(ωt + 0.5969)

tory obtained for this modified command signal is shown in Fig. 8. The
and the desired trajectory overlap within the accuracy of the plotter. Thus,
linear system, it was possible to obtain a modified command signal to
ed trajectory without an iteration scheme. This method will be applied to
del of three-link robot system in section 3, which is an example of a
ic system.

3.0 APPLICATION TO A THREE-LINK ROBOT

3.1 Dynamics of a Three-Link Robot Arm

Fig. 9 shows the selected coordinate reference frame for the three-link robot arm. X, Y, Z, is the inertial system with the origin at joint 1. The fourth coordinate frame has the origin fixed at the tip of the robot arm. The first link can rotate only about the vertical axis Z, carrying the second and the third links. The second link can rotate about an axis fixed in link 2 and is normal to both the first and the second link. The third link moves about an axis parallel to the axis of rotation of the second link.

The three transformations are represented by the following matrices

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} C_1 & -S_1 & 0 & 0 \\ S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix}$$

$$\text{or} \quad r = [A_1]\, r_1 \tag{16}$$

$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_2 & -S_2 & C_2 d_2 \\ 0 & S_2 & C_2 & S_2 d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{bmatrix}$$

$$\text{or} \quad r_1 = [A_2]\, r_2 \tag{17}$$

$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_3 & -S_3 & C_3 d_3 \\ 0 & S_3 & C_3 & S_3 d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_3 \\ Y_3 \\ Z_3 \\ 1 \end{bmatrix}$$

$$\text{or} \quad r_2 = [A_3]\, r_3 \tag{18}$$

where $C_j = \cos\theta_j$, and $S_j = \sin\theta_j$, $j=1,2,3$
A point r_i described with respect to link i can be related to the base coordinate by

$$r = [T_i]\, r_i \tag{19}$$

where, $T_1 = [A_1]$, $T_2 = [A_1][A_2]$, and $T_3 = [T_2][A_3]$

The dynamic equation for the robot arm is obtained as, [17]

$$F_i = \sum_{j=1}^{3} D_{ij}\ddot{q}_j + Ia_i\ddot{q}_i + \sum_{j=1}^{3}\sum_{k=1}^{3} D_{ijk}\dot{q}_j\dot{q}_k + D_i \qquad (20)$$

where:

$$D_{ij} = \sum_{p=\max i,j}^{3} \text{Trace}\left(\frac{\partial T_p}{\partial q_j} J_p \frac{\partial T_p^T}{\partial q_i}\right)$$

$$D_{ijk} = \sum_{p=\max i,j}^{3} \text{Trace}\left(\frac{\partial^2 T_p}{\partial q_j \partial q_k} J_p \frac{\partial T_p^T}{\partial q_i}\right)$$

$$D_i = \sum_{i=1}^{3} -m_p\, g^T \frac{\partial T_p^T}{\partial q_i} r_p$$

I_{ai} is the I^{th} actuator inertia, m_p is the mass of the p^{th} link
g is a vector in the direction of the gravity force
J_p is a pseudo inertia matrix for the pth link

Eq. (20) are used to compute the joint torques required. For the purpose of simulation of the first, second and the third links are assumed to have lengths equal to 0.5m, 0.4m, and masses 4kg, and 2 kg, and 1kg, respectively. The inertias of the links 1,2, and, 3 are assumed to be 5×10^{-4}, and 2.5×10^{-4}, and 1.25×10^{-4} kgm^2, respectively. Actuator inertias corresponding to joints 1,2, and, 3 are taken to be 1×10^{-3}, 5×10^{-3}, and 2.5×10^{-4}, kgm^2 respectively.

3.2 Inverse Kinematics

The trajectory to be traced by the robot arm is given by a set of points which are usually defined in the base (inertial) coordinate system. Therefore, it is necessary to find the joint angles corresponding to each point on the trajectory using the geometry of the robot. Let us assume that the tip of the robot arm follows the given trajectory, so that a transformation of the tip point into the base coordinates can be made as follows:

$$[P] = [T_3]\ [r_3] \qquad (21)$$

where r_3 is actually the origin of the fourth coordinate system. Substituting for T_3 and expanding we get,

$$P_X = -S_1(C_2D_2 + C_{23}D_3) \qquad (22)$$
$$P_Y = C_1(C_2D_2 + C_{23}D_3) \qquad (23)$$
$$P_Z = D_1 + D_2S_2 + D_3C_{23} \qquad (24)$$

Algebraic manipulations of the above equations give the expressions for the joint angles as

$$\theta_1 = -\tan^{-1}(x_1/y_1) \qquad (25)$$

$$\frac{- d_1}{d}) - \cos^{-1}\left(\frac{d^2 + d_2^2 - d_3^2}{2dd_2}\right) \quad (26)$$

$$\frac{{}_1/c_1)^2 + (z_1 - d_1)^2 - d_2^2 - d_3^2}{2d_2d_3}\} \quad (27)$$

ɔmpute the joint angles only once in order to find the command signals
ry. This can be done off-line using the robot central computer in the
iterative technique is then applied to modify the joint angle commands to
gle errors to zero.

rol system

: employed here is to control the motion of the joints so that the tip of the
rajectory. Independent control of each of the joints based on only the
position and rate feedbacks is considered here for the sake of simplicity.
ıed in the base coordinate system is converted into joint command angles
kinematics. The dynamic equations (20) are rearranged for the
ol techniques as follows.

$$\sum_{k=1}^{3} D_{ijk}\dot{\theta}_j\dot{\theta}_k - \sum_{\substack{j=1 \\ j\neq 1}}^{3} D_{ij}\ddot{\theta}_j - D_i) / (D_{ii} + Ia_i) \quad (28)$$

int angular coordinates, T_{ci} is the torque about the joint i. Control
ed from the block diagram shown in Fig.1 as

$$(_1(\theta_i - \theta_c) \quad (29)$$

a sixth order nonlinear differential equation with coupled coefficients.
numerically solved here using a fourth order Runge-Kutta integration

of Learning Control Scheme

the case in which the desired trajectory for the tip of the arm is a circle.
ined in the inertial coordinate system as follows.

s(ωt)

(30)

).5 sin (ωt)

ım is modified so that the joint command angles are obtained from the
/stem through a kinematic transformation. The computer time required
ıpproximately 20 minutes and hence only four iterations are considered
The results obtained are shown in Fig. 11. There are two curves in
ling to the errors associated with X and Z coordinates. The errors in

the Y direction is not so important and is not shown here. It may be seen that the errors tend to become smaller at a faster rate when the error rate information is included in the command input modification procedure. Fig. 12 shows the torque characteristics of the joints 1 and 2 before and after the application of the learning control scheme. Figs. 13 to 15 show the actual response of the robot corresponding to a=0 and b=1. A trajectory very close to the desired circular trajectory could be obtained with a=0.1 and b=1 after 6 iterations (figure not shown).

3.5 Application of Gain and Phase Adjustment Technique

It was shown in section 2.3 that a modified command input could be obtained so that the resulting actual trajectory is very close to the desired trajectory for linear dynamic systems. However, the robots are in general are characterized by nonlinearities and varying moments of inertia parameters. Coupling between the joint coordinates are also very significant. Two examples of trajectory tracing, one a circle and the other a straight line, is considered for the application of the gain and phase modification technique.

The trajectory defined by Eq.(30) describes a circle in the X-Z plane. The actual trajectory traced by the robot is given in Fig. 13. The responses of the robot after first and fourth iterations using the learning scheme are as shown in Figs. 14 and 15, respectively. It may be seen that even after four iterations the trajectory traced is still not a complete circle. The gain and phase modification technique was then attempted as follows.

The response of the system to the command trajectory which is the same as the desired trajectory was plotted in the X and Z coordinate system as a function of time (not shown). Ideally, X coordinate should be a cosine function and Z a sine function as given by Eq. (30). However, depending on the value of w, there will be a phase and gain change. The actual response was found to fit the following functions.

$$X = \cos(\omega t - 0.17) \qquad (31)$$
$$Z = \sin(\omega t - 0.19)$$

Modified command signals were then obtained as

$$X = \cos(\omega t + 0.17) \qquad (32)$$
$$Z = \sin(\omega t + 0.19)$$

The response of the robot to the modified command signal is as shown in Fig. 16. The actual trajectory obtained is seen to be very close to the desired trajectory. It is very interesting to see that the nonlinearity, the coupling between the joint coordinates, gravity and the varying moments of inertias did not have much effect as far as the phase shift and gain values are considered. The present example, however, is very simple because the desired trajectory could be represented by simple sine and cosine functions.

The second example considered is the one in which the tip of the robot traces a straight line. The joint angles will have to go through a nonlinear motion so that the tip of the robot could result in a straight line. Hence, this example is more complicated than the first case. The command function in the inertial coordinate system for the example of a straight line is taken to be

$$X = 0.5 - 0.5\ w\ t$$

 (33)

Y

Z

The respo: system with w = 1 is as shown in Fig. 17. For the sake of simplicity, only the Xite is considered for modification. Learning control scheme is then used to obtain arajectory after 6 iterations as shown. For the application of gain and phase mods, the command signal for the X coordinate is obtained as follows.

$$X = 0.5\ wt + X_e(t + t_p)X_g$$

where X_e ior obtained as

$$X_e \quad X_a$$

and t_p is theshift obtained from the first trail. The response obtained with this modified col is shown in Fig. 17. The phase t_p is taken to be 0.18 secs and the gain, X_g, 1. trial and error). The actual trajectory obtained for this command is almost as gohe trajectory obtained through 6 iterations.

The phase, $t_j$gain, X_g, were also obtained by a Fourier analysis of the error function associated w: trajectory. Fig. 18 shows the response of the corresponding modified command (w = 0.2 secs and X_g = 1.117). The response appears to be at least as good as the nse shown in Fig. 17. Hence, it may be concluded that the Fourier analysis can td to compute the phase and gain value to obtain a modified command and to reduceerrors considerably in the second trial. Further minimization of errors may be done ; the iterative technique, if necessary.

3.6 Applicat to a Real Robot

The performarmprovement technique is attempted on a real robot PUMA 560. Fig. 19 shows a schem diagram of the robot. Instructions to the robot arm is given by using the operating systeoftware, VAL-II[18]. Different values of the coefficient of the error are considered (withe parameter 'a'=0) to study the rate of convergence of the scheme to yield minimumrors. Fig. 20 shows the performance of the robot after 11 iterations for a desired trajectorof the robot to move along the Y axis at approximately 1.5ft/sec. The figure shows therrorin the X direction with the first attempt and after 11 iterations with b=0.2. The erro associated with the first trial is 0.4145mm and , after 11 iteractions , 0.07mm.

The same case was then repeated with various values of the error coefficient, b, and the average errors obtained as a function of the trial number are shown in Fig. 21. The results clearly show that the rate at which the average error decreases is proportional to the value of b. However, larger values of b tends to increase the average error after a few iterations. This trend was also observed and discussed in section 2.

The iterative technique was also applied to improve the performance of the robot under a variety of other operating conditions. The results show, in general, that the lower limit of the average error after reached (which is 0.1mm) is restricted by the repeatability and the noise associated with the joint sensors. Further work is continuing at this time to include the rate information in the application of the learning scheme to the real robot.

4.0 CONCLUSIONS

An off-line learning control scheme is analyzed here which can be used to improve the performance of robots. Necessary conditions for the iterative scheme to reduce errors with each iteration are derived considering a second order servo system model. The scheme is also applied to a mathematical model of a three-link robot similar to PUMA 560. In general, the results show that the learning control scheme reduces errors more efficiently if the error rate information is included in the scheme. The results also show that the scheme may increase the magnitude of the errors, if the rate information is not included in the iteration scheme. Preliminary results of the application of the technique to a real robot has shown that the scheme can be successfully used to improve the performance of actual robots within the limitations of the repeatability and noise characteristics of the robots.

ACKNOWLEDGMENTS

This research work was supported by NSF Grant No. ECS 8313834. The authors wish to thank Drs. Roger Chen, Kam Lau, Nabhi Bedewi, and Mr. Eugene Aronne for their helpful discussions during the course of this work.

REFERENCES

1. Dubowsky, S., and Des Forges,D.T., 'The Application of Model-Referenced Adaptive Control to Robotic Manipulators,' Transactions of the ASME, Journal of Dynamic Systems, Measurement and Control, Sep.. 1979, Vol. 101, pp 193-200.

2. Nelson, W.L., Mitra, D., and Boie, R.A., "End Point Sensing and Load Adaptive Control of a Flexible Robot arm," Proceedings of 24th Conference on Decision and Control, Ft. Lauderdale, Florida, December, 1985.

3. "Optimization of Robot Manufacturing Operations," Report to NSF by Advanced Technology and Research, Inc., April, 1983.

4. Bedewi, N.E., "Computed Reference Repetitive Adjustment Technique for a Robot Arm Control in Three Dimensional Space," M.S. Thesis, Department of Mechanical Engineering, University of Maryland, 1984.

5. "Development of Robot Performance Improvement Technique Through Repetitive Testing" Phase II 6 Months Report to NSF by Advanced Technology and Research, Inc., April 1985, Contract No. ECS 8313834.

6. Craig, J.J., "Adaptive Control of Manipulators Through Repeated Trials," American Control Conference, San Diego, June 1984.

7. Kawamura, S., and Miyazaki, F., and Arimoto, S., "Applications of Learning method for Dynamic Control of Robot Manipulators," Proceedings of 24th Conference on Decision and Control, Ft. Lauderdale, Florida, December, 1985.

8. Arimoto, S., Kawamura, S., and Miyazaki, F., "Bettering Operation of Dynamic Systems by Learning: A New Control Theory for Servomechanism or Mechatronic Systems, "Proceedings of 23rd Conference on Decision and Control, Las Vegas,

ecember, 1984, pp. 1064-69.

, "Learning Control Theory for Dynamical Systems," Proceedings of 23rd on Decision and Control, Ft. Lauderdale, Florida, December, 1985, pp

mata, T., and Nakano, M., "Synthesis of Repetitive Control Systems and ion," Proceedings of 24th Conference on Decision and Control, Ft. , Florida, December 1985, pp 1387-92.

d Kato, E., "Iterative Control and Its Application to Motion Control of - A Direct Approach to Servo Problems," Proceedings of 24th Conference and Control, Ft. Lauderdale, Florida, december 1985, pp 1393-98.

nd Yamakita, M., "Iterative Generation of Optimal Input of A r," IEEE Conference on Robotics and Automation, April 1986, pp 579-584.

nd Yamano,O., "Analysis and design of an Optimal Learning Control Industrial Robots," Proceedings of 24th Conference on Decision and Lauderdale, Florida, December 1985, pp 1399-1404.

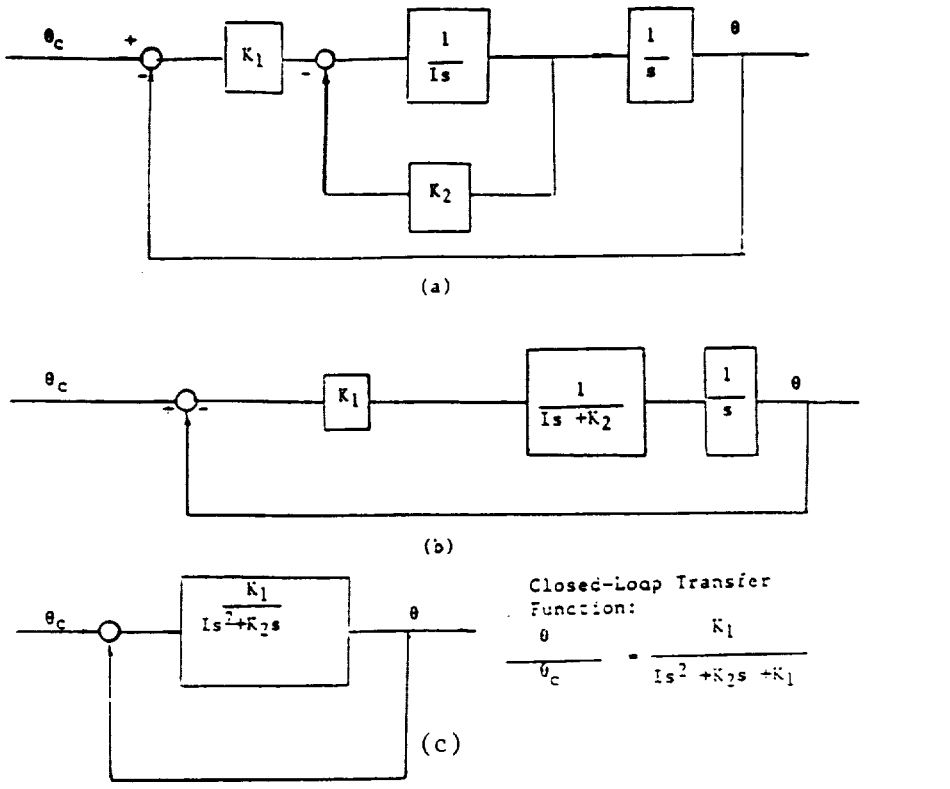
nd Yamano, O., "Learning Control and Its Optimality: Analysis and Its to Controlling Industrial Robots," Proceedings of the IEEE Conference on d Automation, April 1986, pp 248-253.

E.G., "Optimal Learning Control of Mechanical Manipulators in Repetitive roceedings of the IEEE Conference on Robotics and Automation, April 6-401.

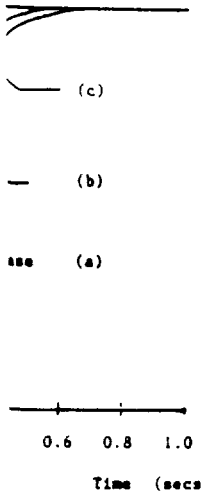
E., Chiang, N.E., and Yang, J.C.S., "Robot Position Accuracy nt Through Repeated Trials," Paper No. 86-DET-44, Design Engineering onference, Columbus, Ohio, October 5-8, 1986.

'Robot Manipulators: Mathematics, Programming and Control," The MIT achusetts, 1981.

User's Guide to VAL" Unimation, Inc., Danbury, Connecticut.



Block-Diagram Representation of the Servo Control System and (c) Block-Diagram Reduction Steps



nput Response for
ent sets of Gain

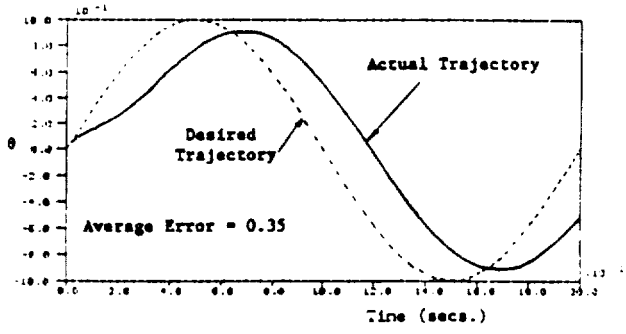


Fig. 3. Transient Response of the Servosystem

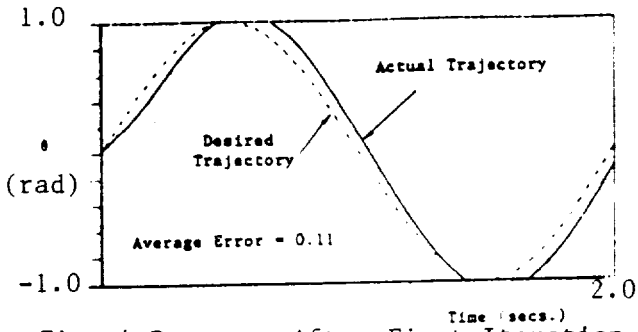


Fig. 4 Response After First Iteration

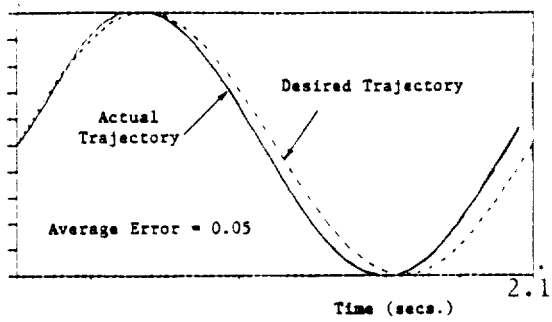
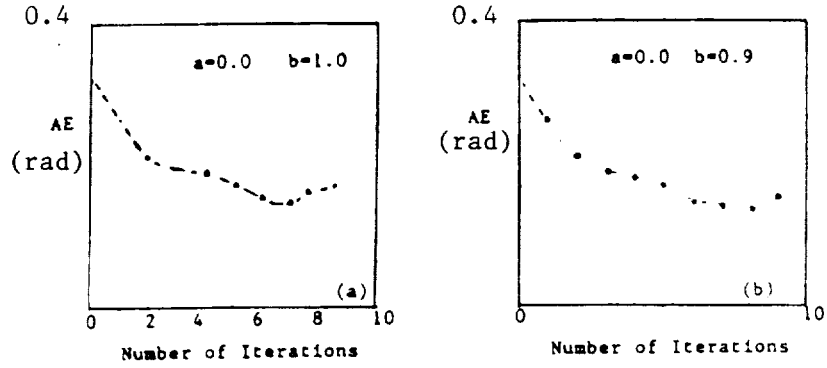
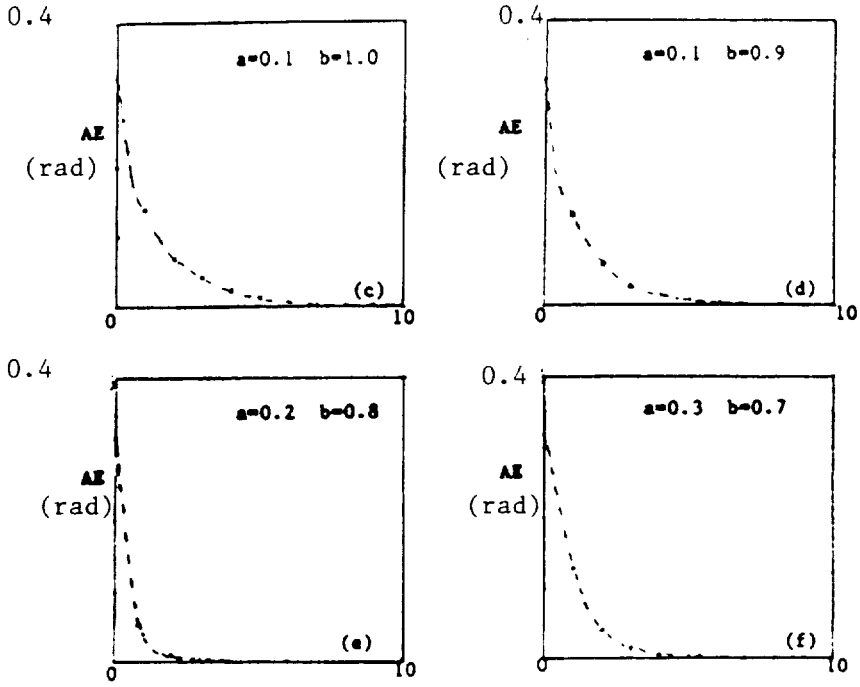


Fig. 5. Response After Third Iteration

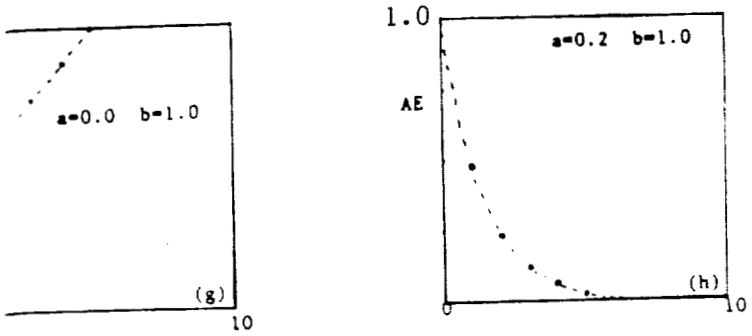




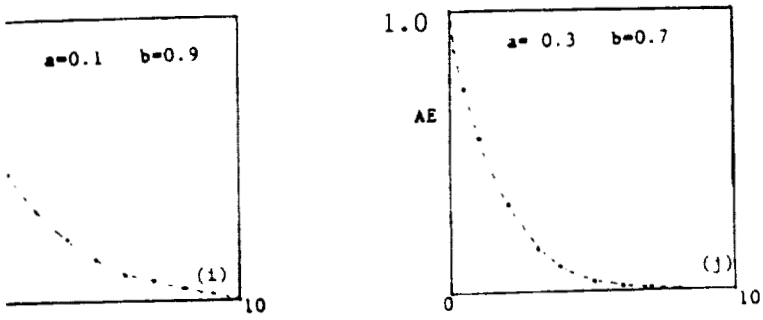
$X_c = \sin \omega t$

$\omega = 2\pi$ rad/sec

Fig. 6. Error Convergence of the Servosystem with the Learning Control Scheme

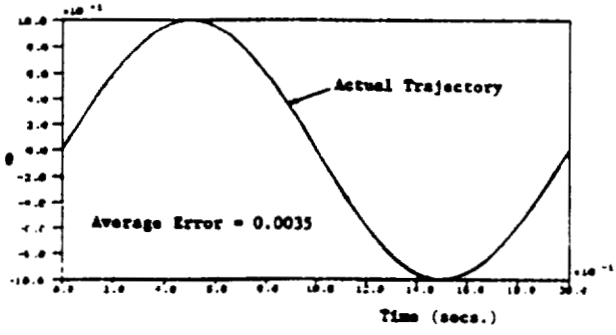


mber of Iterations



c = sin (wt) w = 4 π rad/sec

ror Convergence of the Servosystem with the Learning
ntrol Scheme



ise of the System for the Modified Command Input

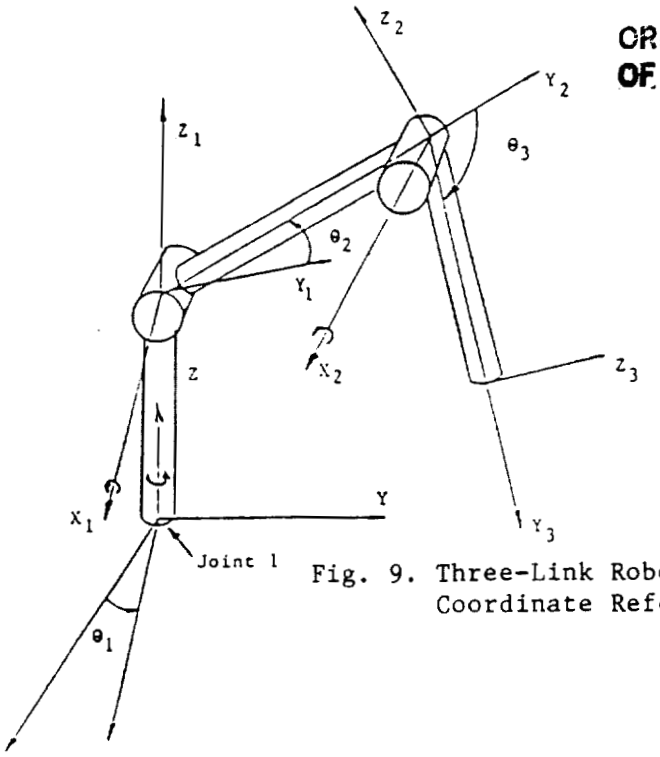
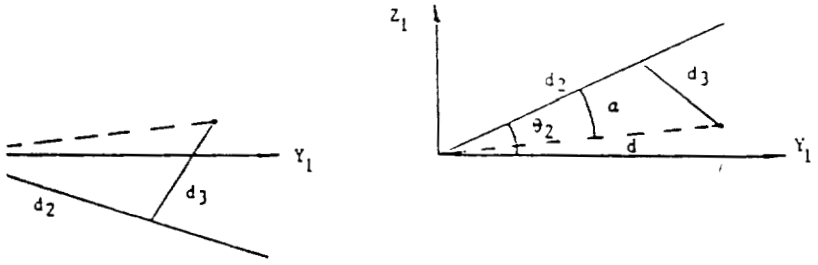
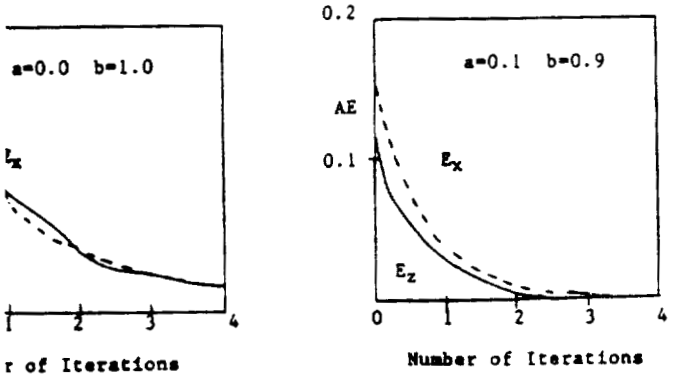


Fig. 9. Three-Link Robot Arm and the Coordinate Reference Frames



ible Configurations of Links 2 and 3 to reach a
int



Convergence for a Three-Link Robot Computer Model

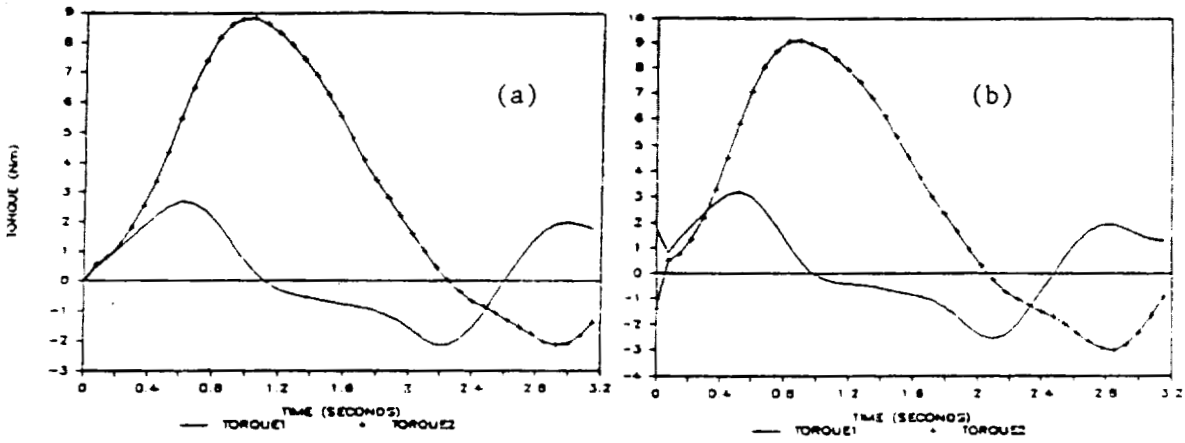


Fig. 12. Torque Characteristics (a) First Trial
(b) After Four Iterations

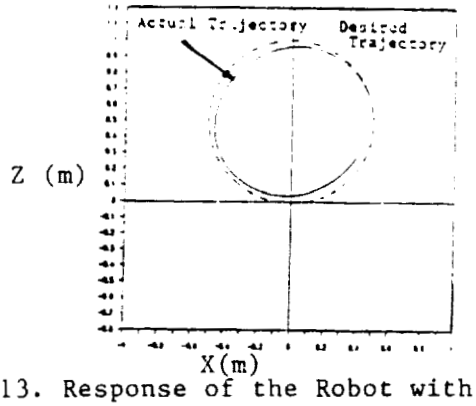


Fig. 13. Response of the Robot with Command Input Same as Desired Trajectory

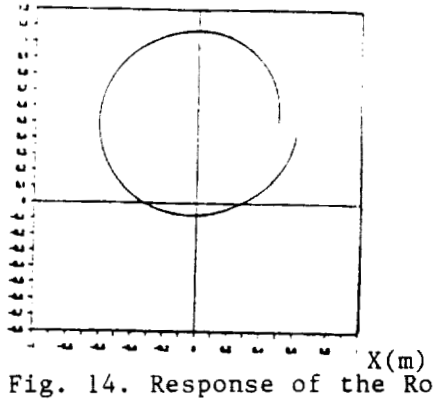


Fig. 14. Response of the Robot After 1 Iteration

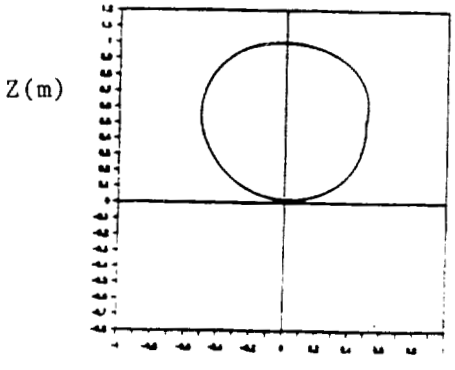


Fig. 15. Response of the Robot After Four Iterations

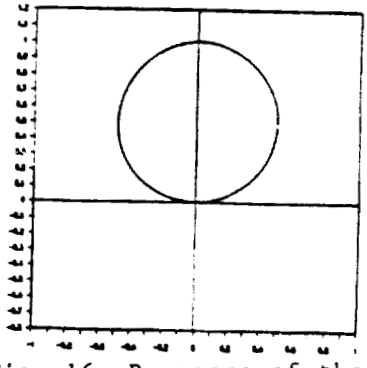


Fig. 16. Response of the Robot for a Modified Command Input

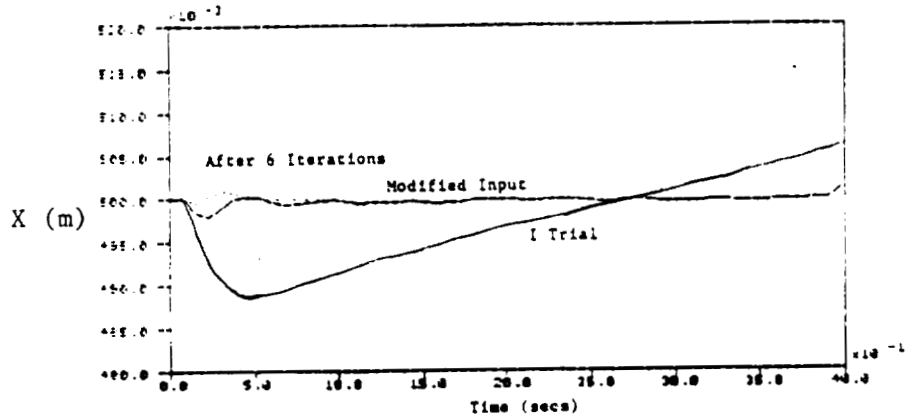


Fig. 17. Response of the Robot After 6 Iterations and A Modified Command Input

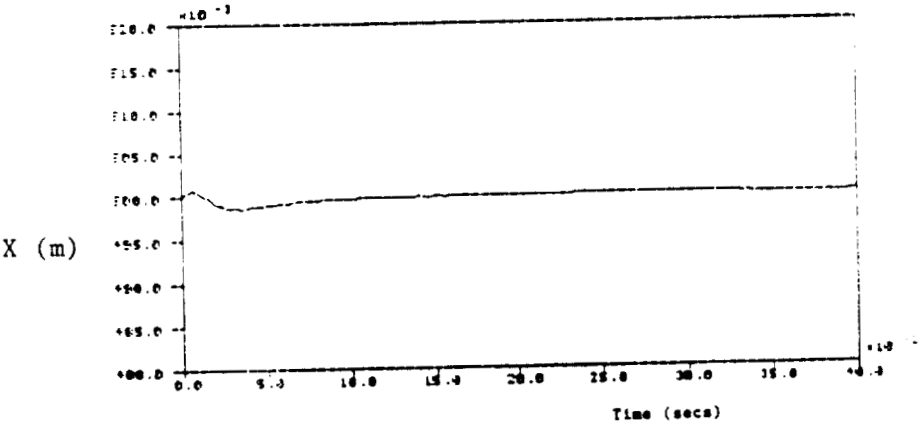


Fig. 18. Response of the Robot for a Modified Command Input (t_p = 0.2 secs.)

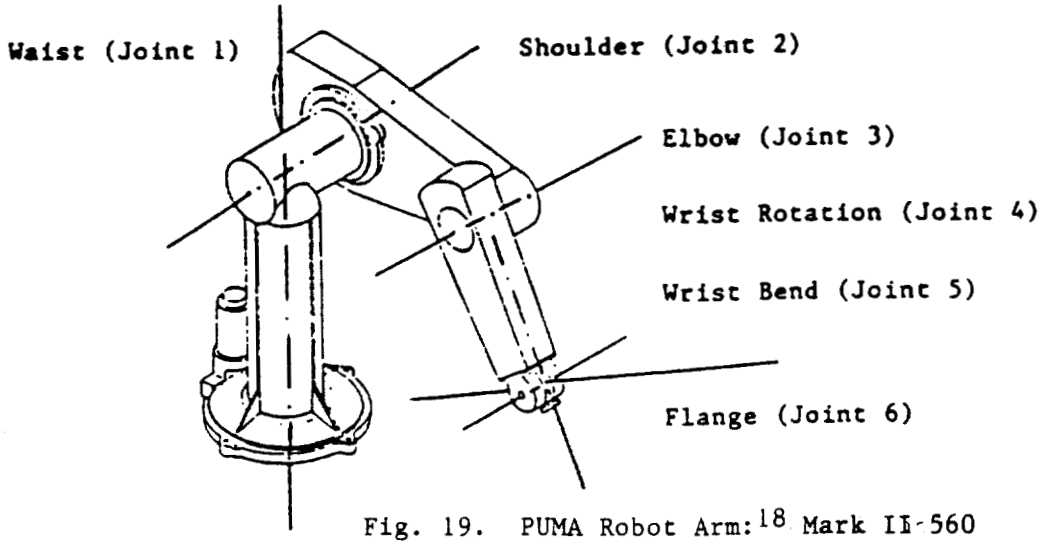


Fig. 19. PUMA Robot Arm:[18] Mark II-560

Fig. 20 STRAIGHT LINE ALONG Y AXIS,
ATTACHED PAYLOAD NONE, SPEED 150

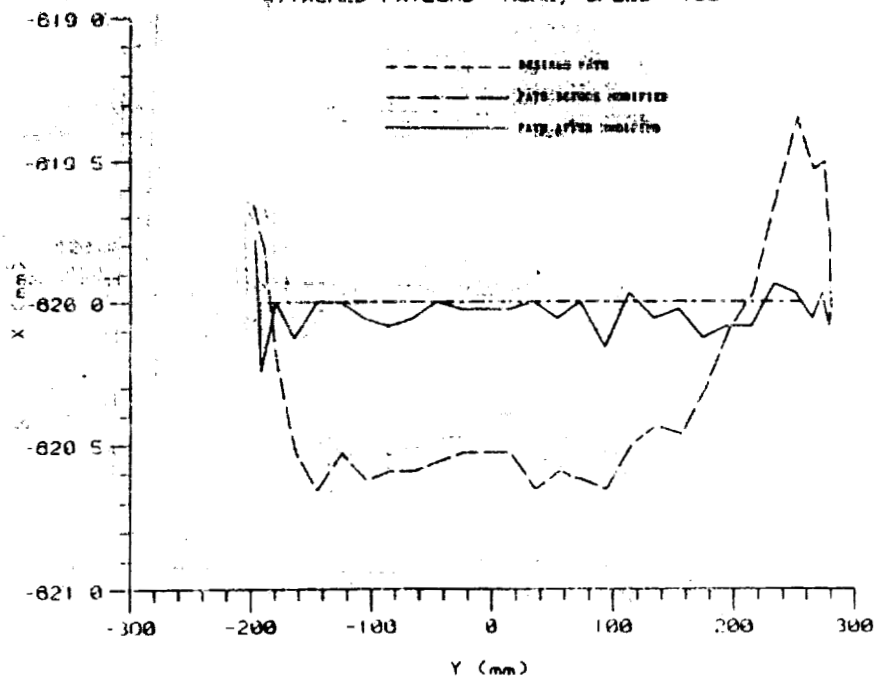
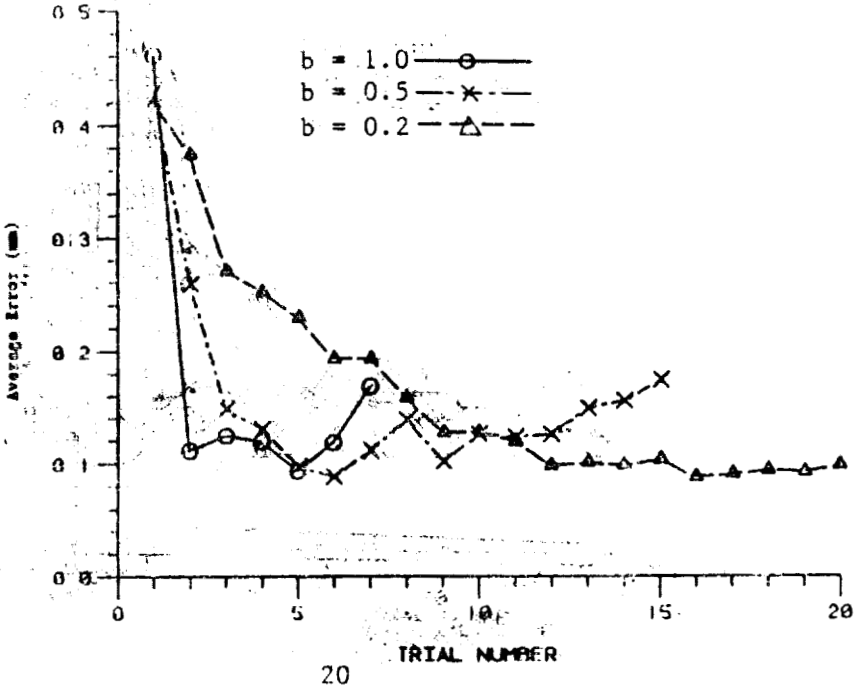


Fig. 21 STRAIGHT LINE ALONG Y AXIS,
ATTACHED PAYLOAD NONE, SPEED 150



NASA SCIENTIFIC AND TECHNICAL DOCUMENT AVAILABILITY AUTHORIZATION (DAA)

To be initiated by the responsible NASA Project Officer, Technical Monitor, or other appropriate NASA official for all presentations, reports, papers, and proceedings that contain scientific and technical information. Explanations are on the back of this form and are presented in greater detail in NMI 2230.1B, "NASA Scientific and Technical Document Availability Authorization."

☐ Original
☐ Modified

(Facility Use Only)
Control No. 112079
Date

I. DOCUMENT/PROJECT IDENTIFICATION

Title: ...ions of Artificial Intelligence (AI) and Robotics (19...) Conf...

Author(s): ... Littlefield and D. Bevers

Originating NASA Organization:

Performing Organization (if different)

Contract/Grant/Interagency/Project Number(s)

Document Number(s) Document Date: May 1987

For externally published documents or presentations, enter appropriate information on the intended publication such as name, place, and date of conference, periodical or journal title, or book title and publisher:

II. AVAILABILITY CATEGORY

Check the appropriate category(ies):

Security Classification: ☐ Secret ☐ Secret RD ☐ Confidential ☐ Confidential RD ☒ Unclassified

Export Controlled Document - Documents marked in this block must be routed to NASA Headquarters (International Affairs Division) for approval.

☐ ITAR ☐ EAR

P. 718

NASA Restricted Distribution Document

☐ FEDD ☐ Limited Distribution ☐ Other - See Section III

Document disclosing an invention

☐ Documents marked in this block must be withheld from release until six months have elapsed after submission of this form, unless a different release date established by the appropriate counsel. (See Section IX).

Publicly Available Document

☐ Publicly available documents must be unclassified and may not be export-controlled nor restricted distribution documents.

III. SPECIAL CONDITIONS

Check one or more of the applicable boxes as the basis for special restricted distribution if the "Other" box under NASA Restricted Distribution Document in Section II is checked. Guidelines are provided on reverse side of form. This document contains:

☐ Foreign government information ☐ Preliminary information
☐ Commercial product test or evaluation results ☐ Information subject to special contract provision

Check one of the following limitations as appropriate:

☐ U.S. Government agencies and U.S. Government agency contractors only ☐ NASA personnel and NASA contractors only
☐ NASA contractors and U.S. Government agencies only ☐ NASA personnel only
☐ U.S. Government agencies only ☐ Available only with approval of issuing office

IV. BLANKET RELEASE (OPTIONAL)

All documents issued under the following contract/grant/project number ______ may be processed as checked in Sections II and III.

The blanket release authorization granted ______ is:
Date

☐ Rescinded - Future documents must have individual availability authorizations.

☐ Modified - Limitations for all documents processed in the STI system under the blanket release should be changed to conform to blocks as checked in Section II.

V. PROJECT OFFICER/TECHNICAL MONITOR

...Chief of DSTD 520 [signature] 5/9/88
Typed Name of Project Office/Technical Monitor Office Code Signature Date Signed

VI. PROGRAM OFFICE REVIEW

☐ Approved ☐ Not Approved

... Dir. of MO&DSD [signature] 5/.../88
Typed Name of Program Officer Program Office and Code Signature Date

VII. INTERNATIONAL AFFAIRS DIVISION REVIEW

☐ Foreign publication/presentation approved. ☐ Export controlled limitation is approved. ☐ Export controlled limitation is not applicable.

☐ The following Export controlled limitation (ITAR/EAR) is assigned to this document:

NAME & TITLE

VIII. EXPIRATION OF REVIEW TIME

The document is being released in accordance ... of submission, as specified by NMI 2230.1B.

Name & Title

Note: This release procedure cannot be used ...

NASA ...

IX. DOCUMENTS DISCLOSING AN INVENTI...

a. This document may be released on ______

12 - TM

Installation Patent or Intellectual Property C...

b. The document was processed on ______

NASA STI Facility

X. DISPOSITION

Forms not approved are to be returned ...

Completed forms should be forwarded to the ... Information Facility, P.O. Box 8757, B.W.I. ... either (check box):

16/026 — 1037

☐ Printed or reproducible copy of docu...

☐ Abstract or standard bibliographic ... Information Facility at the above ...

FF427 Rev DEC 84

NASA
FORMAL
REPORT