International Journal of Intelligent Control and Systems, Vol. 1, No. 1 (1996) 15–30
© World Scientific Publishing Company

A REFERENCE MODEL ARCHITECTURE FOR DESIGN AND IMPLEMENTATION OF INTELLIGENT CONTROL IN LARGE AND COMPLEX SYSTEMS

JAMES S. ALBUS
Chief, Intelligent Systems Division
National Institute of Standards and Technology
Building 220 Room B124, Gaithersburg MD 20899-0001, USA

ALEXANDER M. MEYSTEL
Senior Scientist, Intelligent Systems Division
National Institute of Standards and Technology
Building 220 Room B124, Gaithersburg MD 20899-0001, USA

RCS (Real-time Control System) is a reference model architecture for intelligent systems that represents an interrelated set of semiotic principles typical of natural and artificial intelligence. This architecture allows both for interpreting sensory phenomena as well as for controlling complex technological systems and processes. It consists of a hierarchically layered set of processing nodes. At each layer, entities are recognized, tasks are deliberatively planned, and feedback from sensors closes a reactive control loop. RCS thus integrates and distributes deliberative and reactive functions throughout the entire hierarchical architecture, at many different temporal and spatial scales. It is demonstrated that the system of representation leads to nested knowledge structures which can be used for design and implementation of intelligent controllers for a wide variety of complex systems.

Keywords: Real-time, intelligent control, hierarchical control, reference model architecture, deliberation, and reactive control.

1. Introduction

In order to be considered as a reference model for the study and design of intelligent systems, an architecture should have the following properties:

(1) The architecture should provide a conceptual framework for understanding both natural and artificial intelligence, and should suggest an engineering methodology for designing large scale intelligent systems and processes.
(2) The architecture should demonstrate how to combine and blend both deliberative and reactive behaviors in a single integrated architecture. It should incorporate deliberative mechanisms that can reason about the past and generate plans and strategies for avoiding danger and achieving desirable future goals. It should also contain reflexive mechanisms that can evoke immediate reactions to sensed conditions.
(3) The architecture should describe how to represent knowledge about the world in both long-term and short-term memory, in both iconic and descriptive forms. It should explain how to transform information from one form to the other. It should deal with the abundance of information that needs to be processed, stored, updated, and retrieved. It should show how stored knowledge can be used to predict the results of tentative plans, generate expectations of sensory

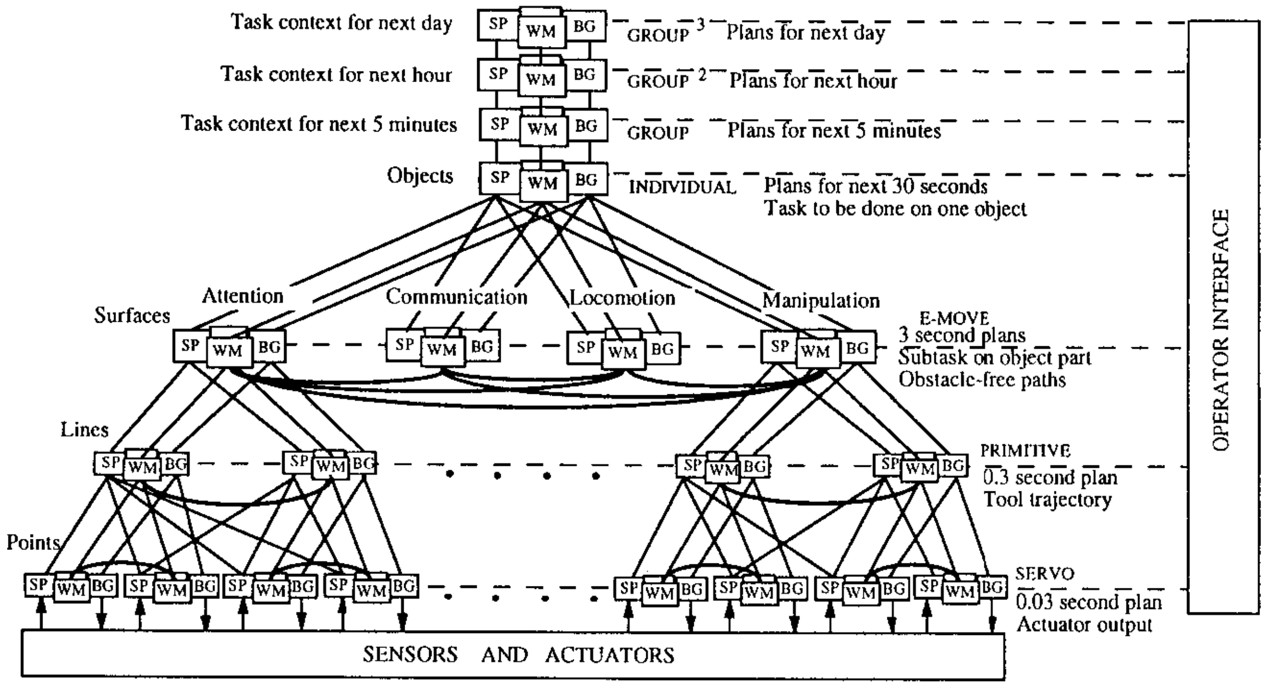


Fig. 1. A RCS reference model architecture for intelligent systems. Processing nodes are organized such that the BG modules form a command tree. Information in the knowledge database is shared between WM modules in nodes within the same subtree. On the right, are examples of the functional characteristics of the BG modules at each level. On the left, are examples of the type of entities recognized by the SP modules and stored by the WM in the knowledge database at each level. Sensory data paths flowing up the hierarchy typically form a graph, not a tree.

input, and respond appropriately to unexpected events.

(4) The architecture should describe how to process signals from sensors into knowledge of situations and relationships, and how to store such knowledge in representational forms that can support reasoning and decision making.

2. The RCS Paradigm

The Real-time Control System (RCS) developed at the National Institute of Standards and Technology and elsewhere over the past two decades possesses the above properties[1-4] RCS is a reference model architecture for the study of both natural and artificial intelligence. It provides a model for bridging the gap between the deliberative and the reactive control. It suggests how to represent knowledge in a variety of forms, and how to process sensory information into a form useful for reasoning and decision making. RCS has been used in the design of several large scale intelligent machine systems. A number of these are described near the end of this paper.[a]

The RCS reference model architecture consists of a hierarchically layered set of processing nodes connected together by a network of communications pathways as shown in Fig. 1. At each layer of the RCS hierarchy, there are both deliberative and reflexive elements. At each level, sensory data are processed, entities are recognized, world model representations are maintained, and tasks are deliberatively decomposed into parallel and sequential subtasks, to be performed by cooperating sets of subordinate agents. Also at each level, feedback from sensors reflexively closes a control loop allowing each agent to respond and react to unexpected events. The result

[a]The RCS reference model architecture has evolved over the past 20 years, and has been implemented in many different versions. The RCS described in this paper is the most recent version, and contains a number of advanced concepts and features that have not yet been implemented.

is a system that combines and distributes deliberative and reflexive features throughout the entire hierarchical architecture, with both planned and reactive capabilities tightly integrated at all levels and time frames.

Each node in the RCS architecture can be constructed from four basic types of processing modules — Behavior Generating (BG), World Modeling (WM), Sensory Processing (SP), and Value Judgment (VJ) — plus a Knowledge Database (KD) module. The nodes are interconnected in a system architecture that communicates information between and within the nodes.

3. Behavior Generating (BG) Modules

BG modules contain Job Assigner (JA), Scheduler (SC), Plan Selector (PS), and Executor (EX) functions (or submodules) as shown in Fig. 2.

The **Job Assigner** accepts input task commands from an Executor (or Agent) in a higher level BG module. It also has access to the current plan of the higher level BG module. The Job Assigner outputs job assignments to a set of Schedulers within the BG module.

The **Schedulers**, one for each agent in the BG module, accepts job assignments from the Job Assigner. The set of schedulers may coordinate their activities to generate a coordinated plan for the agents.

The **Plan Selector** sends tentative plans to a World Model simulator, and receives back evaluations from a Value Judgment plan evaluator. (The WM simulator and VJ evaluator are not shown in Fig. 2.) Based on this information it either selects the best of the plans generated up to that time by the Job Assigner and Schedulers, or requests further planning.

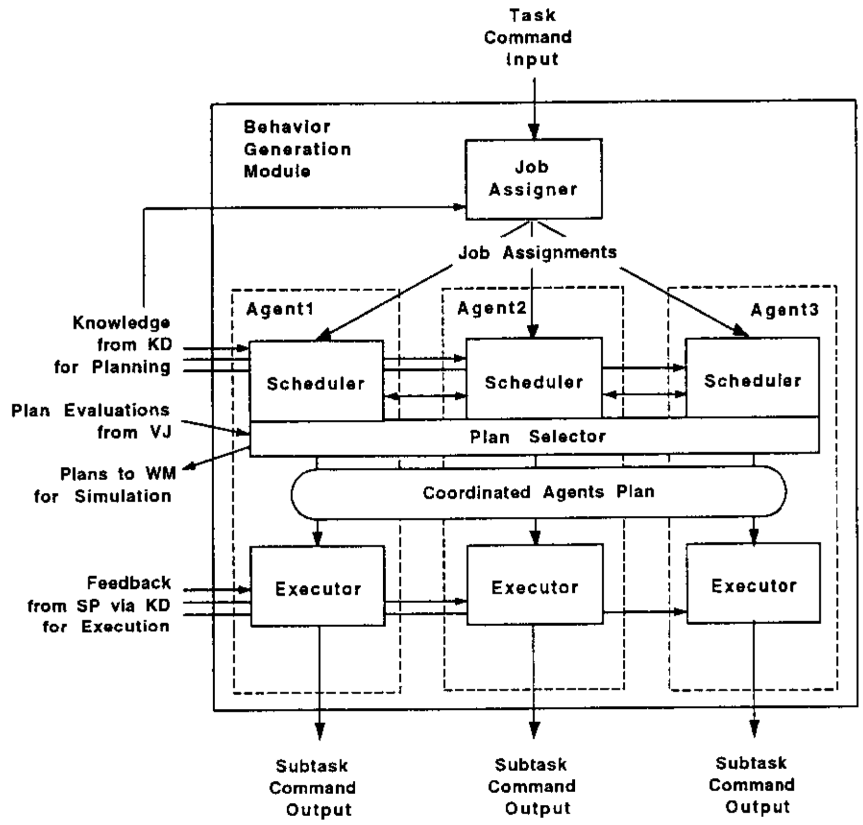


Fig. 2. A Behavior Generation (BG) module showing a Job Assigner that allocates jobs and resources to agents, and three agents, each consisting of a Scheduler and an Executor. A plan selector selects the best of several alternative plans generated by the Job Assigner and Schedulers for execution by the Executors. The Executors with their supporting Schedulers, World Modeling, Knowledge Database, and Sensory Perception modules comprise agents.

The **Executors** execute the selected plan, coordinating actions between agents (when required), and correcting for errors between planned and observed states reported by the world model. Outputs from the Executors become input task commands to subordinate BG modules.

4. Planning

BG modules can accommodate a variety of planning algorithms. These can range from simple table look-up of pre-computed plans or scripts, to real-time search of configuration space, or game theoretic or operations research algorithms for multiagent groups. Regardless of how plans are synthesized, a plan consists of a spatial decomposition of a task into a set of job assignments and resource allocations to agents, plus a schedule of subtasks ordered along the time line for each agent. In many cases, it is required that the agents' schedules be coordinated so as to produce coordinated actions.

In general, planning consists of the following steps:

- Generating a set of tentative plans.
- Simulating the likely results of those plans.
- Evaluating those results according to some cost/benefit criterion.
- Selecting the tentative plan with the best evaluation for execution.

The evolution of the generic planning process is shown in Fig. 3. JA submodule distributes jobs and resources to agents, and transforms coordinate systems from task to subtask coordinates (e.g. from end-point or tool coordinates to joint actuator coordinates). The SC submodules compute a temporal schedule of subtasks for each agent and coordinate schedules between cooperating agents (e.g. joint actuator trajectories are coordinated to generate desired end-point trajectories). Together, the assignment of jobs and resources to agents, the transformation of coordinates, and the development of a (possibly coordinated) schedule for each agent, constitutes the synthesis of a plan. Therefore, output from the JA and SC submodules is an alternative plan.

Each alternative plan is submitted to the WM module for simulation of predicted results. The predicted results are evaluated by the VJ module for cost and benefit of predicted results. The Plan selector then either selects the plan for execution, or requests another alternative plan be generated. Alternatives may involve alternative job assignments or alternative scheduling of jobs.

In highly structured and predictable environments, plans may be computed off-line, long before execution. For example, in manufacturing plants, shop level planning is often done off-line in a batch mode computing environment, once a day, or once a week. However, this often produces plans that become obsolete shortly after execution begins. As the uncertainty in the environment increases, plans need to be computed nearer to the time when they will be executed, and be recomputed as execution proceeds in order to address unexpected events.

RCS can accommodate either pre-computed plans, or planning functions that recompute plans on demand, or on repetitive cyclical intervals. The repetition rate of a real-time planning loop must be at least such that a new plan is generated at each level before the corresponding executor finishes the old plan. In highly uncertain environments, the planner should generate a new plan nearly as fast as the executor completes each step in the plan.

At each RCS level, the schedulers compute plans out to a given planning horizon. The length of the planning horizon is a distinguishing characteristic of a RCS level. The resolution of plans is such that at each level, plans contain on the order of ten sequential subtasks for each agent. Thus, planning horizons shrink and temporal resolution of subtasks increases about an order of magnitude at each lower level, while the number of subtasks in each plan remains constant. Planning horizons at high levels may span months or years, while the planning horizon at the bottom level may be 30 milliseconds or less. The number of levels required in a RCS hierarchy is approximately equal to the logarithm of the ratio between the planning horizons at the highest and lowest levels.

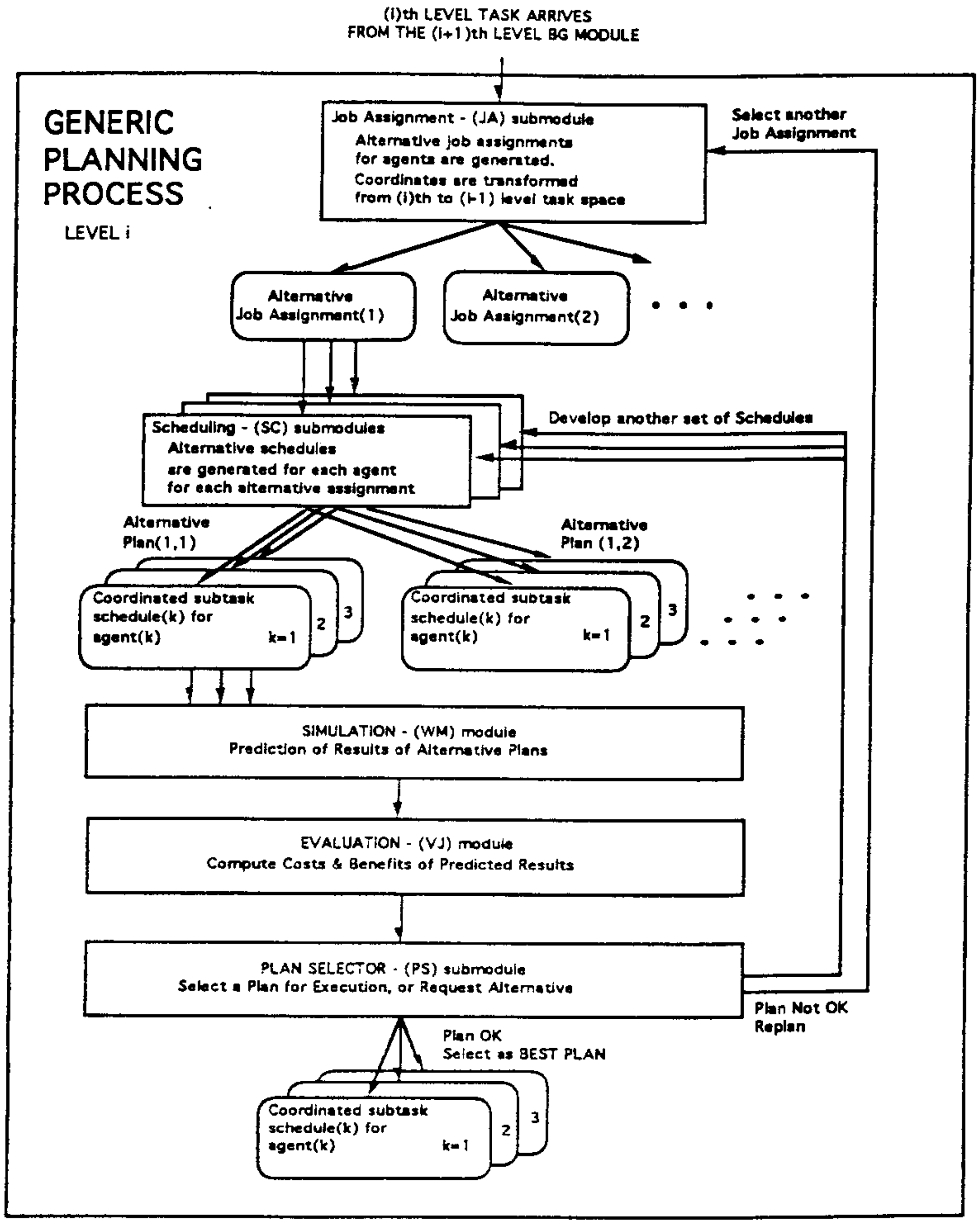

(i)th LEVEL TASK ARRIVES
FROM THE (i+1)th LEVEL BG MODULE
GENERIC PLANNING PROCESS
LEVEL i
Job Assignment - (JA) submodule
Alternative job assignments for agents are generated.
Coordinates are transformed from (i)th to (i-1) level task space
Select another Job Assignment
Alternative Job Assignment(1)
Alternative Job Assignment(2)
Scheduling - (SC) submodules
Alternative schedules are generated for each agent for each alternative assignment
Develop another set of Schedules
Alternative Plan(1,1)
Alternative Plan (1,2)
Coordinated subtask schedule(k) for agent(k) k=1
2
3
SIMULATION - (WM) module
Prediction of Results of Alternative Plans
EVALUATION - (VJ) module
Compute Costs & Benefits of Predicted Results
PLAN SELECTOR - (PS) submodule
Select a Plan for Execution, or Request Alternative
Plan Not OK Replan
Plan OK
Select as BEST PLAN

Fig. 3. Evolution of the generic planning process. The JA function generates alternative assignments for agents. The SC function generates schedules for each agent. The resulting tentative plan is submitted to the WM simulator which predicts a result. The VJ module computes a cost-benefit analysis on that result. The PS then either selects the plan to be executed, or requests additional planning. By iteration through this planning loop, the space of possible plans can be searched, with the PS function selecting for execution the plan receiving the best VJ evaluation.

5. Execution

A plan represents a path, or set of reference trajectories, from the current state to a desired goal state expressed in a vocabulary of task commands that can be accepted as inputs by the Executor (EX) submodules. For each agent at each level, there is an executor that executes its part of the plan. The EX function compares the desired subgoal of its current plan subtask with the current estimated state of the world from the World Model. The EX function then executes a control law designed to reduce the difference between its subgoal and the state of the world. Each EX module thus functions as a closed loop servomechanism, steering its agent to follow the reference trajectory which is the plan.

The loop bandwidth of the control loop at each level is defined by the repetition rate of the EX function at that level. This repetition rate should be an order of magnitude faster than the average step in the plan is completed, and hence about one hundred times the reciprocal of the planning horizon at each level.

Output from each EX submodule becomes an input command to a BG module at the next lower level. At the lowest level, the output from each EX goes to an actuator. At all other levels, the output goes to the Job Assignment submodule in the BG module at the next lower level.

6. Integration of Deliberative and Reactive

The closed feedback loops through the EX functions provide reactive, or reflexive, responses that generate sensory interactive behavior. The planning functions that take place in the job assignment, scheduling, WM simulation, VJ evaluations, and plan selection modules provide deliberative behavior. The RCS architecture thus mixes reactive and deliberative elements in BG modules at each level of architecture. At lower levels of the RCS hierarchy, the planning elements are relatively simple, and the reflexive execution elements predominate. At higher levels, the reverse is true — deliberative planning elements consume most the computing resources, and reflexive execution elements decrease in relative computational demands.

7. World Modeling (WM) Modules

The WM modules perform four basic functions:

(i) WM modules maintain the knowledge database, keeping it current and consistent. They update state estimates in the Knowledge Database (KD) based on correlations and variance between world model predictions and sensory observations at each node. WM update functions may include recursive estimation algorithms, or processes that compute lists of attributes from images, as well as recognition and detection algorithms that perform pattern matching operations necessary to verify the identification of features, surfaces, objects, and groups. Both iconic and symbolic representations are maintained. Updating symbolic representations requires a transformation from iconic to symbolic representations. WM functions enter new entities into the knowledge database and maintain the links between symbolic data structures that define relationships between entities.

(ii) WM modules generate predictions of expected sensory observations that enable Sensory Processing (SP) modules to perform correlation and predictive filtering. They use symbolic representations to generate iconic images, masks, and windows that can support visualization, attention, and model matching.

(iii) WM functions respond to queries from the BG modules regarding the state of the world or the state of the controller. They act as question answering systems, and transform information into the coordinate system required by the task.

(iv) WM functions perform simulations necessary to support the planning requirements of the BG modules. This requires dynamic models to generate expectations, and predict the results of current and future actions. Results predicted by the WM simulations are sent to the VJ modules for evaluations, which are returned to the BG module for plan selection.

8. Sensory Processing (SP) Modules

SP modules process data from visual, auditory, tactile, proprioceptive, force, velocity, position, acceleration, temperature, magnetic, radiation, taste, or smell sensors. SP modules contain filtering, masking, differencing, correlation, matching, and recursive estimation algorithms, as well as feature detection, clustering, and pattern recognition algorithms. Interactions between WM and SP modules can generate a variety of filtering and detection processes such as Kalman filtering and recursive estimation, Fourier transforms, and phase-lock loops. In the vision system, SP modules process images to detect brightness, color, and range discontinuities, optical flow, and stereo disparity. They may utilize a variety of signal detection and pattern recognition algorithms to analyze scenes and compute information needed for manipulation, locomotion, communication, attention tracking, and spatial-temporal reasoning.

9. Value Judgment (VJ) Modules

VJ modules contain algorithms for computing cost, risk, and benefit, for evaluating states and situations, for estimating the reliability of state estimations, and for assigning cost-benefit values to objects and events. VJ modules may compute Bayesian and Dempster-Schafer statistics on information about the world based on the correlation and variance between observations and predictions in order to assign confidence values to data from various sources.

10. Knowledge Database (KD) Modules

KD modules store the data that support the BG, WM, SP, and VJ processing modules in each node. KD modules store information about the world in the form of state variables, entity frames, event frames, rules and equations, and images or maps.

(1) **State variables** represent the current estimated state of the world.
(2) **Entity frames** are list data structures that store symbolic representations of features, objects, or groups that exist in the world, or in the imagination. An entity frame consists of a list head with a name as an address, plus a set of attribute-value pairs, and a set of relations to other entities or events. These relationships represent semantic meaning.
(3) **Event frames** are list data structures that store symbolic representations of state transitions, or situations at particular times and places, or sequences of states or situations that transpire over intervals of time and space in the world. An event frame also consists of a name, a set of attribute-value pairs, and a set of relationships to other events or entities.
(4) **Rules and equations** such as if/then rules, the predicate calculus, and differential equations can express physical laws that describe the way the world works, as well as mathematical and logical formulae that describe the way things relate to each other. Control laws and plant models can also be represented in the knowledge database.
(5) **Images** are two-dimensional arrays of attribute values. Images may be generated in a number of ways. For example, an image may be formed by the optical projection of light from a scene in the world through a lens onto an array of photoreceptors (or pixels) such as the retina or a CCD TV camera. An image may also be formed by pressure on an array of tactile sensors on the skin. An image consists of attributes such as brightness, pressure, spatial or temporal gradients, stereo disparity, or computed values such as range, or flow that are derived from other images. An image may also be generated by internal mechanisms (such as a computer graphics engine) from information stored in symbolic entity frames. In biological systems, this process corresponds to imagination.
(6) **Maps** are also two-dimensional arrays of pixels, wherein icons or symbolic names, in addition to attributes, are attached to pixels.

The KD contains both short term (dynamic) and long term (static) memory elements.

(1) **Short term memory** consists of both symbolic and iconic representations:
    (a) Short term symbolic entity frames represent current entities-of-attention that have either been specified by the current task, or are particularly noteworthy entities observed in current sensory input. Short term entity frames include attributes, pointers to iconic images, and pointers to entities stored in long term memory.
    (b) Short term iconic representations can consist of attribute images generated directly from sensory observations, or by recursive estimation. Short term iconic images can also be generated by internal mechanisms

from short term entity frames. In machine systems, this is done through simulation/animation. In biological systems, it corresponds to imagination. Short term iconic images can be used to mask or window incoming data, or to fuse incoming sensory observations with internally generated images. Short term iconic images persist in memory only so long as they are refreshed by incoming sensory data or by internally generated images.

(2) **Long term memory** contains the entire dictionary of entities that the intelligent system knows about. It consists entirely of symbolic entity and event frames, plus rules and equations. Attributes from long term frame representations may be transferred into short term memory, or vice versa. If a long term symbolic entity is specified by a task command, attributes of the long term memory entity frame can be added to the attribute list of the short term entity-of-attention frame. Alteınatively, if a match is recognized between a current entity-of-attention and a long term entity, newly observed attributes from the short term entity can be used to update the attributes of the long term entity, and attributes of the long term memory symbolic entity can be added to the short term entity. If nothing in long term memory is recognized as corresponding to what is observed, and if the observed entity is judged noteworthy by the Value Judgment function, the short term symbolic entity will be entered as a new entity into long term memory.

The KD is implemented in a distributed fashion, with representations at each node defined by the requirements of the BG and SP functions being carried out in each node.

11. A System Architecture

The system architecture defines the relationships between the functional modules and the knowledge database, and provides a communication system that transmits messages between them. The communications system conveys commands from a BG modules to their subordinates, and returns status. It conveys tentative plans from the BG planners to the WM simulators. It transmits simualtion results to the VJ evaluators, and returns plan evaluations to the BG Plan Selectors. It moves sensory data from sensors to filters and transfers WM predictions to SP comparators. It applies windows to SP spatial integrators and thresholds to SP detectors and recognizers. It communicates the names of recognized entities to the WM knowledge database and conveys correlations and variance to WM update mechanisms. It communicates reward and punishment data to the VJ modules, and communicates evaluations to wherever they are needed.

The various processing and data modules in the RCS architecture act as a collection of intelligent agents (or software objects), sending and receiving messages to and from each other. These messages convey commands and requests, and return status. RCS does not specify the communication mechanism. Messages may actually be communicated by point to point message passing, network broadcast, or shared common memory. In biological systems, messages are conveyed by impulses on neural axons, or via hormones through the blood stream. The RCS reference model requires only that any particular state variable have a single functional source, or writer, but it may have many destinations, or readers.

12. A Generic Processing Node

The relationships and interactions between the BG, WM, SP, VJ, and KD modules in a generic node of the RCS architecture are shown in Fig. 4. The Behavior Generating (BG) modules contain submodules of Planner (PL) and Executor (EX). Planner has submodules of Job Assignment (JA), Scheduling (SC) and Plan Selector (PS). The World Modeling (WM) module contains the Knowledge Database (KD), with both long term and short term symbolic representations and short term iconic images. In addition, WM contains a Simulator where the alternative plans generated by JA and SC are simulated for predicting results. The Value Judgment (VJ) module evaluates predicted results. The Sensory Processing (SP) module contains filtering, detecting, and estimating algorithms, plus mechanisms for comparing predictions generated by the WM module with observations from sensors. It has algorithms for recognizing entities and clustering entities into higher level entities. Perceived results are analyzed in VJ

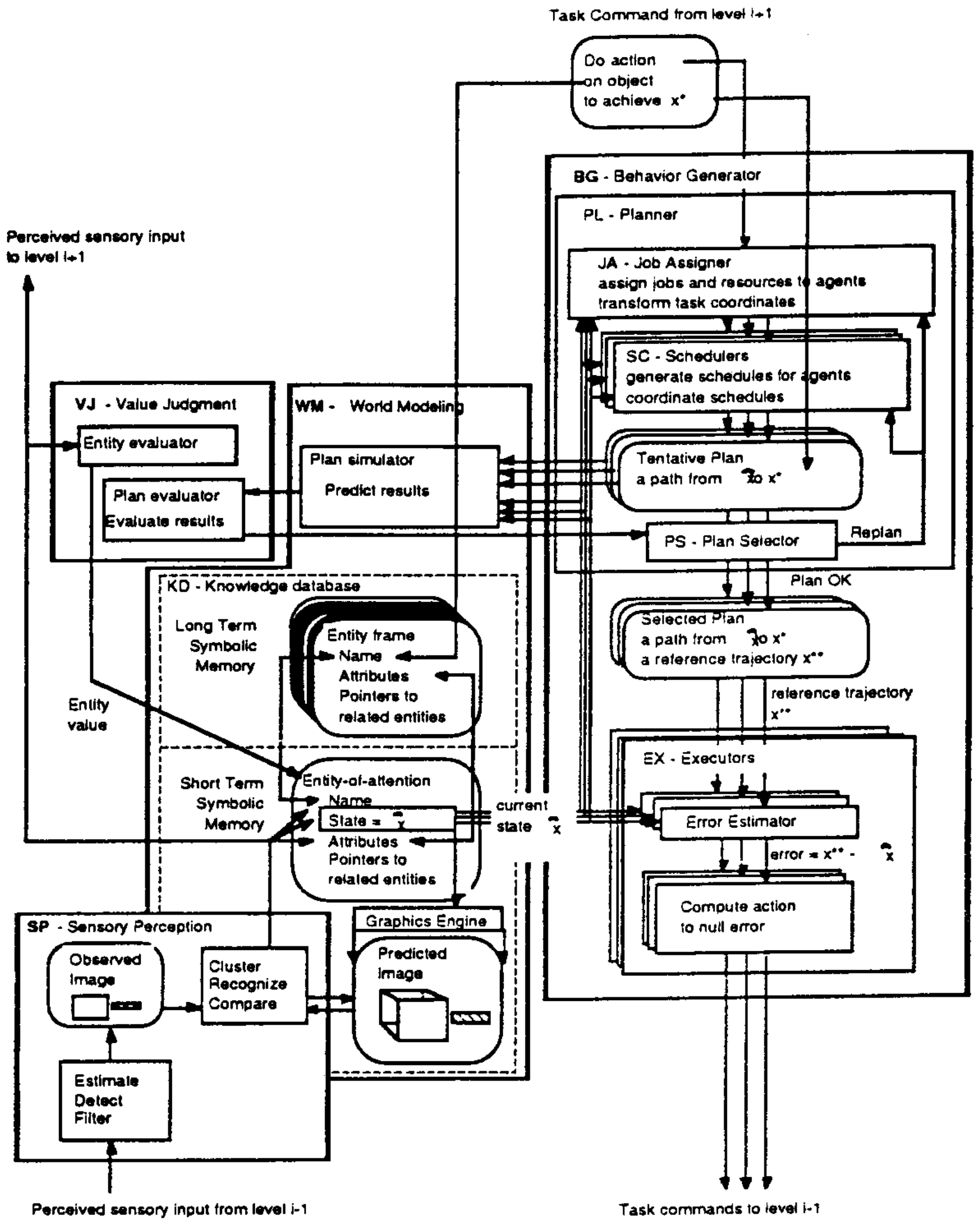


Fig. 4. Relationships within a single node of the RCS architecture. The Behavior Generating (BG) modules contain Job Assignment (JA), Scheduling (SC), Plan Selector (PS) and Executor (EX) submodules. The World Modeling (WM) module contains a plan simulator and mechanisms for updating the Knowledge Database (KD), which contains both long term and short term symbolic representations and short term iconic images. The Sensory Perception (SP) module contains filtering, detecting, and estimating algorithms, plus mechanisms for comparing predictions generated by the WM module with perceived input from sensors. It has algorithms for recognizing entities and clustering entities into higher level entities. The Value Judgment (VJ) module evaluates plans and places values on entities recognized in the observed sensory input.

as good or bad, successful or unsuccessful, and confidence factors are computed based on the variance between observed and predicted sensory input.

Each node of the RCS hierarchy closes a control loop. Input from sensors is processed through Sensory Processing (SP) modules and used by the World Modeling (WM) modules to update the Knowledge Database (KD). This provides a current best estimate $\hat{x}$ of the state of the world x. This information provides a feedback signal to the EX module which compares it with the planned reference trajectory. The EX submodule then uses a control law to compute the compensation required to minimize the eventual difference between the planned reference trajectory and the system performance which emerges as a result of the control process. The current best estimate of the world state is also used by the JA and SC submodules and by the WM plan simulator to perform their respective planning functions. $\hat{x}$ is also used in generating a short term iconic image that forms the basis for masking, filtering, windowing, comparison, recognition, and recursive estimation in the image domain in the sensory processing SP module.

13. Elementary Loop of Functioning

The bandwidth of the control loop through each node is determined by the sampling rate of the sensors, the filter properties of the SP and WM submodules, and the computation update frequency of the EX submodule. The control loop, or elementary loop of functioning, through a typical control node is shown in Fig. 5.

A task command specifying a goal G arrives into BG module of a generic control node. The function of the BG module is to transform the task into a set of subtasks to be submitted to the set of Actuators {A}. All actuators operate on their respective Worlds {W} producing effects that are monitored by the set of Sensors {S}. The signals from the sensors are processed and integrated within the Sensory Perception (SP) module which updates the representation contained in the World Model (WM). The dotted lines between the WM and the set {W} indicate a virtual correspondence between the real world W and its representation in WM. The knowledge in WM is the system's best estimate of the real state of W. The virtual correspondence is maintained via noisy chan-

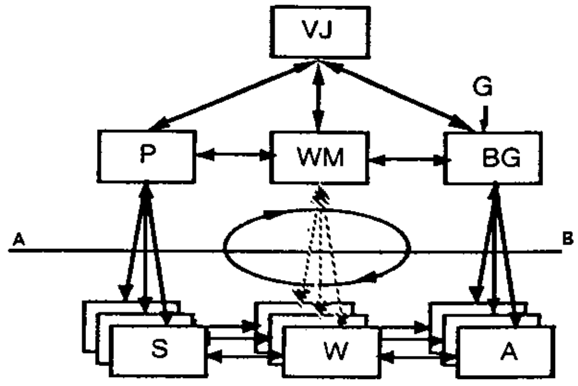


Fig. 5. An elementary loop of functioning.

nel through the Sensors {S} by filtering and recursive estimation processes in the SP and WM modules. To the extent that the correspondence between W and WM is an accurate, or at least adequate, representation of reality, the behavior generated by BG is more likely to be successful in achieving the goal G. To the extent that the correspondence is incorrect, behavior is less likely to be successful.

Figure 5 describes the Loop of Functioning consisting of two domains — Controller System and Controlled System. The line AB is a divider between these domains. It denotes the fact that the modules above this line are components of the controller system while the modules underneath are the controlled system.

In Fig. 6, a second control level is shown. The upper, or outer loop, has lower resolution in both time and space. It has a lower bandwidth, a longer planning horizon, a larger span of control over subordinate agents, and typically deals with larger range of space. The inner loop, has a higher loop bandwidth, a shorter planning horizon, a smaller span of control and deals with a smaller spatial range with higher resolution. The diagram in Fig. 6 is essentially an expanded version of one of the nodes in level two of the architecture shown in Fig. 1.

14. Hierarchical Layering

Intelligent control systems can be extremely complex. Maintaining a rich representation of the world in the knowledge database, and using this representation to plan and execute sophisticated behavior with high probability of success can require enormous amounts of computing power. An intelligent

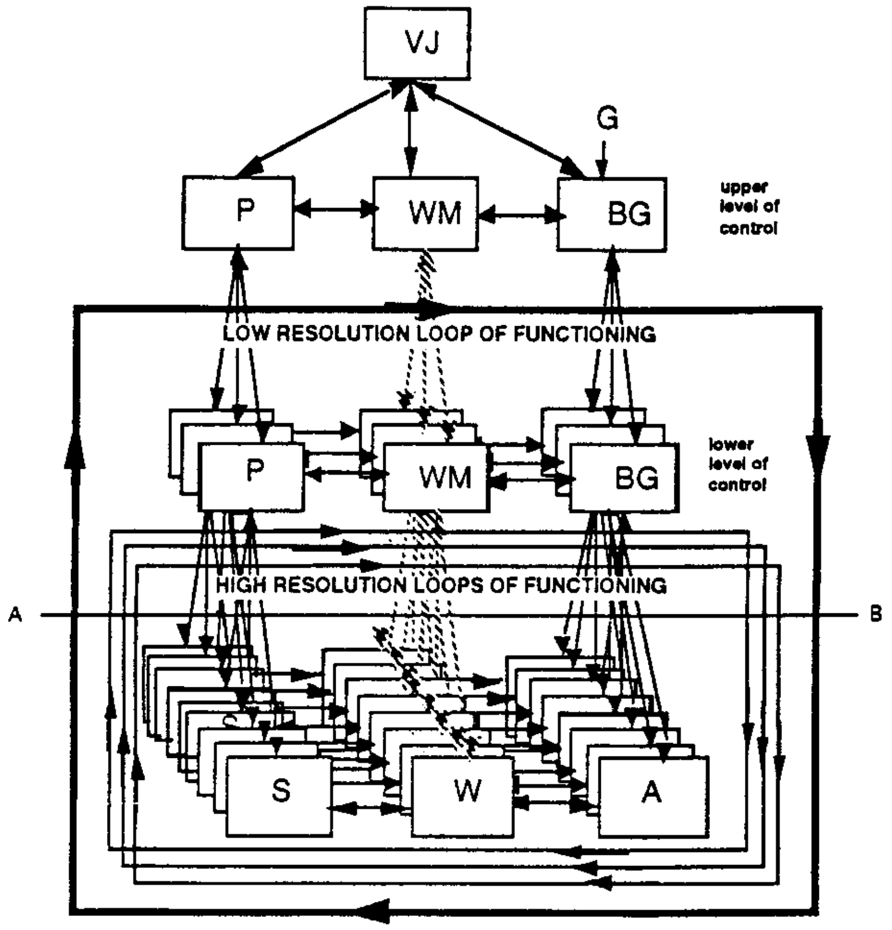

VJ
G
P
WM
BG
upper level of control
LOW RESOLUTION LOOP OF FUNCTIONING
P
WM
BG
lower level of control
HIGH RESOLUTION LOOPS OF FUNCTIONING
A
B
S
W
A

Fig. 6. An elementary loop with two control levels.

control system may need to store symbolic information about thousands of entities in long term memory. Visual images in short term memory may consist of hundreds of thousands of pixels, each of which may have numerous attributes that change from one frame to the next. Correlating incoming sensory data with stored information is a computationally intensive process. Sophisticated intelligent control systems can require gigabytes of memory and gigaflops of processing power.

Hierarchical layering is a means for managing the complexity that is inherent in intelligent systems. In a hierarchical system, higher levels have broader scope and longer time horizons, with less detail. Lower levels have narrower scope and shorter time horizons, with more detail. For example, Behavior Generating modules at the upper levels can make long range plans consisting of very general goals, while at lower levels, Behavior Generating modules can successively refine the long range plans into short term tasks with great detail. At lower levels, Sensory Processing modules typically process data over local neighborhoods and short time intervals; while at higher levels, Sensory Processing modules integrate data over long time intervals and large spatial regions. At low levels, World Model data is typically short term and detailed; while at the higher levels it is broad in scope and general. At every level, feedback loops are closed to provide reactive behavior, with high-bandwidth fast-response loops at lower levels, and slower reactions at higher levels.

Intelligent systems typically operate in a real world environment which is infinitely rich with detail. Yet the computational resources available to any intelligent system are finite. No matter how fast and powerful computers become, the amount of computational resources that can be embedded in any practical system will be limited. Even the number of neurons in the human brain is limited. Fortunately, at any point in time and space, most of the detail in the environment is irrelevant to the immediate task of the intelligent system. Therefore, the key to building practical intelligent system lies in understanding how to focus the available computing resources on what is important, and ignoring what is irrelevant.

Hierarchical perception and task decomposition enables the higher levels with broad perspective to determine where and when to focus computing resources of the lower levels. The higher levels maintain a broad perspective necessary for determining what is important, while the lower levels attend to details. The problem of distinguishing what is important from what can be safely ignored is the central problem of attention. Top down, what is important is defined by behavioral goals and priorities. Bottom up, what is important is defined as the unexpected, unexplained, unusual, or threatening. In either case, hierarchical layering provides a mechanism for focusing attention on what is important in both time and space.

At the top of the hierarchy, categorial imperatives such as ⟨make a profit⟩ (for intelligent manufacturing systems), or ⟨propagate genes⟩ (for biological creatures), influence the selection of goals and the prioritization of tasks throughout the hierarchy.

At intermediate levels, tasks with goals and priorities are received from the level above, and subtasks with subgoals and attention priorities are output to the level below. For example, in military unmanned ground vehicles, intermediate level tasks might be of the form ⟨go to map position (x, y)⟩, or ⟨provide reconnaissance and target acquisition functions for sector (a1, a2)⟩.

At the bottom of the hierarchy and external to the control system, are actuators that act on the world environment, and sensors that transform events in the world into information signals for the control system.

In each node at each level of the hierarchy, a set of lower level agents work together under supervision of a higher level agent. Each of the lower level agents is considered a subagent for the higher level agent. At each level, global goals are refined and focused onto more narrow and higher resolution subgoals. At each level, attention is focused into a more narrow and higher resolution view of the world. Each hierarchical level geometrically refines the detail of the task and the view of the world, while only linearly increasing the computational demands placed upon the intelligent system.

At all levels, agents process input from sensors, or from lower levels via SP and WM modules; they estimate the state of the world as perceived from their level of resolution; they accept goals and priorities from higher level agents; they make plans to accomplish those goals; and they close a control loop. At each level, there is a characteristic loop bandwidth, a characteristic planning horizon, a characteristic set of task skills, a characteristic range of temporal integration of sensory data, and a characteristic window of spatial integration. The result is a system with both deliberative and reactive capabilities tightly integrated at all levels of resolution in space and time.

15. Applications

Over the past two decades, the RCS architecture has been used in the implementation of a number of experimental projects. These include:

(1) A horizontal machining workstation
This project was part of the NBS Automated Manufacturing Research Facility (AMRF).[5] It included an integrated sensory-interactive real-time control system for a robot with a structured light machine vision system, a machine tool, an automatic fixturing system, and a pallet shuttle. The robot included a quick change wrist, a part handling gripper with tactile sensors, and a tool handling gripper for loading and unloading tools in the machine tool magazine. Plans were represented as state-tables, and a wide variety of sensory interactive behaviors were demonstrated. These included locating and recognizing parts and determining part orientation of unoriented parts presented in trays, and automatically generating part handling sequences for part and tool loading and unloading.[6]

(2) A cleaning and deburring workstation
This project was also part of the AMRF. It included two robots, a set of buffing wheels, a part washer/dryer machine, and a variety of abrasive brushes. Part geometry was input from a CAD database. Deburring tool paths were automatically

planned from knowledge of the part geometry plus operator input indicating which edges were to be deburred. Deburring parameters such as forces and feed rates were also selected from a menu by the operator. Part handling sequences were planned automatically for loading parts in a vise, and turning parts over to permit tool and gripper access. Force sensors and force control algorithms were used during task execution to modify the planned paths so as to compensate for inaccuracies in robot kinematics and dynamics.[7]

(3) An advanced deburring and chamfering system
This project is currently nearing completion. The project integrates off-line programming, real-time control, and active tool technologies to automatically place precision chamfers on complex parts manufactured from hard materials such as aircraft jet engine components. The workstation consists of a force sensitive active tool integrated with a 6 degree-of-freedom robot and an indexing table used for part manipulation. The active tool, the Chamfering and Deburring End-of-arm Tool (CADET), incorporates actuators and force sensors to provide control over cutting force and tool stiffness at the part edges. Part geometry is derived from standard IGES CAD data formats. Edge selection is performed by a human operator. Required tool force is automatically generated by formula using the cutting depth, feeds, and speeds input by the operator. A prototype production cell will be installed at Pratt & Whitney's East Hartford, CT site upon completion of the project.[8]

(4) NBS/NASA standard reference model architecture for the space station telerobotic servicer (NASREM)
This project was funded by NASA Goddard Space Flight Center. NASREM was used by Martin Marietta to develop the control system for the space station teletobotic servicer. Algorithms were developed for force servoing, impedance control, and real-time image processing of robotic and telerobotic systems at NIST, Martin Marietta, Lockheed, Goddard, and in a number of university and industry labs in the United States and Europe.[9]

(5) An architecture for coal mining automation
This project transferred RCS architecture and methodology to a team of researchers in the U.S. Bureau of Mines, and in turn, to the mining industry. A comprehensive mining scenario was developed starting with a map of the region to be excavated, the machines to be controlled, and the mining procedures to be applied. Based on this scenario, an intelligent control system with simulation and animation was designed, built, and demonstrated. The same control system was later demonstrated with an actual mining machine and sensors.[10]

(6) A nuclear submarine maneuvering system
This project demonstrated the design and implementation in simulation of maneuvering and engineering support systems for a 637 class nuclear submarine. The maneuvering system involves an automatic steering, trim, speed and depth control system. The system demonstrated the ability to execute a lengthy and complex mission involving transit of the Bering Straits under ice. Ice avoidance sonar signals were integrated into a local map using a CMAC[11] neural network memory model. Steering and depth control algorithms were developed that enabled the sub to avoid hitting either the bottom or the ice while detecting and compensating for random salinity changes under the ice by making trim and ballast adjustments. The engineering support system demonstrated the ability to respond to an emergency such as a lubrication oil fire by reconfiguring ventilation systems, rising in depth to snorkel level, and engaging the diesel engines for emergency propulsion.[12]

(7) A control system for a US postal service automated stamp distribution center
This system demonstrated the ability to route packages through a series of carousels, conveyors, and storage bins, to maintain precise inventory control, provide security, and generate maintenance diagnostics in the case of system failure. The distribution center was designed and tested first in simulation, and then implemented as a full scale system. The system contained over 220 actuators, 300 sensors, and ten operator workstations. An even larger and more complex RCS system for controlling a general mail facility is still under development.[13]

(8) A control system for multiple autonomous undersea vehicles
This system was developed for controlling a pair of experimental vehicles designed and built by the

University of New Hampshire. The RCS control system included a real-time path planner for obstacle avoidance, and a real-time map builder for constructing a topological map of the bottom. A series of tests was conducted in Lake Winnipasaki during the fall of 1987.[14]

(9) An RCS system for remote driving
This system was implemented on an Army HMMWV light truck. One version of the system enables the vehicle to be driven remotely by an operator using TV images transmitted from the vehicle to an operator control station. This version has a retrotraverse mode that permits the vehicle to autonomously retrace paths previously traversed under remote control, using GPS and an inertial guidance system.[15]

A second version of this RCS system has demonstrated the ability to drive the HMMWV automatically using TV images processed through a machine vision system with a real-time model matching algorithm for tracking lane markings. The RCS real-time vision processing system has enabled this vehicle to drive automatically at speeds up to sixty miles per hour on the highway, and at speeds up to thirty-five miles per hour on a winding test track used by the county police for driver-training.[16]

(10) An open architecture enchanced machine controller
The RCS reference model is being used as the basis for an open architecture Enhanced Machine Controller (EMC) for machine tools, robots, and coordinate measuring machines. The EMC combines NASREM with the Specification for an Open System Architecture Standard (SOSAS) developed under the Next Generation Controller program sponsored by the Air Force and National Center for Manufacturing Sciences. In cooperation with the DoE TEAM (Technology for Enabling Agile Manufacturing) program, EMC functional modules have been defined, and Application Programming Interfaces (APIs) are being specified for sending messages between the functional modules. A prototype machine tool controller has been installed and is being used in production in a General Motors plant as part of the DoE-TEAM/NIST-EMC government/industry consortium. The goal of this effort is to develop API standards for open architecture controllers.[17]

(11) A planning and control system for a spray casting machine
The RCS architecture has been applied for planning and control of the automated Spray Casting Machine "OSPREY" which has been developed and manufactured by MTS Corporation (Minneapolis, MN) in cooperation with Drexel University. The system has three levels of resolution.[20]

(12) An autonomous mobile vehicle
An autonomous vehicle was assembled and tested by Drexel University in 1984–1987. The goal of the effort was to investigate the RCS architecture with four levels of resolution "Planner-Navigator-Pilot" on the top of the lower level control of steering and propulsion. The results of this research was described in Ref. 21.

All of the projects listed above that have used the RCS architecture have implemented only a subset of the features of the most advanced theoretical form of the RCS reference model architecture. This is because the RCS theoretical development has remained well advanced over what it has been possible to implement, given programmatic limitations in funding. Current work at NIST and elsewhere is pursuing more complex implementations of RCS. For example, efforts to incorporate human operator interfaces into the RCS architecture that began with NASREM have continued with the Air Force/JPL/NIST Universal Telerobotic Architecture Project (UTAP), and the NIST RoboCrane. Work is also under way to integrate the RCS architecture with the NIST Manufacturing Systems Integration (MSI) factory control architecture, and the NIST Quality In Automation (QIA) architecture.[18] When complete, this joint architecture will define a reference model architecture for manufacturing systems that extends all the way from the servomechanism level to the enterprise integration level. Work is also in progress to develop an engineering design methodology and a set of software engineering tools for developing RCS systems.[19]

16. Discussion and Conclusions

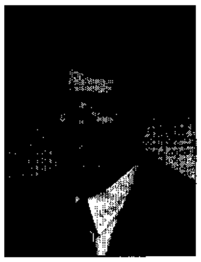
The RCS reference model architecture derives from a control theory approach. It has evolved over the past two decades from a rather simple robot control schema to a reference model architecture for intelligent control system design. From the beginning, work on RCS has represented a conscious attempt to emulate the function and structure of the neurological machinery in the brain. For example, each RCS state variable is analogous to a firing-rate signal on a neuronal axon. Each RCS function, or submodule, has properties that are known, or at least hypothesized, to exist in the neurological structures in the brain. Most RCS functions can be constructed from neural nets such as CMAC[11] that compute arithmetic and/or logical functions on a set of inputs to produce a set of output state variables. These can be carried over communications pathways that mimic neural axons to other functional modules that may use them to perform further functional computations, or to generate addresses, or to store information in memory. RCS functional modules may add, subtract, multiply, differentiate, integrate (both spatially and temporally), compute variance and correlation functions, recognize patterns, transform coordinates, compute statistical parameters, perform filtering, masking, warping, and scrolling operations on arrays and images, generate names or addresses of symbolic variables and lists, remember state and act as finite state machines, store and recall attribute values, perform recursive estimation, execute control laws, and formulate and test hypothesized actions. RCS modules are arranged so as to close control loops and planning loops and to process sensory feedback through a variety of filters and integrators with different time intervals so as to create servo loops with a variety of bandwidths at a hierarchy of levels. All the functional modules in RCS are designed as independent concurrently executing analog, or cyclically executing sampled data processes, that continuously monitor their inputs and compute their outputs in a manner similar to the way that tightly coupled collections of neurons in the brain do. Each RCS module may be implemented as an independent task in a multitasking operating system, as an object in an object oriented software architecture, or as an independent agent in an agent architecture. Just as neurons continuously monitor their synaptic inputs and compute axonal outputs, so RCS modules in a Von Neumann computer cyclically sample their inputs and compute their outputs on a clock cycle determined by the bandwidth of the process.

The RCS reference model architecture thus provides a framework for integrating control concepts with artificial intelligence, neural nets, machine vision, robotics, computer science, operations research, game theory, signal processig, filtering, and communications theory.

References

1. J. S. Albus, Outline for a theory of intelligence, *IEEE Trans. Systems, Man and Cybernatics*, **21** 3, May/June 1991, 473–509.
2. J. S. Albus, Brains, Behavior, and Robotics (Byte/McGraw-Hill, 1981).
3. J. S. Albus, A reference model architecture for intelligent systems design, *An Introduction to Intelligent and Autonomous Control*, eds. P. J. Antsaklis and K. M. Passino, 1993.
4. A. Meystel, Nested hierarchical control, *An Introduction to Intelligent and Autonomous Control*, eds. P. J. Antsaklis and K. M. Passino, 1993.
5. J. S. Albus, C. R. McLean, A. J. Barbera and M. L. Fitzgerald, Architecture for real-time sensory-interactive control of robots in a manufacturing facility, *Proc. Fourth IFAC/IFIP Symposium — Information Control Problems in Manufacturing Technology*, Gaithersburg, MD, October 26–28, 1982.
6. A. J. Wavering and J. C. Fiala, Real-time control system of the horizontal workstation robot, NBSIR 88-3692, National Institute of Standards and Technology, Gaithersburg, MD, December 1987.
7. K. N. Murphy, R. J. Norcross and F. M. Proctor, CAD directed robotic deburring *Proc. Second Int. Symp. on Robotics and Manufacturing Research, Education, and Applications*, Albuquerque, NM, November 16–18, 1988.
8. K. Stouffer, J. Michaloski, R. Russell and F. Proctor, ADACS — An automated system for part finishing, NISTIR 5171, National Institute of Standards and Technology, Gaithersburg, MD, April 1993, and *Proc. IECON '93 Int. Conf. on Industrial Electronics, Control and Insirumentation*, Maui, Hawaii, November 15–19, 1993.
9. J. S. Albus, H. G. McCain and R. Lumia, NASA/NBS standard reference model for telerobot control system architecture (NASREM), NISTTN 1235, 1989, Ed. National Institute of Standards and Technology, Gaithersburg, MD, April 1989 (supersedes NBS Technical Note 1235, July 1987).
10. H. M. Huang, R. Quintero and J. S. Albus, A reference model, design approach, and development illustration toward hierarchical real-time system control for coal mining operations, *Advances in Control and Dynamic Systems* (Academic Press, July 1991).
11. J. S. Albus, New approach to manipulator control: The cerebellar model articulation controller (CMAC), and Data storage in the cerebellar model articulation controller (CMAC), *Trans. ASME Journal of Dynamic Systems, Measurement, and Control*, September 1975.
12. H. M. Huang, R. Hira and R. Quintero, A submarine maneuvering system demonstration based on

the NIST real-time control system reference model, *Proc. Eighth IEEE Int. Symp. on Intelligent Control*, Chicago, IL, August 24–27, 1993.
13. Stamp Distribution Network, USPS Contract Number 104230-91-C-3127 Final Report, Advanced Technology & Research Corp., Burtonsville, MD, 20866-1172.
14. M. Herman and J. S. Albus, Overview of the Multiple Autonomous Underwater Vehicles (MAUV) project, *IEEE Int. Conf. on Robotics and Automation*, Philadelphia, PA, April 1988.
15. S. Szabo, H. A. Scott, K. N. Murphy and S. A. Legowik, Control system architecture for a remotely operated unmanned land vehicle, *Proc. Fifth IEEE Int. Symp. on Intelligent Control*, Philadelphia, PA, September 1990.
16. H. Schneiderman and M. Nashman, Visual tracking for autonomous driving, *IEEE Trans. on Robotics and Automation*, **10** 6 (December 1994) 769–775.
17. F. Proctor and J. Michaloski, Enchanced machine controller architecture overview, NISTIR 5331, National Institute of Standards and Technology, Gaithersburg, MD, December 1993.
18. M. K. Senehi, T. J. Kramer, J. Michaloski, R. Quintero, S. R. Ray, W. G. Rippey and S. Wallace, Reference architecture for machine control systems integration: Interim report, NISTIR 5517, National Institute of Standards and Technology, Gaithersburg, MD, 1994.
19. R. Quintero and A. J. Barbera, A software template approach to building complex large-scale intelligent control systems, *Proc. Eighth IEEE Int. Symp. on Intelligent Control*, Chicago, IL, September 25–27, 1993.
20. B. Cleveland and A. Meystel, Predictive planning + fuzzy compensation equals intelligent control, *Proc. Fifth IEEE Int. Symp. on Intelligent Control*, Philadelphia, PA, September 1990.
21. A. Meystel, *Autonomous Mobile Robots: Vehicles with Cognitive Control* (World Scientific 1991) 600 pages.

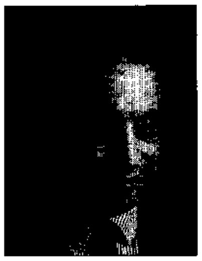


J S Albus is presently Chief of the Intelligent Systems Division, Center for Manufacturing Engineering, National Institute of Standards and Technology. He has been involved in the study of intelligent systems for over twenty five years. During that time, he has designed and managed the development of a number of intelligent systems, including the NBS Automated Manufacturing Research Facility and the NIST RoboCrane. He was co-developer of the NIST Real-time Control System (RCS) architecture, and has received several awards for his work in control theory including the National Institute of Standards and Technology Applied Research Award, the Department of Commerce Gold and Silver Medals, the Industrial Research IR-100 award, the Construction Equipment magazine and Popular Science magazine best technologies awards, and the Joseph F Engelberger Award for robotics technology, presented at the 1984 International Robot Symposium by the King of Sweden.

Before coming to the National Institute of Standards and Technology, Dr Albus worked 15 years for NASA Goddard Space Flight Center where he designed electro-optical systems for more than 15 NASA spacecraft. For a short time, he served as program manager of the NASA Artificial Intelligence Program.

Dr Albus is the author of numerous scientific papers, journal articles, and official government studies. He has also written for popular publications such as Scientific American, OMNI, BYTE, and the FUTURIST.

He has written two books, "Brains, Behavior, and Robotics" (Byte/McGraw Hill 1981) and "Peoples' Capitalism: The Economics of the Robot Revolution" (New World Books 1976).



A M Meystel is a Professor of Electrical and Computer Engineering at Drexel University. Since 1995, he is at the National Institute of Standards and Technology, on leave from Drexel University. He has developed a novel concept and outlined the theory of nested multiresolutional control systems. His concept of "Planner-Navigator-Pilot" architecture was broadly accepted in the area of robotics in the '80s. He is the author of the book "Autonomous Mobile Robots: Vehicles with Cognitive Control" (World Scientific Publishing, 1991). He is on the editorial board of three journals in control and robotics.

Dr Meystel was an initiator of annual IEEE International Symposium on Intelligent Control, and served as general and program chairman at four out of eight Symposia since 1985. He is responsible for AI at the IEEE Technical Committee on Intelligent Control, and is the moderator of the AICS list: an electronic discussion group of Architectures for Intelligent Control Systems. He was the Chair and Co-Chair of four out of five annual IEEE Workshops on Architectures for Intelligent Control Systems. During the last three years, Dr Meystel have presented 10 National and International Invited Lectures dedicated to the design development and better understanding of the multiresolutional approach in different areas of human activity.