OPTIMUM-AIV
A Planning and Scheduling System for Spacecraft AIV

M. M. Arentoft and J. J. Fuchs, Computer Resources International, Denmark *
Y. Parrod and A. Gasquet, MATRA ESPACE, France
J. Stader, AIAI, University of Edinburgh, Scotland
I. Stokes, Progespace, France
H. Vadon, ESA/ESTEC, The Netherlands

January 31, 1991

Abstract

This paper presents a project undertaken for the European Space Agency (ESA). The project is developing a knowledge based software system for planning and scheduling of activities for spacecraft assembly, integration and verification (AIV). The system extends into the monitoring of plan execution and the plan repair phases.

The objectives of the contract are to develop an operational kernel of a planning, scheduling and plan repair tool, called OPTIMUM-AIV, and to provide facilities which will allow individual projects to customize the kernel to suit its specific needs. The kernel shall consist of a set of software functionalities for assistance in initial specification of the AIV plan, in verification and generation of valid plans and schedules for the AIV activities, and in interactive monitoring and execution problem recovery for the detailed AIV plans. Embedded in OPTIMUM-AIV are external interfaces which allow integration with alternative scheduling systems and project databases.

The current status of the OPTIMUM-AIV project, as of January '91, is that a further analysis of the AIV domain has taken place through interviews with satellite AIV experts, cf. [6], a software requirements document (SRD, [7]) for the full operational tool has been approved, and an architectural design document (ADD, [8]) for the kernel excluding external interfaces are ready for review. At the time of the conference the implementation will be well underway expecting a final delivery in September of '91.

*Bregnerødvej 144, DK-3460 Birkerød, Denmark, Phone: + 45 4582 2100, Fax: +45 4582 1766, E-mail: mma@spd.cri.dk

1 Introduction

The size and complexity of the tasks involved in the AIV of spacecraft, raises the need for efficient and flexible planning and scheduling tools. An evaluation of the current available and applied commercial tools reveals their inadequacies towards the general problem of AIV.

In 1988 this lead ESA to award a contract to a consortium consisting of CRI, MATRA ESPACE and AIAI, which should assess the applicability of AI and KBS techniques in a prototype AIV planning and scheduling tool. This study resulted in a set of user and software requirements and a demonstration system exploring some of the aspects of AIV planning, cf. [5].

OPTIMUM-AIV is a follow-up project carried out by CRI, MATRA ESPACE, AIAI, and Progespace. The objectives of the project are three-fold:

1. to develop an operational kernel of a planning, scheduling and plan repair tool consisting of a set of software functionalities for assistance in
   - initial specification of the AIV plan
   - generation of valid plans and schedules for the various AIV activities
   - interactive monitoring of the AIV plan execution
   - identification of immediate effects and plan repair of problems
2. to embed external interfaces which allow integration with alternative scheduling systems and project databases.

3. to provide facilities which will allow individual projects to customize the kernel to suit its specific needs

The realization of these objectives are explained in the sections to come.

The outline of the paper is as follows. First section 2 outlines the operations domain of spacecraft AIV planning, and the benefits and applicability of OPTIMUM-AIV to this domain are introduced in section 3. Based on this outline section 4 lists the explicit domain dependent knowledge to be included in the tool. Before the detailed discussions of the main tool components section 5 provides an overview of the system process stages. Next section 6 shows how plan specification admits the user to consult libraries of past and generic plans and section 7 explains how the generation of plans takes into consideration the logical precedence ordering between activities and specifications of the expected outcome and required configuration of the spacecraft equipment being put together and tested. Then the satisfaction of temporal and resource usage constraints are described. Afterward, the execution monitoring and plan repair is presented, in section 8, as plan status updating, progress interpretation, and consistency checking and recovery organized along identified execution problems. Subsequently, section 9 lists the external interfaces to be embedded in the system and explains about their intended use. Finally, the lessons learned wrt. use of AI techniques in the system are discussed, in section 10.

2 Operations Domain: Spacecraft AIV Planning

This section gives a brief outline of the AIV planning process and life cycle, and hence establishes the function and purpose, environmental considerations, and general constraints of OPTIMUM-AIV.

Spacecraft development projects are typically divided into the following phases:

A : Early feasibility study:
The overall mission objectives of the intended programme are evaluated and a feasibility assessment is made based on operational constraints. This serves as a basis for deciding whether the project should be undertaken or not. The goals of the phase are to derive system requirements, to establish a preliminary model philosophy, and to identify verification aspects and assess their influence on the spacecraft design. An early identification of the needed verification tools and a general planning of the programme and the AIV aspect are also undertaken.

B : Specification phase:
System requirements are extracted in a top-down manner, starting with the total system and ending up with specifications of the various primitive units. The phase is critical to AIV since it must clearly define the AIV approach to be taken in phase C/D. During phase B the general AIV plan, additional facilities plans, ground support equipment (GSE) requirements, test hardware requirements and development plans at lower levels are produced. The general AIV plan is part of the overall project management plan.

C/D : Development and integration phase:
In this phase the design is frozen and manufacturing is undertaken. This is mainly a bottom-up activity where primitive units are put together to form assemblies, assemblies to subsystems, and subsystems are integrated at the system level. The phase is completed with the integration, verification and qualification of the spacecraft system.

This phase implements the plans generated in phase B and produces detailed AIV plans at different levels. This involves propagation of logical constraints, assignment of dates to activities logical flow, verification of resource consumptions vs. availability, and monitoring of the execution, i.e. updating of activity statuses and handling of failures.

E : Operational phase:
This last phase covers the period from the launch until the end of the spacecraft mission. Verification as such is completed before this phase. The electrical GSE has served its purpose and the satellite is instead controlled and operated by the ground segment.

The phases described above defines the AIV life cycle:

Elaboration : Phases A+B:
Philosophy and model(s) are selected, e.g. prototype, protoflight, or project specific philosophy, and structural, thermal, electrical, protoflight, or flight model. Furthermore an evaluation of project parameters takes place, for instance of due dates, manpower, GSE, and cost aspects.

Implementation : Phase C/D:

Detailed AIV plans, at different levels and with various time windows, are generated. The required initial configurations and effects of activities are compiled into a sequencing logic, and time and resources are verified and assigned. The resulting schedules are documented in detailed and summary networks.

Monitoring : Phase C/D:

The sequencing logic, temporal and resource usage constraints, and possibly AIV objectives, are reviewed in case of disturbance. The impacts are analyzed and critical and degraded activities are identified. New AIV plans are produced taking into account the current constraints and additional tasks for repair.

Phase E is only relevant if there are AIV activities during the operation of the spacecraft, e.g. if a reusable module has to be integrated again.

3 System Objectives

This section introduces the benefits and applicability of OPTIMUM-AIV.

The planning tool will assist primarily at the AIV team leader level in the management of day-by-day activities, and secondarily at the project group (interface) level.

The tool will cover the phases B and C/D. In phase B project management will define the high level AIV plans, which must be refined and detailed in phase C/D in order to constitute operational plans. OPTIMUM-AIV will provide a dynamic environment in which the AIV plans can be input and refined using AI based planning and scheduling techniques. Also in phase C/D, the constructed plans are executed. OPTIMUM-AIV will be used to monitor the progress, and to assist the users in identifying and solving any unforeseen problems and in producing new plans.

OPTIMUM-AIV shall provide the following facilities:

- definition, from scratch and as extension, of the AIV plan
- derivation and construction of plans and schedules at several levels
- monitoring and assistance in replanning of project execution

Especially on the last point the planning tool will differ from current planning systems, and assistance in replanning will indeed be the principal objective of OPTIMUM-AIV. The advantage of using knowledge based techniques will be a more flexible system in which planning knowledge is explicitly represented. The knowledge concerning conflict resolution and replanning will be incorporated and used to assist the users in generating feasible plans and solving problems during plan execution.

4 System Knowledge

One important aspect of OPTIMUM-AIV, and an aspect which makes it different from traditional planning and scheduling tools, is the inclusion of explicit domain knowledge.

We distinguish entity knowledge from process knowledge.

Entity knowledge defines and represents the entities that must be manipulated in the domain. For the AIV planning domain the entity knowledge is the knowledge concerning spacecraft systems and models, generic, past and current projects and plans, AIV activities, resources, and global constraints.

Process knowledge represents the knowledge stating how the entity knowledge manipulation may be done. For the AIV planning domain the process knowledge is the general planning and scheduling knowledge, plus explicit heuristics and knowledge about the rationale behind the plan structure.

The entity knowledge is represented as objects in classification hierarchies, which are also applied in the current management of large AIV programmes.

Spacecraft systems and models are decomposed in a part-of hierarchy.

Projects and plans are modularized so that each subplan may be worked on independently without necessarily having to load the full ensemble of AIV plans.

AIV activities have been classified according to their function in the AIV process, e.g. reception, preparation, assembly, integration, functional/performance test, environmental test, etc. Alternative classification dimensions could be the technological nature of activities or the spacecraft system which is the object of the

activity. Indeed the AIV activity hierarchy may be mapped onto the spacecraft part-of hierarchy through relations expressing that an AIV activity is performed on certain spacecraft subsystems. The use of this type of mapping is to verify the actual AIV plan against rules and policies for subsystems in the spacecraft decomposition.

The description of an activity carries over to the description of a project, or plan, since a plan is simply a description of a larger capability, or macro-activity. This macro-activity is expanded into a subplan of child activities, which in turn may be expanded into more detailed subplans, and so on. Each subplan is independent from other subplans in the sense that it has a unique starting point and a unique completion point which are exactly equal to the start and completion of the parent activity, i.e. each activity is decomposed into sub-activities independently.

Individual activity descriptions contain information about:

- preconditions which must hold for the activity to be appropriate and to be triggered,
  e.g. house.keeping.module CONNECTED.TO
  data.handling.subassembly
- effects (and side effects) of using the activity,
  e.g. tm-tc-loc INTEGRATED.IN
  house.keeping.module
- constraints, including time and resources, e.g. the activity requires two electricians for three days
- the activity itself such as its objectives, documentation, etc.

An activity is generally regarded as a plan fragment in its own right. Hence there may be partially ordered activities contained within the description. However the planning system assumes responsibility for completing the partial order description of the final plan. These descriptions are also generally parameterised, generic descriptions which are instantiated at the time of use. This offers flexibility and assists with a least commitment approach to planning and scheduling.

Resources have been categorized along two dimensions. First, according to predefined resource classes, e.g. GSE, manpower, test facilities, money, etc. Secondly, according to the nature of the resource, i.e. shared or consumable. Resources are shared if their availability must be specified as a function of time, e.g. manpower. Resources are consumable if there is an initial stockpile available which can only be depleted by activities in the plan, e.g. money.

Resource descriptions contain information about:

- availability profile, e.g. the resource is present during the third week of May
- alternative and indirect resources
- the resource itself

As in the case with spacecraft systems and activities the resources are also classified in an object hierarchy where generic functions are inherited to the specific resource instances.

Activity global constraints can be associated with a schedule. They express overall temporal relations between activities in a certain context. The context is defined by a given status of the involved activities and a certain configuration of the spacecraft system. The context may contain arbitrary variables that may be related in general predicates. For instance we have:

```
IF    ACTIVITY acoustic.and.vibration.test
      of.Class environment.test
      works.on.System ?s
      ∧
      ACTIVITY ?x
      of.Class integration
      works.on.System ?e
      ∧
      SYSTEM ?e SUBELEMENT.OF ?s
      ∧
      ?x ≠ power.supply.subsystem.integration
THEN acoustic.and.vibration.test AFTER ?x
```

The process knowledge is represented as rules and tables recording user preferences and decisions.

Heuristics are associated with projects and/or individual activities. Heuristics are different from constraints; they are used to decide on strategies in order to restrict the search space. These strategies are applied when all constraints have been satisfied and the system is left with degrees of freedom allowing it to follow user preferences.

The rationale behind the plan structure is written in a structure that explains how certain spacecraft system configurations have been satisfied at specific points in the current plan and records the alternative satisfaction means that were considered. This structure is generated during planning and scheduling and used

to restore consistency when execution problems occur. The idea of rationale recording originates from [2], [3], and [4].

5 System Process Stages

An explicit distinction of the different stages of the system processes helps clarify the purpose and rationale of the system functions:

- Knowledge Editing and Plan Specification
  - Definition and input of general domain knowledge: spacecraft system, activity, resource, global constraint classes and instances
  - Specification of the actual planning problem: project events and strategies, relations between domain object instances
- Plan and Schedule Generation
  - Planning: find a logically valid plan in sufficient detail
  - Scheduling: include time and resources in the plan
- Project Monitoring and Plan Repair
  - Monitoring the execution of the schedule: record the progress, remind the user of activities to be started soon
  - Detection of problems and their immediate impacts: derive local inconsistencies
  - Schedule repair: reschedule or edit the current schedule locally, e.g. up to the next milestone
  - Plan repair: more serious problems may interfere with the plan logic

The first stage requires mostly user input and thus editing facilities, and some support like input validation etc. The latter stages make use of the information gained at this first stage and they may require additional information from the user. These latter stages are more interactive. In particular the monitoring and plan repair stage require an extensive dialogue with the user.

The second stage build the plan and the schedule in advance of it being required. It also records justifications for planning decisions taken by the user in resolving conflicts.

The predictive approach adopted for the second stage is poor on recovery when failure arises. Since the third stage must support ease of repair we have chosen a more reactive approach for the project monitoring and plan repair stage. It will proceed forward in time only, using dependency recording techniques aimed at enabling repair to be limited to only those components known to be affected by the forced changes.

The different approaches for plan and schedule generation and for execution problem recovery reflect the fact that

> "... there is a trade-off between the predictability of the environment in which the plan is to be executed and the degree of reactivity which is necessary to successfully achieve the goals of the plan ...",

quoting directly from [1], p. 124. OPTIMUM-AIV is capable of admitting both a predictive approach and ease of repair.

There is not an overall strategy management set of rules, as it is doubtful if the human AIV expert will be able to balance the requirements of the technical objectives along with each of the strategic objectives. Rather the system recognises the complexity of objectives and devises a multi-perspective strategy aimed at meeting *all* of the objectives, rather than a single, or minimal, perspective focussed around some of the objectives such as handling all AIV tasks, meeting all deadlines, or matching resource demands with availabilities, e.g. estimated costs to budgeted costs.

6 Knowledge Editing and Plan Specification

In this initial stage most of the work relies on the user inputting and specifying the AIV activities, their required spacecraft components, resources, and interfaces, as well as their decomposition which will eventually constitute a plan for the project. This information appears from the AIV plan defined by project management in phases A and B of the spacecraft AIV life cycle. The information is entered through structured editor environments that has a simple compiler convert the input to internal form.

The system enables the user to retrieve information from past AIV plans and to browse into the activities

of these plans. The past plans are indexed according to their main characteristics: the type of the spacecraft system, the types of AIV models used, and the subsystem of the spacecraft for which the plan has been applied. It is thus possible to incorporate heuristics specifying typical scenarios and durations of certain activities and to assist in the assessment of project duration, resource consumption and cost. In this way optimistic, probable, and pessimistic estimates can be based on experience.

The experience has been recorded during past AIV programme executions. The recording is facilitated through a node pad facility and special activity, resource, and global constraint attributes, where the user may record previous results of using the system. Here the user may comment on the actual performance of an activity, his experience in using a resource or in applying a global constraint. These experience attributes complements the information that can be derived from comparison of estimated versus planned values.

The case-based approach of using past plans is combined with the use of generic plans. Generic plans specify a number of typical activities for a certain (component of a) spacecraft system. The assumption is that general principles of spacecraft AIV may guide the initial plan establishment. The activity attributes could define in which order they must be undertaken. Generic plans, or prototype plans, are thus a collection of imaginary AIV activities which must typically be undertaken to perform the ideal AIV process for a selected ideal system or model. They are generic in the sense that no actual programmes will ever be able to use the plans without making modifications to them. Furthermore they are generic in the sense that all the generic activities and the generic resources must be instantiated to represent the actual world. That is, scheduling information, precise resource specifications, etc. must be added in order to properly instantiate the activities to represent an actual plan, or schedule. OPTIMUM-AIV provides mechanisms which allows the user to search through the various generic plans and to make instantiated versions which can be used as the basis for actual AIV plans.

7 Plan and Schedule Generation

Constraint Satisfaction

During the plan and schedule generation stage we distinguish five kinds of activity constraints. These constraints are imposed by the environment of the planning problem, e.g. resource usage constraints, or by the nature of the problem, e.g. precedence constraints.

Precedence :
These predecessor and successor constraints specify explicit user defined orderings on the activities. The constraints are expressed as directed links between activities They may be used to constrain the temporal specifications of the activity. That is, if any of the predecessors or successors are committed to certain time feasibility windows, then this may reduce the duration of the possible time window of the current activity.

Precondition :
These are the more general constraints, which may specify that certain results must have been obtained, or some equipment be available before the activity can be undertaken. The former type of condition may be used to constrain the plan network by adding predecessor links to the activities which have the required condition as an effect. The latter type of constraint may be used to expand the current plan through the addition of e.g. transport activities which will have the required condition as its effect.

Temporal :
Time is one of the major constraining factors in the spacecraft AIV planning domain. Temporal constraints are manifested in a number of ways. First, through the specification of delivery and completion dates, which the various activities must be scheduled to satisfy. Secondly, as a maximum duration which the activity must be undertaken within. These types are what may be called absolute temporal constraints. Furthermore a number of relative or second-order temporal constraints may be deduced. They are deduced on the basis of precedence relations and on the possible temporal limitations on the availability of e.g. resources. Indeed the establishment of temporal specification from precedence relations

is one of the major activities in traditional OR scheduling algorithms.

Resource Usage :
This type of activity constraint specifies which and how much of various resources an activity demands. The constraints are expressed as references to resource instances, and a specification of the required consumption profile. The information is used to constrain the time feasibility windows in which the activities can be scheduled.

Global Activity Constraints :
Global activity constraints express overall temporal relations that must hold given a status of the involved activities and of the spacecraft system.

Verification of precedence and precondition constraints takes place during planning, temporal and resource usage constraints are propagated during scheduling, whereas global activity constraints can be satisfied at any system process stage. Typically, however, global activity constraints will be checked after scheduling by simple goal processing, or backward chaining, through the context/temporal relation rules.

Planning

Plan generation entails verification of the plan logic, assistance in conflict resolution, and construction of new precedence relations based on preconditions and effects of activities. The basis is the initially specified AIV plan which must be refined and detailed.

The plan logic verification is divided into checking of user-defined precedence relations between activities and validation of preconditions and effects of activities vs. actual spacecraft system states.

The checking of precedence relations includes detection and resolution of dangling references to predecessor and successor activities, and of cycles specified by the precedence links.

The validation of activity and spacecraft system states checks for interactions between parallel activities and propagates system configurations from the start activity to the project due date. The propagation ensures that the effects of one activity do not violate the preconditions of a succeeding activity, i.e. that the ordering of activities is consistent with the preconditions and effects specified for each activity. There might be two types of conflicts: a precondition of an activity is

1. in conflict with the actual state of the system
2. not found in the actual state of the system

Possible modifications to restore the consistency of the plan logic are

- modification of the preconditions and/or effects of one or many activities
- change in the precedence relations between activities
- addition or deletion of activities to introduce or avoid the configuration in question

Scheduling

Schedule generation involves management of temporal and resource usage constraints; verification, conflict resolution, and construction. The relation between an activity and its decomposition is an active relation in the sense that definitions and changes made at one level propagate to the other levels in the plan hierarchy. This holds true both for time definitions and resource assignments.

Time feasibility windows (TFW) for the activities are calculated by a forward and a backward pass of the logical plan. These passes work on estimated bounds, rather than exact values, as specified by the user for activity durations and times.

Within the TFW's the system places the activities according to the local strategy associated with each activity. If a local strategy is not specified the project as a whole has defined a global strategy which determines the preferable assignment of actual times to activities.

These preferences regarding activity placements may be overruled by violation of resource constraints. The management of shared resources, called resource smoothing, is an inherently intractable problem. We have constructed the following simple heuristics for resource smoothing: calculate the shared resource usage profile, compare this profile with the availability profile, shift activities within their TFW's if necessary, and, if shifting is not sufficient, solve the overconsumption problem in cooperation with the user.

The management of consumable resources is simplified by the fact that the timing of their usage is unimportant. Only the total usage is relevant and the system must simply guarantee that this does not exceed the initial availability. Our problem is one of representing and propagating resource consumption constraints in an efficient manner.

By specifying upper and lower bounds on resource usage it is a relatively simple and flexible task to track the more detailed specification of actual resource usage as a plan is refined into lower levels of detail, as in hierarchic planning. In fact the maintenance of bounds makes this a useful checking and pruning mechanism, as the expectation should be that lower levels of detail merely provide a more accurate specification of actual use and should not be unnecessarily constrained by complete specifications at higher levels.

Although the resource management algorithms bear some similarity to the maintenance of temporal constraints there are important differences which make resource management of a strictly consumable resource easier, cf. [4], p. 21:

- time is not a resource because the consumption of time by activities in parallel is independent of the number of actions in progress. Thus it is inconvenient to represent the consumption of time by a particular activity.
- the sheer number of temporal constraints is often far greater than the number of resource constraints.
- Resource constraint propagation is the maintenance of a conjunction of constraints of a particularly simple form. With temporal constraints, the planner must often impose additional constraints to satisfy a condition. Such constraints are often a disjunction of simpler constraints.

Knowledge About Plans and Schedules

During the planning and scheduling processes information is created about the proposed plan. This information consists not only of the plan structure itself but also of information about the structure. Together these make up a knowledge rich representation of the plan. Information about the reasons for ordering activities in a certain way, about which states and domain objects are affected by which parts of the plan, and about which activities and/or links were introduced into the plan for what purpose is recorded at this stage. Individual conditions required in a plan are deliberately tied to the effects which will necessarily ensure their satisfaction.

The advantage of a knowledge rich representation is that it enables the system to reason about the plan. This is a necessary requirement for determining immediate impacts of changes to the plan during execution monitoring and for assistance in consistency recovery and plan repair. It is also a useful precondition for detecting and possibly avoiding conflict situations and for explaining and justifying planning decisions.

8 Project Monitoring and Plan Repair

The major use of OPTIMUM-AIV will be during the plan execution phase. This phase covers the period from the time when the AIV plan starts to be executed until the planned process is completed.

In any planning domain there is a possibility that problems occur when the plan is executed. We distinguish between usual plan failures and plan failures which are specific to the AIV domain: the failure of tests.

The usual plan failures are caused by changes in the actual environment in which the plan is executed. They are often caused by unexpected events or organisational issues like unavailable resources or supplies. When problems occur during plan execution the original plan becomes, at least partly, invalid and it has to be revised. This can be done by replanning where the original plan is discarded and a completely new plan is generated that takes into account the state of execution and the changes of circumstances that lead to the plan failure. In domains like AIV where activities are heavily interdependent to external activities or external resources this approach is not acceptable. Based on the original plan, bookings have been made and temporal interfaces have been specified. It is important that these interfaces are changed as little as possible and thus as much as possible of the original plan is to be retained. In these cases the plan has to be repaired.

The failure of tests make the original plan invalid just like the other problems mentioned above. However, they are expected as possible outcomes of tests and thus in most cases lines of action are predefined, as part of the domain knowledge, to deal with the situation of a test failure. There are various strategies that can be adopted depending on the type of failure. These strategies, or problem scenario subplans, are represented as special cases of generic plans. Some problem scenario plans may be included in the system from scratch, but most will be entered during the actual use of the system in the plan execution phase.

The severity of plan failures varies. Some plan failures have very local effects and problems can be solved by

local schedule repair. It may suffice to move start and finish times of a few activities, or to make use of alternative resources or non-nominal availability profiles. However, sometimes this local change of the schedule has effects on other parts of the schedule, e.g. when the alternative resource is scheduled for another activity. In these cases more global changes are necessary to generate a new valid schedule. Effects of plan failures can become so serious that the logic of the plan is affected. It may be impossible to schedule still outstanding activities using the original plan under the given conditions. Then it becomes necessary to repair the actual plan itself rather than just the schedule in order to retain consistency.

The issue of plan repair and schedule repair is very much a research issue in AI planning and thus a fully automated solution to these problems is far beyond the scope here. OPTIMUM-AIV *assists* the user in schedule and plan repair in an interactive way, rather than *performing* repair itself.

The possibility of plan failures, whether expected or not, and the need for plan repair require the monitoring of plan execution. In the simplest case the changes in the current state of execution are given when a plan failure occurs. However, a plan which has not been updated for a long period while work has been undertaken will be more difficult to recover. In the AIV domain where plan failures are frequent and there is need for fast plan repair it is necessary to monitor plan execution more closely. It is important that the plan is kept up-to-date regularly without much effort. Therefore the system gives strong facilities for entering the execution progress and for making small and large adjustments to the plan.

The system gathers information about the actual progress to have basis for determining whether the plan has failed. Then it uses that to determine how execution goes, i.e. to detect discrepancies between the proposed schedule and the monitoring information. This in turn is used to identify what parts of the schedule are inconsistent with, or affected by, the execution state. The system assists in changing the schedule by displaying relevant information in trying to repair the schedule. Finally it is checked whether the new schedule, user or system generated, is consistent.

The consistency checking and recovery is organized along execution problem types caused by user changes. We assume that the user is quite capable of solving execution problems him/herself and will use the system to speed up the process, to make sure that nothing is forgotten and to explore different solutions, what-if scenarios. For each execution problem type there are specific standard recovery methods that are most appropriate for solving the problems. However most problems can be solved in other ways too, which will also be available for use.

9 External Interfaces

ARTEMIS Interface

The system will have embedded an interface to the widely spread ARTEMIS scheduling tool. The interface is intended to be used primarily for

- import of space project data, i.e. activities and events (but not interfaces and hammocks), constraints, and resources datasets,
- export and display of plans, and
- report writing and graphics
- aggregation, i.e. summarize numeric data held in network datasets, e.g. resource requirements for all activities.

It can also be used for network construction, examination of the network logic, time analysis and updating, resource-limited or time-limited scheduling, and multiple network processing. However, in these latter uses of ARTEMIS it will not be feasible to return the results directly to OPTIMUM-AIV.

Database Interface

OPTIMUM-AIV will be able to interface to satellite related project databases. This reduces the work to input data into the system, and ensures the coherence between external satellite databases and the system database.

It allows the loading of activities, resources and constraints from such external databases. The management system and the format of those databases might vary but they are convertible into relational tables. Therefore the system essentially provides an SQL interface to fill its internal database.

Programming Interface

The system is designed such as to allow external documentation programs to be written. It provides an

interface that permits any user to develop their own documentation, in particular any new representation of the plan and schedule. That means that all activities, resources and constraints and any schedule will be accessible by any external program (written in C, Pascal or Ada).

10 Applied AI techniques

This section extracts AI planning and scheduling techniques integrated in the system. The applied techniques complement existing features of current project management tools, and in particular of ARTEMIS.

OPTIMUM-AIV adopts the *non-linear planning* paradigm which enable plan representation to contain causally independent activities which can be executed concurrently. It searches through a space of partial plans, modifying them until a valid plan/schedule is found.

Another important characteristic of the system is *hierarchical planning*. The term hierarchical refer to both the representation of the plan at different levels, and also the control of the planning process at progressively more detailed levels.

The scheduling task is considered as a *constraint satisfaction* problem solved by constraint-based reasoning. The constraints are propagated throughout the plan, gradually transforming it into a realizable schedule. Invariably not all of the constraints can be met, such that some have to be relaxed.

During plan specification and generation the system operates on *explicit preconditions and effects of activities* that specify the applicability and purpose of the activity within the plan. With this knowledge it is possible to check whether the current structure of the plan introduces any conflicts between actual spacecraft system states, computed by the system, and activity preconditions, which have been specified by the user. Such conflicts would arise if one activity deletes the effect of another thus removing its contribution to the success of a further activity. The facility for checking the consistency of the plan logic, by dependency recording, is not possible within existing project management tools, that assumes that the user must get this right.

Also during planning, the system *records the rationale* behind the plan structure i.e. user decisions on alternatives are registered. This is used to assist during plan repair where the user tries to restore consistency. Information can then be derived about alternative activities, soft constraints that may be relaxed, and potential activities that may be performed in advance.

11 Conclusion

The current status of the OPTIMUM-AIV project, as of January '91, is that a further analysis of the AIV domain has taken place through interviews with satellite AIV experts, cf. [6], a software requirements document (SRD, [7]) for the full operational tool has been approved, and an architectural design document (ADD, [8]) for the kernel excluding external interfaces are ready for review. At the time of the conference the implementation will be well underway expecting a final delivery in September of '91. It is foreseen that the implementation will be in Common Lisp and Common Lisp Object System.

The domain analysis has identified areas of AIV expert knowledge which must be incorporated into the system. The SRD has derived a model of the domain and has established the functional and operational requirements which must be satisfied to meet the demands of the AIV experts. The ADD has proposed a design which supports the typical mode of interaction between the user and the system. The AIV experts input the plan specifications and revisions and the system goes through a series of analyses, and identifies, visualises, and assists the user in case of conflicts.

The paper has discussed in detail the requirements of the AIV experts, the obtained results with regard to knowledge elicitation and requirement formalization, and the results of the design phase in terms of constraint identification and satisfaction and execution problem detection and recovery.

References

[1] W. Swartout. *DARPA Santa Cruz Workshop on Planning.* 1987.

[2] A. Tate. *Goal Structure: Capturing the Intent of Plans.* European Conference on AI, Pisa, Italy, September 1984.

[3] D.E. Wilkins *Domain Independent Planning: Representation and Plan Generation.* Artificial Intelligence, 22, 1984.

[4] K. Currie and A. Tate. *O-Plan: the Open Planning Framework.* Final report to the SERC on Alvey grant number IKBS-151, AIAI, 1989.

[5] J.J. Fuchs, G.S. Pedersen, and A. Gasquet. *EPS-AIT: Final Report.* EPS-AIT-CRI-FR-0001-89, CRI, MATRA, AIAI, 1989.

[6] Y. Parrod and I. Stokes. *OPTIMUM-AIV: AIV Knowledge Acquired.* OPT-MATRA-TN1000-0890, MATRA, Progespace, CRI, AIAI, 1990.

[7] M.M. Arentoft, Y. Parrod, and J. Stader. *OPTIMUM-AIV: Software Requirements Document.* OPT-CRI-SRD-1090, CRI, MATRA, AIAI, Progespace, 1990.

[8] M.M. Arentoft, Y. Parrod, J. Stader, and I. Stokes. *OPTIMUM-AIV: Architectural Design Document.* OPT-CRI-ADD-0291, CRI, MATRA, AIAI, Progespace, 1991.