Real-Time Control for Manufacturing Space Shuttle Main Engines:

Work in Progress

Corinne C. Ruokangas
Rockwell Science Center, Palo Alto Laboratory
444 High Street, Suite 400
Palo Alto, CA 94301
ruokangas@score.stanford.edu

ABSTRACT

During the manufacture of space-based assemblies such as Space Shuttle Main Engines, flexibility is required due to the high-cost and low-volume nature of the end products. Various systems have been developed pursuing the goal of adaptive, flexible manufacturing for several space applications, including an Advanced Robotic Welding System [Sliwinski 87] for the manufacture of complex components of the Space Shuttle Main Engines. The Advanced Robotic Welding System (AROWS) is an on-going joint effort, funded by NASA, between NASA/Marshall Space Flight Center, and two divisions of Rockwell International: Rocketdyne and the Science Center. AROWS includes two levels of flexible control of both motion and process parameters: Off-line programming using both geometric and weld-process data bases, and real-time control incorporating multiple sensors during weld execution. Both control systems were implemented using conventional hardware and software architectures. The feasibility of enhancing the real-time control system using the problem-solving architecture of Schemer [Ruokangas 88] is being investigated and is described in this paper.

Schemer is a knowledge-based system designed to provide more complex real-time control by supporting real-time response to multiple and conflicting interrupts, and problem solving under time and resource constraints; on-going research supports the incorporation of techniques from decision analysis. The Schemer architecture is event driven and is similar to a blackboard system. It supports the interruption, suspension, and resumption of problem solving tasks, concepts which are essential to providing real-time response to changes in the environment [Fehling 87]. Schemer provides support for intelligent real-time modification of Off-line programming plans and resolution of sensor conflicts.

This paper summarizes the development of a prototype system for simulation of real-time control of robot motion and the welding process using multiple sensors. The system provides simulation of prioritized, event-driven responses to multiple sensors affecting multiple processes, with the sensors providing both cooperative and conflicting information. A single vision sensor is used to modify robot motion. In parallel with modifications of motion, multiple penetration sensor systems are simulated to affect weld-current values. Further development of the system will include additional sensor fusion techniques such as decision theory methods, and plan transformation rules. While the current implementation is on a Symbolics 3600 series system in Common Lisp, other hardware platforms are being evaluated for additional studies towards implementation in manufacturing.

INTRODUCTION

AI in Space Applications

There is widespread interest throughout the aerospace community in the application of Artificial Intelligence (AI) developments to both space and earth based activities, including efforts in planning, scheduling, and real-time control. Effective utilization of key AI components can support NASA in future space exploration, including Space Station and Mars Rover projects, satellites, and spacecraft communications and control.

"The primary goal of AI is to make machines smarter" [Winston 87], that is, to provide more autonomous systems which are more useful, competent and less demanding of human interaction. Significant development of space exploration and space-related projects, both flight and ground based, relies on the development and implementation of autonomous systems. An autonomous system, operating in a complex, dynamic environment such as space or manufacturing, should have the capability to both define and execute plans to achieve its goals. In many environments, limitations of time, information, and other critical resources constrain the determination and use of the plans. The system must be able to manage its reasoning and other activities to make the best use of available resources. Problem-solving under these conditions has been referred to as resource-bounded problem-solving - "controlling and adapting problem-solving actions to meet critical, contextually-determined constraints" [Fehling 88]. One arena in which resource-bounded problem-solving autonomous systems will prove invaluable is the manufacturing of space-based assemblies which must be robust enough to operate in remote locations. This paper discusses the development of such a system, to be applied to the manufacture of Space Shuttle Main Engine components.

A research area which holds promise for influencing advancements in autonomous systems is intelligent real-time control, or problem-solving under time constraints. Commercial AI shells, designed for the development of consulting type systems, provide inadequate support for real-time process control systems, i.e. those which require problem solving in a dynamic processing environment, in an interruptible and prioritized event-driven manner. Implementation of intelligent, flexible real-time control in real world environments has proven to be a considerable challenge to the AI research community. The resource-bounded problem-solving architecture described in this paper addresses several of the fundamental challenges intrinsic to intelligent real-time control, such as appropriate response to multiple and varying priorities of interrupts, and conflict resolution.

SSME Background

The flexible real-time control architecture described in this paper can be applied in various areas of motion and process control throughout manufacturing and aerospace activities. The specific application discussed here is the automated welding of complex parts of the Space Shuttle Main Engine (SSME), which could be advanced by a high degree of run-time adaptivity both for seam following and weld process modification. The SSME is a low-volume, high precision product. Because part geometries vary significantly, it is difficult to accurately predict the variations in geometric position and process parameters defined in a robotic welding schedule. During weld execution, the welding process itself can induce heat distortions, further complicating the task of automatic welding. In 1983, a development project was initiated by NASA Marshall Space Flight Center to support robot-based welding of components of the SSME. At that time, commercial robot systems did not provide either the desired adaptive real-time control nor off-line programming for motion and process control; the goal of the project was development of a technology base for penetration sensors, seam-tracking sensors, and integrated off-line generation of process commands [Sliwinski 87]. The Advanced RObotic Welding System (AROWS), developed by the cooperative effort of two Rockwell divisions and demonstrated for NASA in late 1987, incorporated two levels of adaptive control, as shown in figure 1.

• Off-line Programming (OLP) was based on existing geometric data bases and weld parameter data bases developed by interaction with Rocketdyne welding engineers. The overall function of the OLP system was initial generation of both motion and process commands, using graphical simulation.

• The adaptive real-time control system included the development and implementation of both motion correction and weld penetration sensors, as well as the Sensor Controller for overall coordination of the sensor subsystems and control of the robot and weld parameters during weld execution.

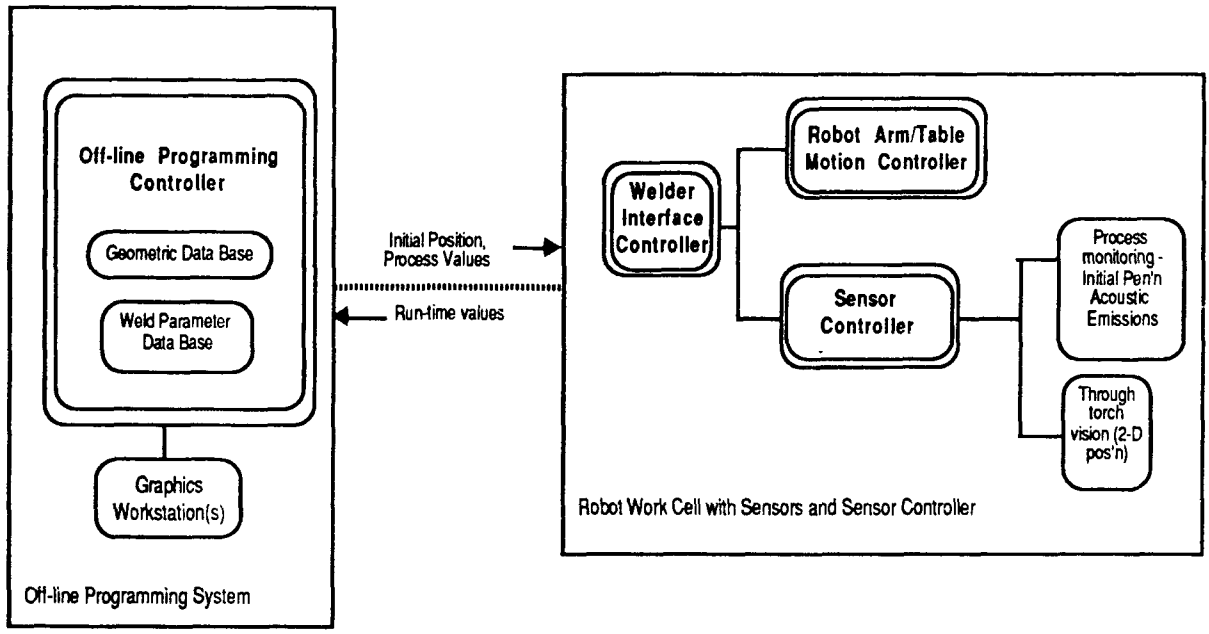


Figure 1: The adaptive control elements of the Advanced Robotic Welding System: Off-Line Programming and Real-time control

AI for Control Applications: Schemer

Conventional control and programming environments are often limited in several key areas. In commercially available AI shells, techniques are used which surmount several of these limitations; however, these shells are primarily designed to develop consultative, off-line systems rather than on-line or real-time systems. Schemer supports the application of AI techniques to real-time problem-solving environments. Specifically, Schemer is being used to enhance the real-time control of a robot and welding equipment for manufacture of the Space Shuttle Main Engines, utilizing a simulation environment developed at Rockwell Palo Alto Laboratory.

Limitations to Conventional Systems: Conventional control theory is frequently limited in its ability to provide control for unstable processes and for systems without quantitative models [Coughanowr 65]. A typical closed-loop control system gathers information from sensors, calculates an error between the sensor value and a desired setpoint, and applies an offset based on this error. To be successful, these systems rely on both accurate and repeatable sensor signal processing, and on control based on models of the dynamics of the process being manipulated. However, accurate processing of sensor signals requires a model relating information from the sensor to the process under control; if the sensor is under development, it is not well understood and a complete model is frequently not available. If randomness is found in the sensor data, or conflicting indications are determined between sensors, these inconsistencies in data can propagate to induce oscillations in the process under control.

In addition, conventional programming environments do not support iterative prototyping and modification. These techniques are essential to the development of incompletely specified systems, since modifications must be applied to the overall system as information becomes available. The ability to deal with uncertainty is an important concept, since models are frequently incompletely defined; most conventional environments do not support this concept.

Commercially available AI Systems: Several systems or shells such as KEE™ and ART™ are available for the development of consultant type systems, also known as *expert systems*. These shells provide symbolic representation and inferencing, and heuristic search. Applications include Prospector, Mycin, and ADDAMX [Garcia 87]. The applications are frequently diagnostic, tutorial or predictive [Hayes-Roth 83] in style, and primarily provide off-line rather than on-line or real-time control. They are also commonly fragile in dealing with unanticipated conditions, not incorporating the concept of graceful degradation.

Real-time Control - Schemer: While incorporating the strengths of AI shells, such as symbolic representation and rapid prototyping, the Schemer architecture was also designed to respond to varying priorities of interrupts occurring in dynamic environments. Schemer is event-driven, and thereby provides appropriate response to multiple asynchronous interrupts. It provides the basis for adaptive problem-solving under time and resource constraints; in addition, it can incorporate various problem-solving techniques, both mathematically and heuristic based. Hence, it is appropriate for use in real-time control. While Schemer is an architecture, it has been implemented in several variations. The current implementation, discussed in this paper, was developed at Rockwell Palo Alto Labs (RPAL). In addition, research is on-going at RPAL both in terms of the architecture and additional implementations.

AROWS-Schemer: The specific application discussed in this paper is the enhancement of real-time control of the Advanced Robotic Welding System (AROWS) using Schemer in simulation mode. The existing real-time control system, based on conventional hardware and software, has been constrained by the lack of models of both the overall welding process and of the sensors. The sensors were developed in parallel with the implementation of the existing controller; no detailed models relating sensor data to the weld process are available. In addition, it is possible for the sensors to generate conflicting information. Due to these constraints and the complexity of the process, it was determined that Schemer should be applied in a feasibility study to determine the impact of an AI architecure on the operation of the real-time controller. In this manner, the initial concepts of the real-time controller may be enhanced without the restrictions inherent to the sensor development process; parallel efforts continue in sensor research. The use of Schemer for this application was found to be appropriate, both in the ability to respond to multiple interrupts from various sources and in the ability to iteratively develop, modify and augment the application in a clear manner. The details of this feasibility study and the effectiveness of Schemer in this particular environment are provided in subsequent sections.

Further directions, AROWS-Schemer: Additional future efforts with respect to this particular project could include interfacing with actual workcell components. While the simulated data has been generated in a manner to map closely onto actual sensor data, some unforeseen problems could arise in interfacing with specific hardware platforms or specific sensor systems. Schemer has been designed, however, to support the concept of graceful degradation, as opposed to many expert systems which are brittle in situations which exceed their expertise. The Schemer response to previously undefined states will be based on its ability to deal correctly with continuous data and its lack of dependence on explicitly stated rules.

SCHEMER BACKGROUND

Schemer Architecture

Schemer is a resource-bounded problem-solving architecture, with multiple implementations. The AROWS-Schemer application has been developed on a Common Lisp implementation currently running on Symbolics 3600 series systems, Macintosh II systems, and Xerox 1100 series systems. Schemer supports the interruption, suspension, and resumption of individual problem solving tasks, concepts which are essential to providing real-time response to changes in the environment.

Schemer has proven especially useful as the problem-solving architecture for systems that must perform satisfactorily in complex, dynamic environments. Successful Schemer applications have been built for a number of real-time, "process management" applications such as diagnosis or control of complex manufacturing processes [D'Ambrosio 87], automated performance management of advanced avionics systems [Guffey 86] and monitoring and task-management in a distributed information processing system [Fehling 84].

As shown in Figure 2, the Schemer architecture is similar to a blackboard architecture.

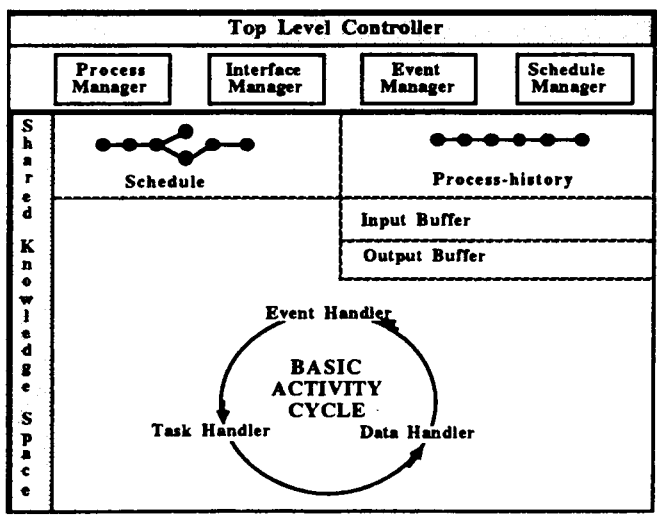


Figure 2: The Schemer Architecture:
A Top Level Controller providing prioritized interruptibility,
and Shared Knowledge Space for communication, containing Handlers, Schedule, and History

The architecture is comprised of three major components:

- Problem solving elements called Handlers or Procedural Elements (PEs), containing procedural and/or declarative knowledge;
- A global store, called Shared Knowledge Space, containing the Handlers as well as a current, prioritized Schedule of pending executable Handlers, and an audit trail, called the History;
- A Top Level Controller (TLC) that manages the system's activity.

The Shared Knowledge Space includes areas for communication with the environment (Input and Output Buffers), for communication of information between various problem solving elements (Handlers), and for state recording.

The overall control structure, the Top Level Controller, provides the prioritized interruptibility of the system. As indicated in figure 2, the TLC contains four Managers consisting of highly optimized code; the minimized execution time of a complete cycle provides maximum interruptibility. The TLC performs data transfer, determines which Handlers may be executed, maintains a variable-priority schedule, and causes Handler execution.

In the design of the Schemer architecture, every Handler is interruptible, and maintains its own local data space. The executable body of a handler may be a Common Lisp expression, a set of invocations of other Handlers, or an embedded Schemer.

This triad of major architectural components has been implemented in several instantiations. Following are further details of the specific implementation used for the AROWS-Schemer.

Schemer Implementation: TLC + PEs

This particular Schemer system implementation contains the basic elements interacting through a Shared Knowledge Space: the Top-Level Controller (TLC) which serves as the management structure, and Procedural Elements (PEs) which serve as the problem solving elements. The TLC controls execution of the overall process. The PEs include executable bodies and reference the Data Stores for inter-element communications and local state storage. The Shared Knowledge Space consists of the Schedule, a Data Store, and Ports.

The TLC consists of four managers, which repeatedly execute in the order indicated in figure 3. Since each manager consists of minimal code, the cycle responds rapidly to events in the environment, and in a prioritized manner. The PROCESS MANAGER executes the body of the first Procedural Element on the Schedule. The INTERFACE MANAGER queries all input / output (I/O) Ports to determine if any new data transfers are possible; it transfers all data available at input Ports to internal data storage areas, and transfers out-bound data from internal data storage to output Ports. The EVENT MANAGER, based on changes in internal Data Stores, determines if any Procedural Elements can be executed, i.e. whether the trigger conditions of any Procedural Elements have been satisfied, and enters these elements on the Schedule. The SCHEDULE MANAGER then executes any initializer code for new Procedural Elements on the Schedule, and orders the elements by priority. The cycle then repeats, with the PROCESS MANAGER executing the body of the highest priority Procedural Element on the Schedule.

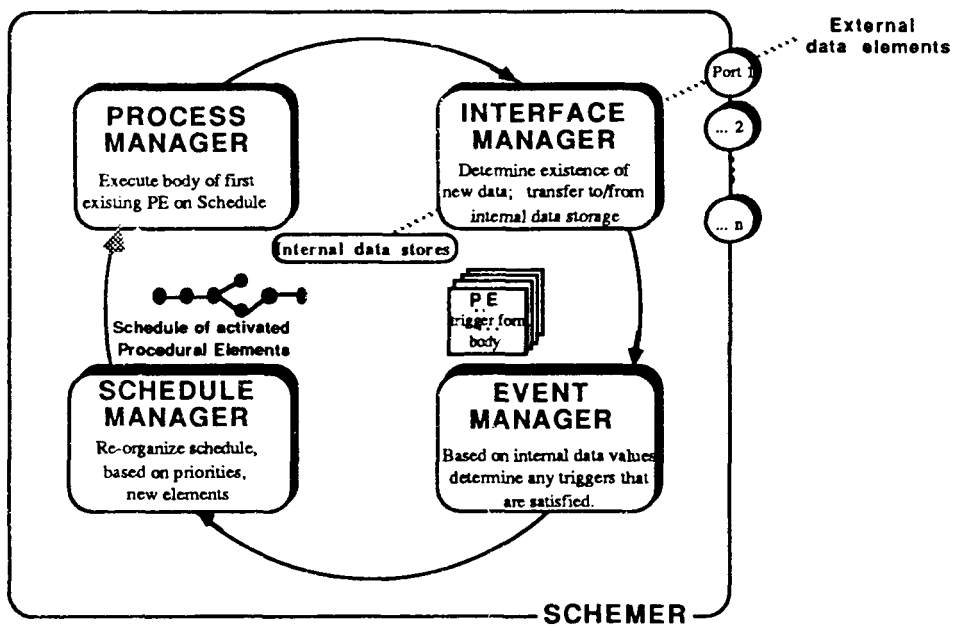


Figure 3: Overview of Schemer system implementation, including cyclic execution of the four managers in the Top Level Controller and the relationship of the Schedule, Procedural Elements, and external events.

Procedural Elements: Procedural Elements (PEs) are structures containing the Lisp or Schemer code to be executed based on the current state of the overall system. In addition to code, PEs also contain local variables, a list of relevant shared variables, and as appropriate, trigger conditions, initializing code, and priority levels. PEs may be activated in three manners: 1) their trigger conditions may be met, and they are scheduled by the Event Manager 2) another PE, during

its execution, may explicitly schedule them for deferred execution, and 3) another PE may explicitly call them immediately, as a "sub-routine" to the original PE. In general, a PE includes a trigger definition, priority, and initialization code to be executed as it is placed onto the Schedule; PEs that are activated by other PEs need not incorporate all of these elements. Normally, PEs which must respond to asynchronous events in the environment are triggerable with high priority levels. PEs requiring lower priority treatment, frequently of lower time-criticality, are activated within triggered PEs. Possible elements of a PE are shown in figure 4, with sample values included.

PROCEDURAL ELEMENT STRUCTURE

component	sample values
BODY | INNER SCHEMER	(setf mod-current ae1)
[TRIGGER]	(> confidence-ae1 confidence-ae2)
[BASE-PRIORITY]	ordinary
[INITIALIZER]	()
SHARED-VARIABLES	(mod-current ae1 confidence-ae1 confidence-ae2)
PERSISTENT-VARIABLES	()
ACTIVATION-VARIABLES	()

Figure 4: Structure of a Procedural Element, including optional fields and sample values

Future Schemer Enhancements

Resource-bounded problem-solving requires that the system be aware of its current and past activities and its future commitments, as well as the relationship of these factors to conditions in the environment. However, in most realistic environments, the information available to the system is most frequently incomplete, and subject to change. The system must be able to deal with uncertainty in its knowledge of the world, and use what knowledge it has to bring about effective actions. An initial aspect to reaching a solution to incomplete information is the ability to respond to changes in the dynamic environment by the prioritized execution of specific tasks triggered by the specific state; the existing Schemer implementation incorporates this capability. An additional solution is the incorporation of the ability to deal with uncertainty, specifically by using a probabilistic decision-theoretic approach. Such an approach to control reasoning can prescribe how the problem-solving system can select among multiple, alternative problem-solving methods on the basis of how well each method satisfies the basic problem requirements, resource constraints, state of information, and the system's priorities and attitude toward risk. Mathematical decision-theory [Savage 72] can be used to provide a rigorous, verifiable and domain-independent basis for problem-solving control. Development is currently in progress to incorporate into Schemer the control of a system's actions based on decision analysis [Breese 88]; a basic mechanism is under development in the form of a set of general, domain-independent principles according to which the system controls and coordinates its actions under uncertainty.

Additionally, parallel research efforts are involved in modifying the TLC cycle, so that Managers are not necessarily executed in a defined loop. Rather, the Managers themselves will be event-driven. This supports implementation of the Schemer architecture in a parallel processor environment.

SPACE SHUTTLE MAIN ENGINE APPLICATION

The use of Schemer to enhance the existing real-time control system for the robotic welding of Space Shuttle Main Engines has been implemented as a simulation of the existing sensor controller tasks. The AROWS-Schemer application is a prototype system for simulation of both the robot motion and the weld process based on modification of pre-programmed values by multiple sensors. The existing application incorporates trend analysis, combination of sensor values over time and of cooperating sensors, use of confidence factors, and resolution of conflicting sensor data.

As indicated in figure 5, the application simulates and further enhances the actual work cell activities. That is, it accepts as initial input pre-programmed values for both the robot motion and the weld process parameter of current. During weld execution it coalesces data from multiple sensors to modify the pre-programmed values to more exactly match the actual variations in part geometry and material.

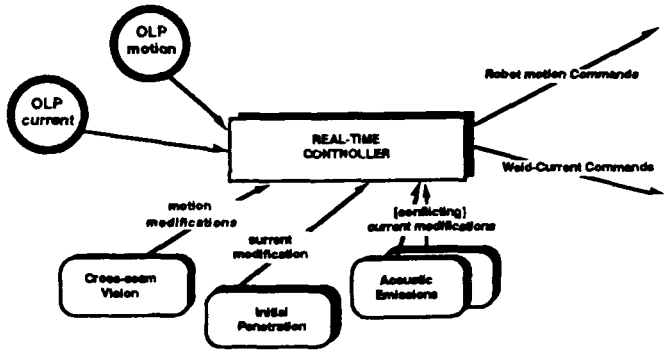


Figure 5: Pre-programmed motion and weld-process values are combined during weld execution with multiple sensor values.

As in the actual work cell, the real-time sensors are **vision** for cross-seam motion correction, and **penetration** sensors for weld-current correction [Gutow 87, Smith 87]. The Schemer simulation, however, is enhanced by the incorporation of multiple, interacting sensors. While the vision sensor is interleaved in time with the penetration sensors, there are both cooperative and conflicting penetration sensors. An **initial penetration** sensor monitors the weld until it determines that weld penetration has occurred, and modifies the pre-programmed weld-current to achieve initial weld penetration; at that time, the **acoustic emission (AE)** sensors begin penetration monitoring. Two AE sensors mounted at separate positions on the work piece generate possibly conflicting values which are inversely proportional to penetration; the pre-programmed weld-current value is modified by the AE value with the highest confidence factor.

The simulation is graphical, and uses lists of position and current values as initial input; it then responds to sensor values as they are entered asynchronously by the user (vision) or as simulated signals (penetration sensors). The output is graphical, indicating the corrected motion and current values. At this point in the feasibility study, no actual sensor data is used as input, and no formatted commands are generated for the workcell; at the time the system is implemented on a hardware platform more suited to a factory environment, these capabilities could be incorporated.

Progress Milestones

The initial goal of this project was determination of the feasibility of using Schemer to enhance the existing conventional sensor controller. The defined milestones included

- graphical simulation of a single process (motion) modified by a single sensor (vision)
- incorporation of multiple sensors modifying independent processes
  - the initial vision sensor modified the motion process

- a generic penetration sensor modified the weld-current process parameter

The two sensors did not interact in any fashion beyond co-existence.

- complex sensor interaction, including cooperating sensors, conflicting sensors, error checking, sensors affecting multiple processes

All milestones have been achieved; details of the milestone for complex sensor interaction are described in the following section. The Schemer environment was found to be appropriate for the development of this control system; it provides an environment for easy prototyping, partitioning of tasks, and response to changes in the dynamic welding process. Additional efforts include the incorporation of decision analysis techniques for quantitative model definition and use, further interaction with welding engineers for qualitative model definition, use of actual sensor-generated data, and implementation on a hardware platform suited to the factory environment.

Demonstration of Complex Sensor Interaction, Affecting Multiple Processes

The Procedural Elements defined in the AROWS-Schemer are listed in table 1. In general, each sensor is simulated by a PE, and displayed on screen by use of implementation-specific monitor/window routines.

Procedural Element	Purpose
Time-Update	generate pseudo-run-time
OLP-Update	determine next Off-Line-Programmed (OLP) motion value
Posn-Out	combine OLP and vision sensor values, send to Robot; triggered by existence of new OLP or new vision value
Black-Box	called by Posn-Out to perform geometric calculations
Current-PE	combine OLP weld-current value with offsets from penetration sensors; triggered by existence of new penetration sensor values
AE-PE	generates new values from multiple AE sensors, including confidence factor based on simple trend analysis
Initial-Pen	generates modifications for weld-current until penetration occurs

Table 1: Procedural Elements of AROWS-Schemer and their purpose

An example of a specific PE is shown in figure 6. This PE performs the function of combining the appropriate Off-Line-Programmed position value with the most recently determined vision sensor value for cross-seam correction. It initiates execution of another PE ("Black-Box") for the actual geometric calculation, and is triggered by the existence of either a new OLP value or a new vision cross-seam value.

```
;;;
;;;  POSN-OUT   generate position to send to robot, as function of OLP and sensor offset
;;;

(define-procedural-element Posn-Out

    :body (progn
          (cond ((zerop cross-seam)
                 (setf posn-correct posn-olp ) ; use olp value alone if cross-seam = 0
                (t
                 (run-pe schemer::schemer 'Black-Box)) ; else calculate geo-correction

          (setf last-posn-olp posn-olp    ; update persistent vars
                last-cross-seam cross-seam
                execution-pending nil)
    :trigger   (:level (and
                       (or     ; if either new olp
                           (unequalp posn-olp last-posn-olp)
                           (unequalp cross-seam last-cross-seam) ) ; or new vision value
                         (equalp execution-pending nil) ) ) ;and not already scheduled

     :initializer (setf execution-pending nil)  ; in support of level triggering (.vs. edge)

     :base-priority high    ; higher priority than most other PEs

     :persistent-vars ((last-posn-olp '(0 0))
                       (last-cross-seam 0)
                         (execution-pending nil)) ; initial values for vars local to this PE

     :shared-vars (cross-seam  posn-olp posn-correct)   )
```

Figure 6: An example Procedural Element, Posn-Out
which runs at high priority to insure the robot always has the most recent corrected position value.

Conflict Resolution: There are several situations where conflicts may arise between sensors. In general, similar sensors monitoring a process may provide conflicting information about that process based on sensor irregularities, or dissimilar sensors may generate conflicting data due to variations in processing algorithms and their applicability to a specific task. There are several strategies which could be used to resolve the conflict: one sensor could be designated as the primary sensor, with the secondary sensor data used only when process history indicates invalidity of the primary sensor; world models could be used, including trend analysis of sensor data as well as the specifics of the task; or sensor determined confidence factors could be compared.

In the AROWS-Schemer, two Acoustic Emission (AE) sensors are simulated, each generating independent data. The choice of correct weld-current modification must be made between the two independent values at each time slice, as time progresses under control of Time-Update. Both sensors are treated at equal priority level, and simple trend analysis is used to generated confidence factors in each set of data, as indicated by equation (1).

$$\text{confidence} = 100 - \text{abs}[\,\Delta \text{ this sensor's previous \& current values}\,] - \text{abs}[\,\Delta \text{ current value and last applied value}\,] \quad (1)$$

The AE with the higher confidence factor is then applied to the pre-programmed weld current. As calculated, the confidence factor tends to minimize the variations in the applied value of the AE sensors, i.e. the sensor value that is most consistent with both its previous value and with the last applied value is chosen. Hence, the pre-programmed values receive a higher weighting factor than "noise" in the sensor values. This is an ad hoc method of defining confidence, with no experimental basis. A mathematically based technique that will provide a more formal basis is described in the next section.

Examples of the application during execution are shown in figures 7-9, as time progresses. The initial pre-programmed values are indicated by dark circles and lines for both robot motion (a circular overlay weld) and weld-current (three pre-programmed levels are indicated). During execution, the user may generate varying vision sensor values by keyboard input; these are applied, after error checking, to the robot-motion values as cross-seam offsets. The corrected values are indicated by clear circles and dashed lines. Concurrently, at the beginning of the weld, an

Initial-Penetration sensor determines the correction to be applied to the weld-current. Penetration sensor values are combined over time before being applied as offsets to the pre-programmed weld-current values, yielding a smoothed correction to the initial values. Once weld-penetration has occurred, the AE sensors become active for monitoring penetration and modifying weld-current to maintain penetration. The AE value which is actually applied is indicated in the AE screen area by darkened circles.

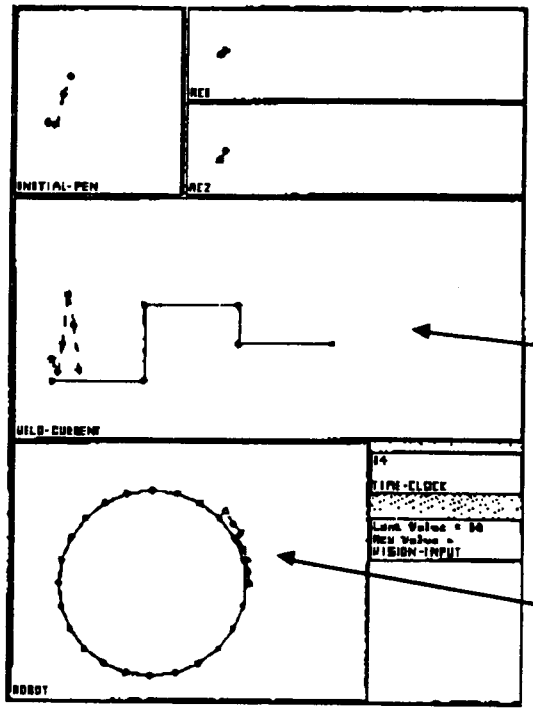
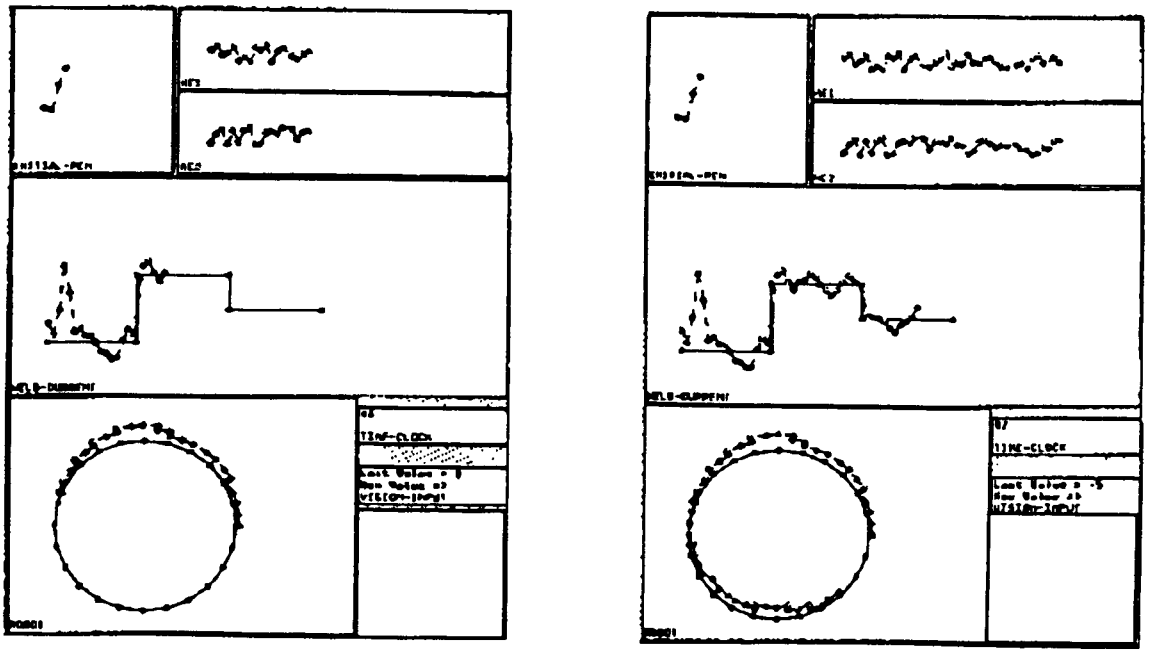


Figure 7: Simulation screen dump, at time = 14.

Initial penetration sensor has just completed control of current, and AE sensors now begin to compete.

Vision sensor has modified the pre-programmed path, as indicated by clear circles.



Figures 8,9: Additional screen dumps during execution of simulation, taken at time = 46, 87, including more acoustic emissions and vision data

Decision Analysis Techniques: Although not yet incorporated in the Schemer architecture, decision analysis methods are being applied toward this application. Influence diagrams are used to represent the model upon which decisions are based; the principles explicitly represent and respond to uncertainty and incomplete information [Breese 88]. An influence diagram has been defined for the use of the vision sensor, and probabilistic relations have been defined. At the time of this writing, no automated application of the influence diagram has been incorporated. It is expected that the influence diagram will be used both in the initial model definition, i.e. as an input to the AROWS-Schemer demonstration, and in real-time decision making within Schemer. An initial influence diagram is shown in figure 10.

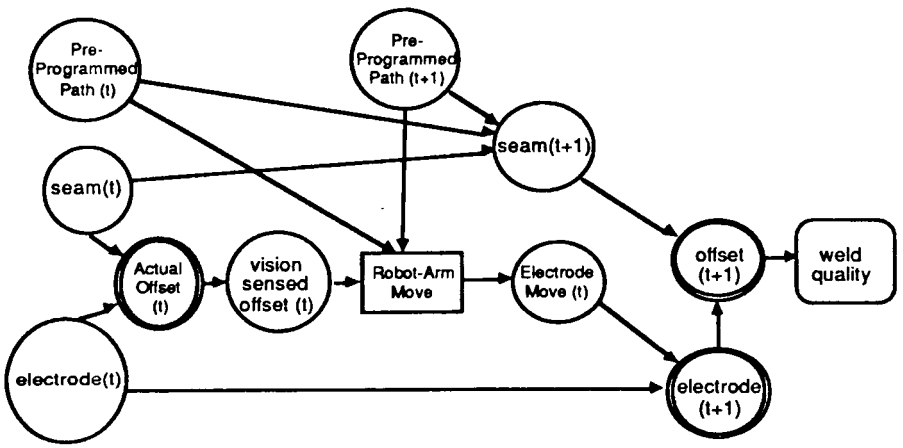


Figure 10: Influence Diagram for the vision sensor affected pre-programmed motion values

The influence diagram represents the information known at the time a decision is made, for example "Robot Arm Move" is based on both the pre-programmed values and the vision sensor value. The vision sensor value is affected by the actual offset of the electrode from the seam, as well as the quality of the vision sensor itself, in terms of the general seam being processed and with respect to a recent set of images acquired. The overall quality of the weld, with respect to position, is determined by the offset of the electrode from the actual seam. All of these relationships can be defined in a probabilistic manner, and solved for the optimum decisions, providing a mathematically based model for the control of the workcell.

Future efforts: While the existing system uses keyboard input for the vision sensor, value could be gained by using a filtered random number generator to create the vision sensor values. In this mode, additional signal processing and trend analysis could be used as the basis of geometric plan transformation, depending on the specific task and the trends detected in the sensor values. Influence diagrams are also under development for representation of the complete set of work cell activities, that is the dependence of weld quality on penetration as well as position accuracy. In this case, the detailed interaction of sensors could be simulated and a more accurate model of the decision processes could be developed. Finally, while the current implementation is on a Symbolics 3600 series system in Common Lisp, the Schemer implementation itself ports to any hardware platform supporting Common Lisp. Efforts are continuing to determine an appropriate system, and are influenced by the capabilities of existing conventional systems in the work cell. When an appropriate platform has been determined, additional efforts could include actual sensor data acquisition as well as command generation.

SUMMARY

A prototype system has been described which successfully applies a resource-bounded problem-solving architecture to simulated real-time control of robotic welding of Space Shuttle Main Engine (SSME) assemblies. The Schemer architecture supports real-time response to multiple sensor interrupts, both cooperative and conflicting. It is event-driven and responds to varying priorities in an adaptive manner; the implementation is general, and can be ported to platforms supporting Common Lisp. Schemer can be used in support of real-time control in both space-based and manufacturing environments, and provides the basis for prioritized dynamic response to changing environments, as well as the ability to incorporate mathematically based techniques from the field of decision analysis. The Schemer architecture was found to be appropriate for the development of an SSME manufacturing control system; it provides an environment for easy prototyping, partitioning of tasks, and response to changes in the dynamic welding process. The SSME application exemplifies the strengths of this knowledge-based architecture over conventional architectures, especially in the ability to respond immediately and in a prioritized manner to changes in a dynamic processing environment and the ability to apply both mathematically and heuristically based knowledge to a real-time activity.

ACKNOWLEDGEMENTS

The work described in this report was funded under NASA contract NAS8-40000 mod 130. The author is indebted to members of Rockwell Science Center, Palo Alto Laboratory: to Michael R. Fehling and B. Michael Wilber for their definition and implementation of the Schemer architecture, to Jack Breese for his direction in decision analysis techniques, to Greg Arnold for his expertise in the existing sensor controller efforts, and to Art Altman for his editing support. In addition, appreciation is extended to Rocketdyne personnel Matthew A. Smith and David Gutow for their exisiting work in acoustic emission and vision sensors, respectively; to Joseph M.F. Lee of Rockwell Science Center as Project Manager; and to Fred Schramm as NASA Contract Monitor.

REFERENCES

[Breese 88a] Breese, John S., Eric J. Horvitz, and Max Henrion, "Decision Theory In Expert Systems and Artificial Intelligence", Technical Report 3, Rockwell International, 1988

[Breese 88b] Breese, John S. and Michael R. Fehling, "Decision-Theoretic Control of Problem Solving: Principles and Architecture", Proceedings AAAI Workshop on Uncertainty in Artificial Intelligence, 1988

[Coughanowr 65] Coughanowr, D.R., Process Systems Analysis and Control, McGraw-Hill, 1965

[D'ambrosio 87] D'Ambrosio, Bruce, M.R.Fehling, S.Forrest P.Raulefs, and M. Wilber, "Real-Time Process Management for Materials Composition in Chemical Manufacturing", IEEEExpert, June 1987

[Fehling 87] Fehling, Michael R. and John S. Breese, "A Computational Model for the Decision-Theoretic Control of Problem Solving under Uncertainty", Technical Report, Rockwell International, 1988

[Guffey 86] Guffey, J., "AI takes Off: Expert Systems that are Solving In-Flight Avionics Problems", Aviation Week and Space Technology, February 1986

[Garcia 87] Garcia, Raul C., "An Expert System To Analyze High Frequency Dependent Data for the Space Shuttle Main Engine Turbopumps", Proceedings AI for Space Applications, 1987

[Gutow 87] Gutow, David, Member Technical Staff, Advanced Automation and Robotics, Rocketdyne, Private communications

[Hayes-Roth 83] Hayes-Roth, Frederick, Donald A. Waterman, Douglas B.Lenat, Building Expert Systems, Addison-Wesley, 1983

[Ruokangas 88] Ruokangas, Corinne C., B.M.Wilber, D.L.Larner, and M.C.Anderson, "Schemer: A User's Guide", Technical Note, Rockwell Palo Alto Lab,1988

[Savage 72] Savage, L.J. The Foundation of Statistics, Dover:New York, 1972

[Sliwinski 87] Sliwinski, Karen E. and Corinne C. Ruokangas "Adaptive Robotic GTA Welding for the SSME: An Integrated System", Conference Proceedings Robots 11, 1987

[Smith 87] Smith, Matthew A., Member Technical Staff, Advanced Process Instrumentation, Rocketdyne, Private communications

[Winston 87] Winston, Patrick H., "Artificial Intelligence: A Perspective", AI in the 1980's and Beyond, MIT Press, 1987