On the Development of a Reactive Sensor-Based Robotic System

Henry H. Hexmoor
William E. Underwood, Jr.

AI Atlanta, Inc.
119 East Court Square
Decatur, GA 30030
(404) 373-7515

ABSTRACT

Flexible robotic systems for the space applications need to use local information to guide their action in uncertain environments where the state of the environment and even the goals may change. They have to be tolerant of unexpected events and robust enough to carry their task to completion. Tactical goals should be modified while maintaining strategic goals. Furthermore, reactive robotic systems need to have a broader view of their environments than sensory-based systems. We offer an architecture and a theory of representation extending the basic cycles of action and perception. This scheme allows for dynamic description of the environment and determining purposive and timely action. We are exploring applications of our scheme for assembly and repair tasks using a Universal Machine Intelligence RTX robot, but the ideas are extendable to other domains. This paper describes the nature of reactivity for sensor-based robotic systems and implementation issues we have encountered in developing a prototype.

1. Introduction

Almost every system in our daily life is reactive. They asynchronously accept input and produce output. This output may be normal or abnormal according to the intended function of the system. When they are situated in an environment, reactive systems produce responses based on their intended function and they do this by changing their goals and actions in response to new recognitions in the environment or shifting situations. Similarly, reactive programs form response behaviors based on "purposes" of the system within which they are embedded in response to asynchronous input from the environment. An operating system is a reactive system, whereas a program that blindly follows a set of instructions is not. Reactive systems developed to this date are aimed at producing routine, almost reflexive, behavior in an environment. Span of cognition and perception in these systems is narrow and often does not include an examination of "mental" states such as changes in goal priorities and long range expectations. We call this sort of reaction low-level reactivity and contrast it with high-level reactivity concerned with changing goals and reactive planning. Low-level reactivity might be suitable for small scale agents that are resource rich and have relatively low impact on the overall activities. On the other hand, intelligent agents slated for space applications have limited resources and need to react not only in the local sense of the word but also in a rational manner by building or altering goals or their specifications. This is required for reasoning in a dynamic problem space.

Most everyday activities are immediate and a myopic view of the world suffices to construct models of engagement in the world with no internal representation of the world. We argue that one needs to plan across activities as well as about them and this calls for a model of an agent interacting with a changing environment capable of adapting its behavior and reasoning at various levels of abstraction to purposively react. Purposive systems perform actions in relation to their functional requirements [Kim, 1988]. Reactive behavior is needed for improving use of resources for goal achievement. This model considers directed perception and interruptible cognitive

processing in the perception-action cycle. Planning and reflective capabilities are crucial to robust reasoning agents.

Planning systems can be augmented with levels of abstraction in order to cope with combinatorics of detailed robot activitity. By carefully limiting the scope of planning to levels of abstraction, it becomes easier to identify stereotypical circumstances. A characteristic of reactive systems is hierarchies of activity. This hierarchy is a toolbox metaphor of behavioral skills. At one extreme the behavior may call for a dexterous manipulator with much sensory information, and at the other extreme it may provide a sufficing jesticulation behavior with minimal resources. This action hierarchy provides a degree of responsiveness in the environment. However, the increased overzealousness may be harmful. Central to the notion of reactivity are the two questions of when and how to react? An intelligent agent ought to react when the utility of reaction is most favorable in light of its goals. The concept of utility is used to construct a measure of desirability for courses of action. An intelligent agent also needs to decide whether to continue with a course of action or whether new circumstances are amenable to a better course of action. To quantify this, each course of action is assigned an expectation of completetion which is monitored continously and updated as new information becomes available. This is tantamount to a feasibility measure of the alternate actions. Choice of motor and sensory activities is determined by arbitration of action at execution while courses of action are discriminated by reasoning at higher cognitive levels.

The Nature of Reactivity

To clarify the different types of reactivity required of a system, we discuss various types of reactivity. We define low-level reactivity as generation of behaviors in response to signals from the environment and identify a need for high-level reactivity.

Reactivity at the low level is a response to *incremental awareness* of environmental variations and details. The prime facie principle is "do the best you can at the moment". Characteristic of low-level reactivity is a goal to be satisfied and a tactic (method) for achieving the goal. Robotic compliant motion is a type of reactivity where fine motion parameters are determined as the environmental constraints are perceived incrementally. An example is sliding along the surface of a table where the geometry of the table is not fully known and are discovered only by sensing local surface geometries. Another type of reactivity is to form responses where the environment including one's resources slowly changes. Activities of the robot can be changed to cope with small changes in the environment. Tracking an object on a conveyor belt is an example.

Low-level reactivity resembles hill-climbing algorithms and is often not sufficient to guarantee successful achievement of goals. Pure low-level reactivity may lead an agent to repetitive actions, undo actions, irreversible traps, or other undesirable situations. We suggest invocation and/or formation of goals with varying strength in response to environmental signals. Most urgent goals might be expanded to methods for accomplishing goals. These methods in turn will be presented to the low-level reactive module for execution. This method of changing goal priorities is a reactive scheme which will be dubbed high-level reactivity.

If one examines the behavior of reactive systems such as human beings, one finds that they react to crisis situations, to situations that impact preserving their good condition, property, and products of their effect, to situations conditined by their profession or role, and to situations that satisfy theur basic needs. These are all examples of high-level reactivity. Purposive robots in a dynamic environment will encounter similar situations and need mechanisms for achieving similar behavior.

2. Related Work

A number of low-level reactive systems has been developed in the last few years. Brooks [1985] describes an architecture for incorporating control mechanisms of a robot in specialized behavior units at hierarchies of increasing competence. His recent work has been directed toward building increasingly smaller agents with evolving patterns of behavior. This approach is not appropriate for space applications bacause it assumes resource rich environments rather than resource limited. Vachtsevanos and Hexmoor [1986] present a reactive approach to obstacle avoidance based on rapid replanning capabilities. Agre and Chapman [1987] advocate a model of interaction with the environment that is based on local schemas and demonstrate achievement of complex behaviors without the use of traditional planning techniques using world models. Kaebling [1987, 1988] presents specification languages REX and GAPPS that capture behaviors of agents for parallel actions and provide definition of constructs with constant bound. Agent behaviors are defined in various levels of sophisticaction which provides a hierarchy of behavioral choice based on the scope of available information. Dean and Boddy [1988] present an analysis of time-dependent planning. They introduce a class of algorithms they call "anytime" algorithms which can be suspended and resumed with negligible overhead and which will return an answer whenever terminated. The answers returned improve as a function of time in a well-behaved manner.

A number of planning issues that arise pertaining to reactive systems. For example, reactive systems can be inefficient planners. Drummond [1988] discusses how overzealous reactive systems fail. He presents a problem of stacking three blocks where the middle block cannot be put in place last. He calls this the "B Not Last" (BNL) problem. Plans are expanded in plan nets and situated control rules are provided to avoid potential traps. Monitoring and sensing the environment is also problematic. Gini [1985] describe a system that generates expectations of plans by discovering intents of plan steps. This is used to monitor execution of plans and replan when errors are fully identified. Her recent work [Gini, 1988] clearly points out that "real-time" perception is unattainable and planning based on this assumption is problematic. Planners need to keep track of reasons for plans and their formation in order to reason about and revise them. Very few teleologically adequate planning systems have been developed. An approach toward this end is consideration of mental attitudes like intention, desire, and belief. Georgeff [1987] describes such a need for reactive systems. His work served to demonstrate that intelligent goal-directed behavior in dynamic environments necessitates basing behavior generation on higher cognitive function such as intention. Our recent work in [Underwood and Hexmoor, 1989] is another step in this direction of establishing teleological foundations for planning. Our earlier work uses a schema-based hierarchical planning model. This model has been used in automatic planning of robotic fabrication and assembly tasks in airframe manufacturing [Underwood, et al, 1984 and 1988]. This approach allows modification of prior planning constructs to generate new plans.

3. A Reactive Robotic System Architecture

We propose an architecture for low-level and high-level reactivity that is illustrated in Figure 3.1. The low-level reactive apparatus is similar to previous reactive systems with essential components of perception and an action arbiter. A novelty of our architecture is incorporation of high-level reactivity and a supervisory level of planning and reasoning which guides choice of low-level reactive behaviors.

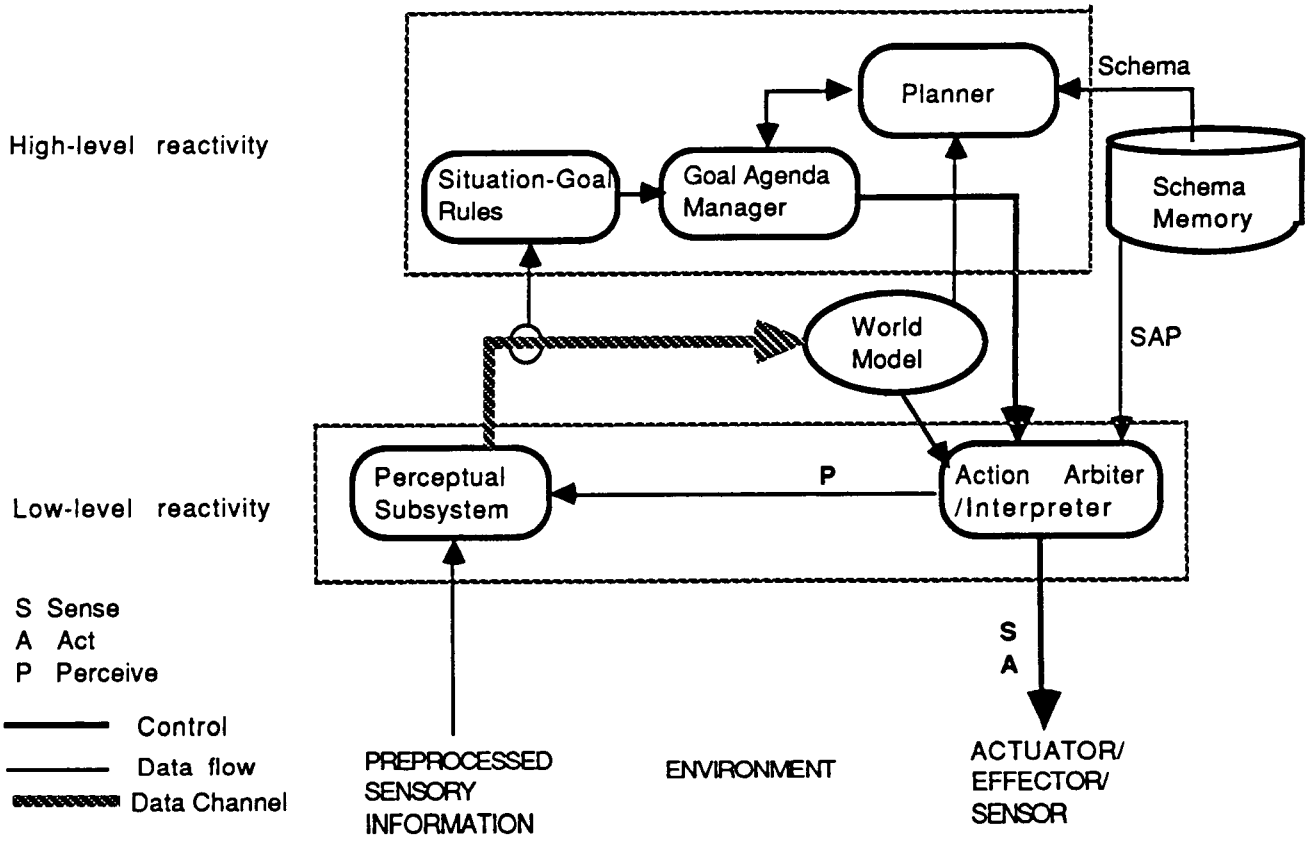

Figure 3.1 An Architecture for an Intelligent Reactive Robotic System

The world model and the schema memory are used for both low-level and high-level reactivity.

World model: This component is the database of the perceived environment and contains current information about location of parts and situation parameters along with time tags. Situations are robot activities like grasp and parameters are information used to complete the activities. The world model also contains information about the geometries of solid objects. To preserve internal consistency, only the perception subsystem adds information into the world model. The world model is consulted by the action arbiter and the planner. The world model is also intended to contain information about dynamic environments.

High-Level Reactivity

In this section we describe the objects, structure and mechanisms of high-level reactivity depicted in the upper portion of Figure 3.1.

Goals must be considered in a reactive system because they arise as a reaction to a situation, as a result of planning to achieve a goal. Schank and Abelson [1977] suggest a taxonomy of goals that are useful in automated language understanding. We previously applied this taxonomy in robotic planning and now find it useful is representing high-level robotic reactivity to situations. Particular types of goals of interest are as follows:

o Achievement goals arise from an agents role or function and have to do with the achievement of goals associated with that function.

o Preservation goals have to do with preserving or maintaining the good condition of a system, its possessions, or the results of its achievement goals. For example, P-Condition represents the goal of preserving optimal operating condition.

o Crisis goals are a special case of preservation goals that arise in response to crisis situations, e.g. fire.

o Satisfaction goals are goals corresponding to a recurring strong operational need which when satisfied are extinguished for a time, e.g. , the need for storage of electrical energy.

o Instrumental goals are any goal which facilitate realizing other goals. For example, I-Prep(part, op) represents the goal of preparing a part for an operation.

o Delta goals are distinguished types of instrumental goals that have to do with a change of state. For example, D-Prox(part, loc) represents the goal of changing the proximity of a part to location.

Situation-Goal Rules: Situation-goal rules are used to represent the goals that should be achieved in a particular circumstance. We will discuss various classes of situation-goal rules distinguished by the class of reaction they generate.

Self-Repair or Maintenance Reactions
An example of a situation goal rule in this class is: If malfunction(x), then P-Condition(x). The interpretation of this rule is that if a malfunction is perceived, then pursue the goal of preserving the optimal operating condition of x, P-Condition(x).

Threat Avoidance Reactions
An example of a rule in this class is: If some natural proceess (or other agent's actions) might cause a negative change in operating conditions, then consider the goal of blocking the natural process (or goals of the threating agent).

Another rule in this class is that for preserving possessions or other objects of value. For instance, if an agent has a tool that is useful for achieving its goals, or it has expended time and energy in accomplishing an assembly task, and it perceives a situation that threatens its possession or achievment, then it should consider the goal of preserving possessions or preserving the product of its efforts.

Reactions Conditioned by the Role of an Agent
These rules capture the robotic agent's reactions to situations in which it should respond to achieve some goal for which it was designed. These situations include requests from other agents.

Reactions based on Operational Requirements
And example of a rule in this class is: If Battery-Charge(agent) = low, then S-Energy(agent). This rule represents a reaction to the situation that an agent's battery charge is low. The agent should pursue the goal of satisfying this need for electrical energy.

Goal Agenda Manager: High-level reactivity is initiated by the triggering of situation-goal rules that are monitoring perceived situations that are being passed to the world model from the perceptual subsystem. When triggered these rules pass their associated goal to a goal agenda manager. The agenda manager determines the precedence among new goals and goals currently on the agenda. The precedence of goals is: crisis goals have precedence over satisfaction goals, satisfaction goals have precedence over achievement goals, and achievement goals have precedence over preservation

goals. Since instrumental and delta goals arise in planning for accomplishment of any of the goal types above, they inherit the precedence of their ancestor goal.

The agenda manager requests that the planner find a plan for achieving the goal at the head of the agenda. The planner returns an instantiated schema as a sequence of goals, a planbox, or a script. The agenda manager puts this instantiated schema at the head of the agenda and links it to the originating goal for this plan. If the goal at the head of the list is not an instantiated script or planbox, it is still a general subgoal, so the agenda manager will send this goal back to the planner for refinement. Instantiated scripts and planboxes are called methods. When the goal at the head of the agenda is an instantiated script or planbox, the agenda manager will send a goal/method pair, consisting of the instantiated script or planbox and its ancestor goal, to the action arbiter. The action arbiter will be discussed in the section on low-level reactivity. However, in this context, the action arbiter returns a message as to success for failure in achieving the goal. The action arbiter will continue to attempt to achieve success until given another goal. The agenda manager must determine when repeated goal failure amounts to goal blockage. In the case it decides that a goal is blocked, it creates a goal to remove the blocked goal, puts it at the head of the agenda, and requests the planner to consider this new goal.

If the situation-goal rules generate a new goal and this new goal takes precedence over a currently pursued goal, the pursued goal is temporily suspended. A problem that can occur during suspension is that completed subplans and preconditons can come undone. When the agenda manager reactivates a previously suspended subgoal, there is a need to check the current situation against prior achievments to restablish the plan structure.

The goal agenda manager also associates a priority with the goal/method pairs sent to the action arbiter. This is needed to interrupt the operations under the control of the action arbiter when a reaction is required to a goal of high precedence, for example, a crisis goal.

Schema Memory: Schema memory is a data base of all common sense schemas for use by the planner in composing networks of goals and methods necessary for determining appropriate cources of action. Schemas contained in this knowledge base are used to compose a hierarchical plan. Plans constructed from this knowledge are a specification of what to do in the perceived situation. This is unlike 'expectation-based' planners like STRIPS.

There are three types of schemas: scripts, named plans, and planboxes. A script is a plan that has become routine. Named plans are general plans that have worked previously. Planboxes are a more general type of script, intermediate between named plans and scripts.

Structure of schema memory is depicted in Figure 3.2. A-Goals and P-goals are associated with named plans, C-Goals are associated with scripts, S-Goals are associated with named plans and scripts. Named plans are in turn associated with other D-goals and I-Goals, and Scripts are associated with Sense-Act-Perceive (SAP) microcommands. SAPs are discussed in the section on low-level reactivity. D-Goals are associated with planboxes which in turn are associatedwith SAPs. The leaves of the schema structure are all SAPs.

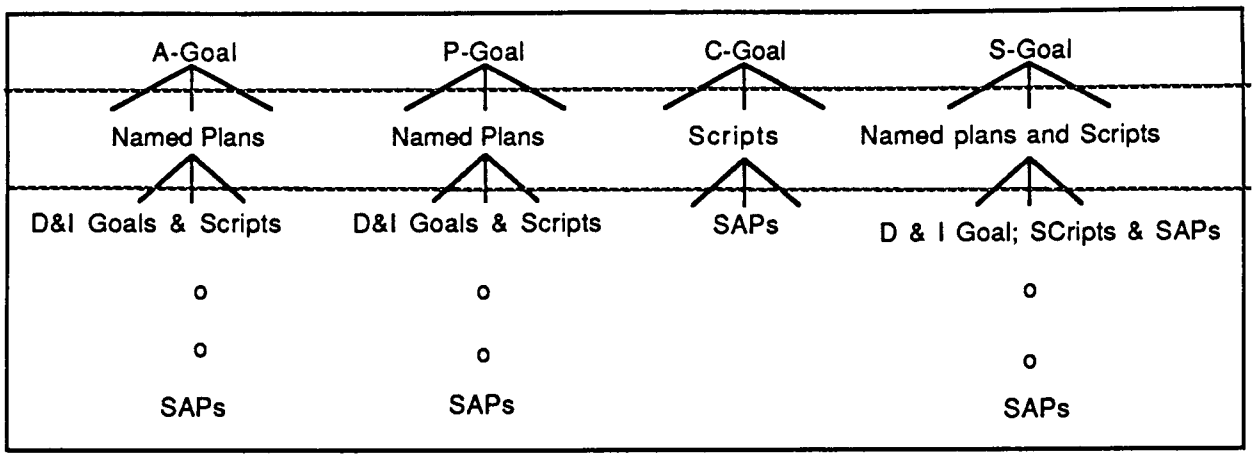

Figure 3.2 Structure of Schema Memory

Planner: The planner responds to requests from the agenda manager to find a plan for achieving a goal by searching the schema memory for relevant named plans, planboxes or scripts. It uses the world model to instantiate these schema and returns them to the agenda manager. The planner responds to a plan request with a single instantiated schema. If the first element of this schema is itself a goal, the agenda manager will request a plan for achieving that goal. Thus the plan is expanded on the agenda. At any point that the schema at the head of the agenda is an instantiated script or planbox, the agenda manager passes that method and its ancestor goal to the action arbiter, rather than requesting planning for goals further down the agenda. Thus the planning will be reactive to situations encountered during execution of the partial plans.

If the planner receives a request from the agenda manager to remove a blocked goal, it will search for an alternative named plan or planbox or it might respecify the goal by substituting a different value for a parameter of the goal. For example, if an instrumental goal is blocked, then the planner might either select a different planbox or substitute a different instrument for this goal. This captures a typical reaction of an agent who is blocked in achieving its goals. When these alternative schema for achieving a goal, a a measure of desirability based on performance can be assigned for selecting among alternatives.

Figure 3.3 illustrates a plan for assembly of two parts as it was expanded on the agenda.

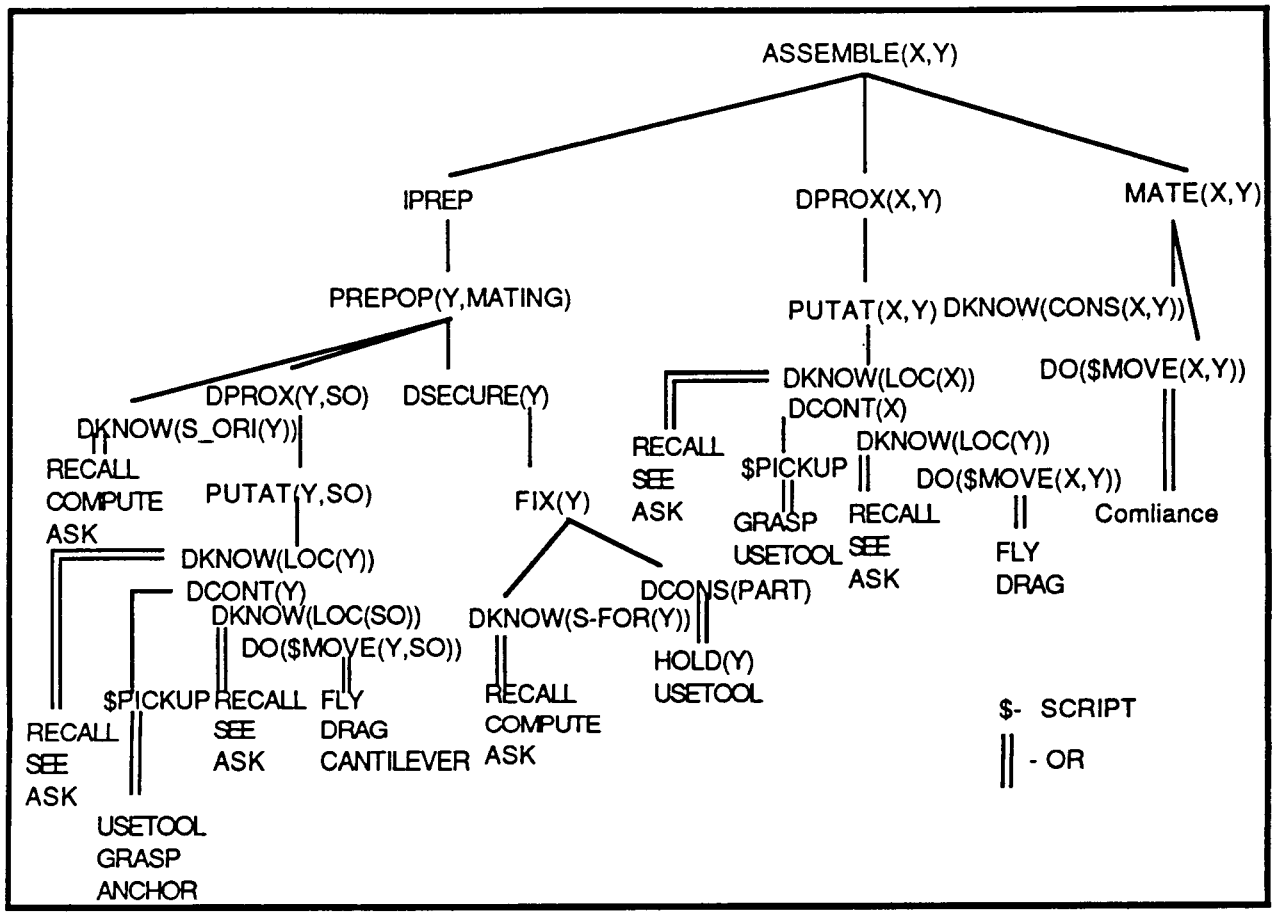

Figure 3.3 Plan for achieving the goal Assemble (X,Y)

Low-Level Reactivity

In this section we describe the structure and mechanisms for low-level reactive behaviors depicted in the lower portion of Figure 3.1.

Perception: This component is responsible for sensory perception of the current state of the environment, including the state of the robotic agent. It operates independently and continuously updates locations of parts and other information. This module must allocate time for processing all sensory input. When it receives a request for finding the location of a particular part, it time shares its computational resources to assess the requested location while continuing to process other sensory information.

Action Arbiter: This component is responsible for generating sensory and motor actions for reactive behavior. It is provided with a goal to achieve and a method for achieving it. Methods are instantiated schemas for achieving goals. It may be given a D-PROX goal and a Pickup script or a PUTAT planbox as a method for achieving D-PROX. Furthermore, methods have distinct phases of operation. Figure 3.4 depicts the structure of a method. An example of a method is a Pickup script with the following four phases: open, approach, grasp, lift.

```
Goal: <goal>
Target: <{specification of rela>
History: {executed-succeeded, executed-failed}
Method: Do
            Phase 1: <Phase 1 SAP target specification>
            Phase 2: <Phase 2 SAP target specification>
            ......
            Phase n: <Phase n SAP target specification>
        End
```

Figure 3.4 Structure of a Method

Sensory, perceptual, and motor actions are packaged into sense-act-perceive (SAP) microcommands and issued in a control loop with their parameters changing continually. Figure 3.5 shows the structure of a typical SAP. Motor commands to the robot are one of the set {Move, Open, Close, Pause, and Halt} each with several parameters. Sensory commands are one of the set {Picture, Imprint, and Force/Torque}. High-level perceptual commands are one of the set {Locate-Object, What/How-Moving}. Targets in a SAP define symbolic relations that need to hold among objects. These symbolic definitions are constructed using a functional assembly specification language we are developing [Hexmoor and Underwood, 1989]. Simple specifications are descriptions of unary or binary conditions/relations among object(s). An example of a unary SAP target is the "condition:open-wide, object:gripper", represented as openwide-gripper. A binary target is "object1:grippper condition:rightside-of object2:book", represented as gripper-rightside-of-book. Parameters for commands are computed by interpolating intermediate increments based on the current values of target specifications in the world model. Static information about object geometry may be found in the world model. These intermediate increments correspond to the resolution of sensory perception. The threshold of this resolution is a prime factor for successful interaction in our environments, natual or man made.

Each SAP contains attributes which help in determining its appropriateness to the current situation. The action arbiter chooses a set of SAPs which have similar goals and methods to the current goal/method and places them in a queue, called the SAP Exeution Queue (SAPEQ). These SAPs are independent of one another. This queue is continuosly monitored for information and resource requirements. All SAPs whose resource requirements match the currently available resources become candidates for execution and are placed in the order of their sophistication on another queue called the SAP Candidate Queue (SAPCQ). The SAP with the highest rating of sophistication is deployed for excution.

```
Method: {scripts, planboxes}
Goal: <Goal-type>
Sensory Resources: {vision, tactile,force/torque,...}
Other Resources: {tools,fixtures,parts,...}
Target: <{specification of relations}>
Sophistication: <Level>
History: {executed-succeeded, executed-failed}
Repeat
      -Using world model update parameters, abnormal termination and suucessful termination
         conditions for current sensory and motor actions for the next increment;
       -Concurrently transmit sensing commands to sensors and
                          transmit motor action commands to robot with priority and
                          transmit perceptual commands to perception module with priority;
     -Compute status of execution
Until (Status = abnormally terminated or successfully terminated)
```

Figure 3.5 Structure of a SAP

Two varieties of low-level reactivity can be differentiated based on the rate of change of the situation. First, the situation might be static or unchanged. SAP parameters are updated incrementally as the environmental contraints are perceived. Compliant motion is an example of this type of reactivity. Failure of SAPs to achieve the goals of their actions in these situations are due to limits of resolution of sensory perception and the effectors.

Senondly, the situation might change slowly enough that the situations are well within direct sensory perception thresholds, so that adjustments of parameters and termination conditions can be computed. Reactivity in these situations amounts to goal refinement. Each SAP is given the capability of incrementally adjusting the actions of all actuators, sensors and the perceptual module. An example of this variety of reactivity is tracking an object on a moving conveyor belt. Some refer to this variety of low-level reactivity as adaptivity.

The situation might change so rapidly that the situations are below the resolution of sensory perception. Adjustments to parameters in these situations might not be adequate to achieve the goal, so that new goals might have to be substituted for old ones. In these cases new SAPs will be selected.

Upon termination of a SAP, status of the SAP is examined in light of the overriding goal, the method and the changes in the environment. This status is one of the set {terminated normally, terminated abnormally}. In the special case of a static world, a retrial of the same SAP is issued, and in case of repeated failure, supervisory levels of control will intervene. Otherwise, depending on how much the world has changed, SAP queues are reprocessed. Figure 3.6 summarizes these relations in a decision matrix.

	SAP succeeded	SAP failed
Situation Unchanged	Pick the next Goal/Method*	React1: try again
Situation Changed Slowly	Pick the next Goal/Method **	React2: Update SAPCQ and retry
Situation Chaged Rapidly	Pick the next Goal/Method ***	React2: Update SAPCQ and retry

* No reaction was necessary
** Reacted by changing SAP parameters in-flight
*** Probably unrelated changes
React1:First variety of reaction
React2:Second varity of reaction

Figure 3.6 Action Arbiter Decision Matrix

Figure 3.7 summarizes the activities of the action arbiter. Note that with the availability of most information and resources, execution proceeds with the most certain scheme of accomplishing the task transmitted to the robot. We refer to this as executing at level 1. Otherwise, conditions are tested for a less certain version of accomplishing the same task.

```
Repeat
    -Select the next goal/method
    -From schemas select SAPs with similar goals and methods to the current goal/method and
     put them on the execution queue (SAPEQ)
   Repeat
        -From the execution queue select SAPs requiring resources similar to currently available
         resources and put them on SAPCQ in the order of their sophistication
      -Execute the SAP with highest sophistication from the SAPCQ which
         may not have failed more than once
        -Remove SAP from SAPEQ if SAP failed twice
    Until (Empty SAPCQ or Empty SAPEQ or SAP is terminated successfully)
   -Empty SAPEQ
    -Generate status report and send to agenda manager
Until (no goal/method)
```

Figure 3.7 Action Arbiter

The action arbiter may receive interrupts from higher levels with three levels of priority. The lowest priority interrupt will cause a pause in the motor actions. The next higher priority interrupt may direct the action arbiter to a different goal/method queue. The highest priority interrupts may be sent through the arbiter to the robot to stop an overzealous low-level reactivity. To handle high priority interrupts a service routine is often necessary. For example, a high priority interrupt might activate a service routine to grip an object more tightly to prevent it from slipping from the gripper while the object is being moved. Service routines are types of SAPs and there is a supply of service routines which are scheduled for execution and are transmitted to the robot with the highest priority to redirect the activities of all actuators and effectors as well as sensors and the perceptual module.

4. Implementation Issues

We are using a six degree of freedom Universal Machine Intelligence RTX robot in prototyping this conceptual design. We are adding sensory capabilities of force/torque sensing at the wrist, a vision system with a camera looking down on the workspace, and a tactile sensor at the inside of the gripper jaws.

The task of spatial reasoning in the perceptual module is by far the most difficult to implement for real-time reactivity. Vision and tactile sensory preprocessors generate grids of data for intepretation. Each image needs to be understood individually and correlated with other sensory data for recognition of situations. One approach to implementation is to use a hierarchical perceptual processing system, much like a blackboard model of problem solving, where specialized processing modules intepret data and form perceptual hypotheses for higher perceptual modules. At the topmost level patterns of perception trigger percept schemas that update the world model along with a time stamp. Since a reactive system requires efficient perception, one might opt for additional processing power, but in resource poor space applications this might not be a feasible alternative. Directed perception and improvements in perceptual algorithms are a more likely implementation alternatives in resource poor applications.

Implementation of high-level reactivity requires a flexible control framework. This might also be achieved through a blackboard architecture with a control blackboard [Hayes-Roth 1988, Underwood, et al 1988]. However, more efficient implementations of this framework will be necessary to support the real-time requirements of reactivity.

A SAP can be considered as a feedback loop where commands are generated based on the previous state of the world. Figure 4.1 show a SAP's feedback loop. In each cycle perception commands

are sent to the perceptual module, sensory commands to the sensors, and motor action commands to the robot controller. The world model is updated and new parameters are computed. As argued by Gini [1988], it is not realistic to assume rapid perception. With the advent of smart sensors, it is possible to delegate elementary monitoring tasks to the sensors. This enables inclusion of an embedded feedback loop within the sensor with faster sampling rates in the order of milliseconds versus seconds for the outer SAP feedback loop.

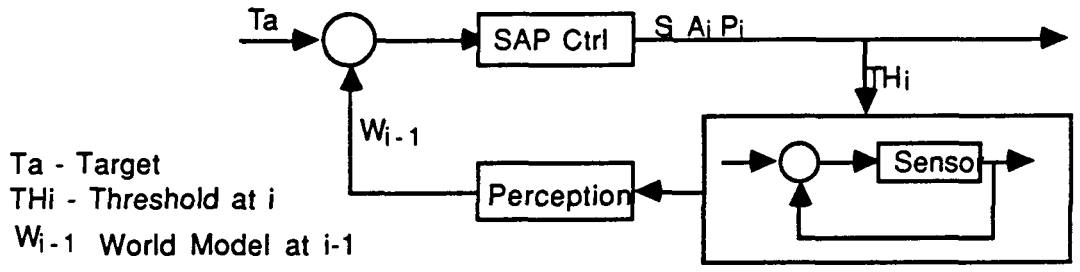

Figure 4.1 SAP as a Feedback Loop

The sampling rate of SAPs are adjusted for types of motion. Gross motions need lower sampling rates, versus fine motions where more rapid samplng os required.

5. Summary and Conclusion

Autonomous systems must react and adapt to situations in their environment. Two levels of reactivity should be distringuished. High-level reactivity to situations is at the conceptual level. Low-level reactivity is at the sensory, perceptual and motor level. Whereas other considerations of reactive systems have addressed the latter type of reactivity, we propose an architecture that can realize both levels.

High-level reactivity can be realized by a continuously active perceptual system, situation-goal rules that are triggered by perceived situations, an agenda manager that schedules consideration of new goals in the context of current and pending goals, a schema-based planner that suggests appropriate patterns of behaviors for achieving goals, and by capabilities for low-level reactivity.

Low-level reactivity adopts the principle of "do the best you can at the moment" and is tantamount to either adapting a motor action or substituting motor actions at different levels of competence to accomplish a given tactical goal. Motor acts are adapted to a situation by specifying symbolic targets for motor acts and adjusting their parameters including their termination conditions. In a changing environment, methods for accomplishing tasks can change in appropriateness and the availability of different means of accomplishing the same task affords a higher degree of reactivity.

A significant reaserch direction is to integrate reactive planning with learning capabilities that are cognizant of effects of action over time. These systems might learn patterns of failure and success and generalize plans. An extension to reactive systems is to consider multi-agent reactive systems and have them reason about other agents instantaneous behaviors.

Automation of assembly and repair tasks is difficult and abundant uncertainties indicate flexibility in adjusting to the environment is necessary. Robotic assembly in space will require robust systems that can react to situations. Although much of the discussion in this paper has addressed the domain of sensor-based robotic assembly, the architecture and techniques developed are applicable to mobile robots, such as the mars rover, and many other space systems that could benefit from a higher degree of autonomy .

References

Agre, P.E. and Chapman, D. (1987), "Pengi: An Implementation of a Theory of Activity", In *Proceedings of AAAI 87*, pp. 286-272.

Brooks, R. (1985), "A Robust Layered Control System for a Mobile Robot", A.I. Memo 864, MIT AI laboratory.

Dean, T. and Boddy, M. (1988), "An Analysis of Time Dependent Planning", In *Proceedings of AAAI 88*, pp. 49-54.

Drummond, M. (1988), "Situated Control Rules", In the *Proceedings of the Rochester Planning Workshop.*

Georgeff, M., Lansky, A.L., and Schoppers, M.J. (1987), Reasoning and p[lanning in Dymamic Domains: An Experiment with a Mobile Robot", TN 380, SRI International.

Gini, at al, "Symbolic Reasoning as a Basis for Automatic Error Recovery in Robots", TR 85-24, Computer Science Department, University of Minnesota.

Gini, M. "Automatic Error Detection and Recovery", TR 88-48, Computer Science Department, University of Minnesota.

Hayes-Roth, B. (1988), "Making Intelligent Systems Adaptive", STAN-CS-88-1226, Department of Computer Science, Stanford University.

Hexmoor, H. and Underwood, W. (1989), "An Ontology and Repersentation for Flexible Assembly", In preparation.

Kaebling, L.P. (1987), "An Architecture for Intelligent Reactive Systems", In Michael P. Georgeff and Amy L. Lansky, editors, **Reasoning about Actions and Plans**, pp. 395-410, Morgan Kaufman Pubishers.

__, (1988), "Goals as Parallel Specifications", In *Procedings of AAAI 88*, pp. 60-65.

Kim, S. (1988), "Information Framework for Robot Design", In International Encylopedia of Robotics, Richard C. Dorf, editor, Wiley and Son Publishers, NY.

Liu, Yanxi andArbib, M.A. (1986), "A Robot Planner in the Assembly Domain", COINS TR 86-36, University of Mass., Amherst, MA.

Schank, R. and Abelson, R. (1977), "Scripts, Plans, Goals and Understanding", Lawrence Erlbaum Associates , Inc., Publishers, Hillsdale, NJ.

Underwood, W. and Hexmoor, H. (1989), "Intentional Conepts in Industrial Management and Engineering", Submitted to The Second International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Tullahoma, Tennessee.

Underwood, W., MacGreggor, B.K., and Scruggs, R.M.(1984), "Scripts, Plans, Goals and Robot Planning", *Focus on Manufacturing Technology Research*, Georgia Institute of Technology, March 1984.

Underwood, W., Hexmoor, H., and Scruggs, R.M. (1988), "Intelligent Task Planning and Execution for Robotic Control", Technical Report AIAI-88/01, Artificial Intelligence Atlanta, Inc., Decatur, Georgia.

Vachtevanos, G., Hexmoor, H., "A Fuzzy Logic Approach to Robotic Path Planning with Obstacle Avoidance", *Proceedings of the International Conference on Control, Decision*, Athens, Greece, 1986.