J. EXPT. THEOR. ARTIF. INTELL. 9(1997)157–174

The NIST Real-time Control System (RCS): an approach to intelligent systems research

JAMES S. ALBUS

Chief, Intelligent Systems Division National Institute of Standards and Technology Gaithersburg, MD 20895, USA 301-975-3418

Abstract. Differences between the deliberative and reactive aspects of behaviour have led to different scientific approaches that have divided the field of cognitive research for at least a century. In the tradition of the behaviourist school of psychology, many present day researchers in advanced robotics and artificial intelligence have abandoned inquiry into the complexities of cognitive behaviour in favour of a minimalist philosophy that focuses primarily on stimulus–response reactivity. In contrast, the Real-time Control System (RCS) developed at NIST and elsewhere over the past two decades provides a model for bridging the gap between deliberative and reactive behaviours. RCS is a reference model architecture for intelligent systems design that consists of a hierarchically layered set of processing nodes. In each node, there are both cognitive and reactive elements. 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 all levels and time frames. A comparison is made between RCS and subsumption and some illustrative examples of RCS applications are given.

1. Introduction

In recent years, many researchers in advanced robotics and artificial intelligence have abandoned inquiry into the complexities of cognitive behaviour in favour of a minimalist philosophy that denies the need for complicated world models and internal representations of knowledge, sophisticated analysis of situations and events, or complex reasoning about the past or planning for the future. In many laboratories, it has become fashionable instead to study machines that simply react, without resort to explicit representations or logical reasoning (Brooks 1990). Attempts to understand the mechanisms of perception and cognition have been replaced by the study of systems wherein signals travel directly from sensors to actuators without mediating cognitive processes that estimate, recognize, reason, or analyse. The study of behaviour is reduced to that which is emergent and reactive without consideration of what is intentional and planned. Abstract knowledge and reason play little or no role.

Differences between the deliberative and reactive aspects of behaviour have puzzled philosophers and scientists for centuries. There have developed different schools of inquiry that have divided the field of cognitive research for at least a century (Flanagan 1991). The current minimalist philosophy shares much in common with the behaviourist movement that dominated the field of experimental psychology earlier

0952–813X/97 $12·00

in this century. The behaviourist school of psychology championed by Watson and Skinner (Skinner 1953) admitted no concept of intentionality or internal representation, neither in their experiments, nor in their theories. Stimulus–response was their entire domain of interest. Learning based on reward and punishment of behaviour was the primary focus of experimentation. In some respects, Rodney Brooks is to robotics what B. F. Skinner was to psychology.

The rationale given by the behaviourists for banishing notions of intent and internal knowledge was that it was necessary to eliminate unobservable factors such as free will, motivation, and intention, and to disallow supernatural agents such as the spirit and soul, in order to place the study of human behaviour on a rational scientific basis. However, in so doing, the behaviourist school also eliminated from consideration the most important features of intelligent behaviour, namely the ability to pursue internal goals, to think about and plan for the future, to reason using abstract concepts, and to imagine things that are not directly observable.

The basis for the minimalist approach to robotics research is partly a revolt against traditional AI preoccupation with symbolic logic, and partly pragmatic. There are fundamental problems inherent in attempting to describe the richness of the natural world entirely in terms of symbols and lists. It is difficult to control high-performance machines in complex real-world environments using symbolic logic and predicate calculus. There are immense practical problems in building algorithms that can perform image processing, symbolic reasoning, and search-based planning fast enough to incorporate these operations in a closed-loop reactive control system. It is not uncommon to see supposedly 'intelligent' laboratory robots moving painfully slowly, with long pauses for image analysis, planning, and reasoning between each movement. Image processing systems often require many seconds, even minutes or hours, of processing time to analyse a single picture.

It is much easier to build robotic systems that react in real time if computational complexity is kept to a minimum. Minimalist machines that react simply but immediately to sensory input offer an attractive alternative to more complex traditional machine intelligence. The behaviour of the minimalist robots is certainly interesting to watch, and to many observers appears more intelligent than that of their slower moving counterparts. It is surprising to most people how much complex behaviour can be generated by simple machines equipped with a few sensors when placed in a complex environment. Robots with minimal sensing and computational power can wander about a room, avoid collisions, and acquire objects of various kinds. Computer simulations of minimalist cooperative behaviour such as flocking, or schooling, or even hunting in packs have shown how complex behaviours can emerge from very simple algorithms using only primitive sensors. Studies of agent architectures have demonstrated that groups of independent agents using simple negotiating tactics can generate mutually beneficial strategies quickly and efficiently.

There is also considerable evidence from the natural world that minimalists strategies can be successful. Ants and termites are able to build nests consisting of complex networks of tunnels and chambers with only the most primitive sensory apparatus and virtually no reasoning capabilities. Spiders can build webs that are engineering marvels, and use them skillfully to capture prey. There are many species that have survived quite successfully for very much longer than human beings without any significant deliberative capabilities whatsoever. Reflexive behaviour coupled with instinct is obviously all that is needed for lower forms such as insects to succeed in propagation of the species.

However, this does not provide an explanation for intelligence near the top of the evolutionary ladder. Human behaviour is strongly influenced by internal knowledge, abstract reasoning, and goal directed intentions. Human visual observations are rich in detail. We perceive the visual world not simply as an array of pixels or moving blobs, but as scenes filled with complex objects in sophisticated relationships that can be understood and subjected to reasoned analysis. Objects and situations can be evaluated in terms of abstract knowledge retrieved from memory and not directly observable. We routinely devise intricate strategies to pursue goals, to avoid risk, and to compete for wealth, power, and social status. In humans, spoken and written languages can convey abstract ideas, and can relate tales of adventure and romance that evoke emotional feelings of love, hate, fear, and triumph.

Humans are also capable of reasoning about the past and planning for the future. They can pursue actions based on goals that are often unrelated to current sensory input. People can make plans for tomorrow, next week, and next year. They can visualize the results of future actions, and estimate costs, risks, and benefits. They can construct mental images of things that can only be imagined, such as gods and devils, angels and ghosts, germs and galaxies, molecules and black holes. They can imagine and dream, hope and fear. Much of human behaviour is more strongly influenced by internal models such as career goals, or belief in future rewards in heaven, than by current conditions that can be directly observed by sensors.

The ability to see at a distance, to recognize objects, to analyse situations, to estimate states of the world that are not directly observable, to reason about and plan for the future, to calculate the capabilities and intentions of an opponent, and to communicate this kind of information with others – these are the characteristics of higher levels of intelligence. Without accounting for these phenomena, no theory can claim to explain intelligence. These capabilities cannot be achieved without internal representations that are rich and complex, and that permit representation of both iconic and symbolic (i.e. depictional and propositional) knowledge. Understanding how such behaviours are generated should remain at the centre of advanced robotics and artificial intelligence research.

Thus, although the minimalist approach has demonstrated that simple reflexive mechanisms can produce complex and interesting behaviour, this important result should not be interpreted as a shortcut to understanding intelligence or as a substitute for the study of higher level cognitive mechanisms. Simple behaviours alone seldom succeed except when applied in large numbers. Termites and cockroaches are successful primarily because they reproduce so rapidly that the fate of any individual is irrelevant. The fact is that simple strategies rarely defeat sophisticated intelligence when applied one-on-one. The run-and-hide tactic of the mouse is only temporary protection against a determined wait-and-pounce strategy of a hungry cat. The schooling behaviour of fish provides no defence against a well designed fisherman's net. The use of sticks and clubs rarely succeeds against the bow and arrow, or the high powered rifle. Primitive hunting and gathering societies have little chance against agricultural or industrial technologies.

This does not mean reflexive behaviour is unimportant, even for humans. There is often no time to plan, and even well laid plans cannot always prevent bad things from happening. Unexpected events occur, and when they do, the ability to react quickly may be more important than the capability of deep analysis. Even humans have, and need, reflexes. Many human abilities to hunt prey, escape danger, and compete for territory and social status depend on quick reaction to unexpected sensory input.

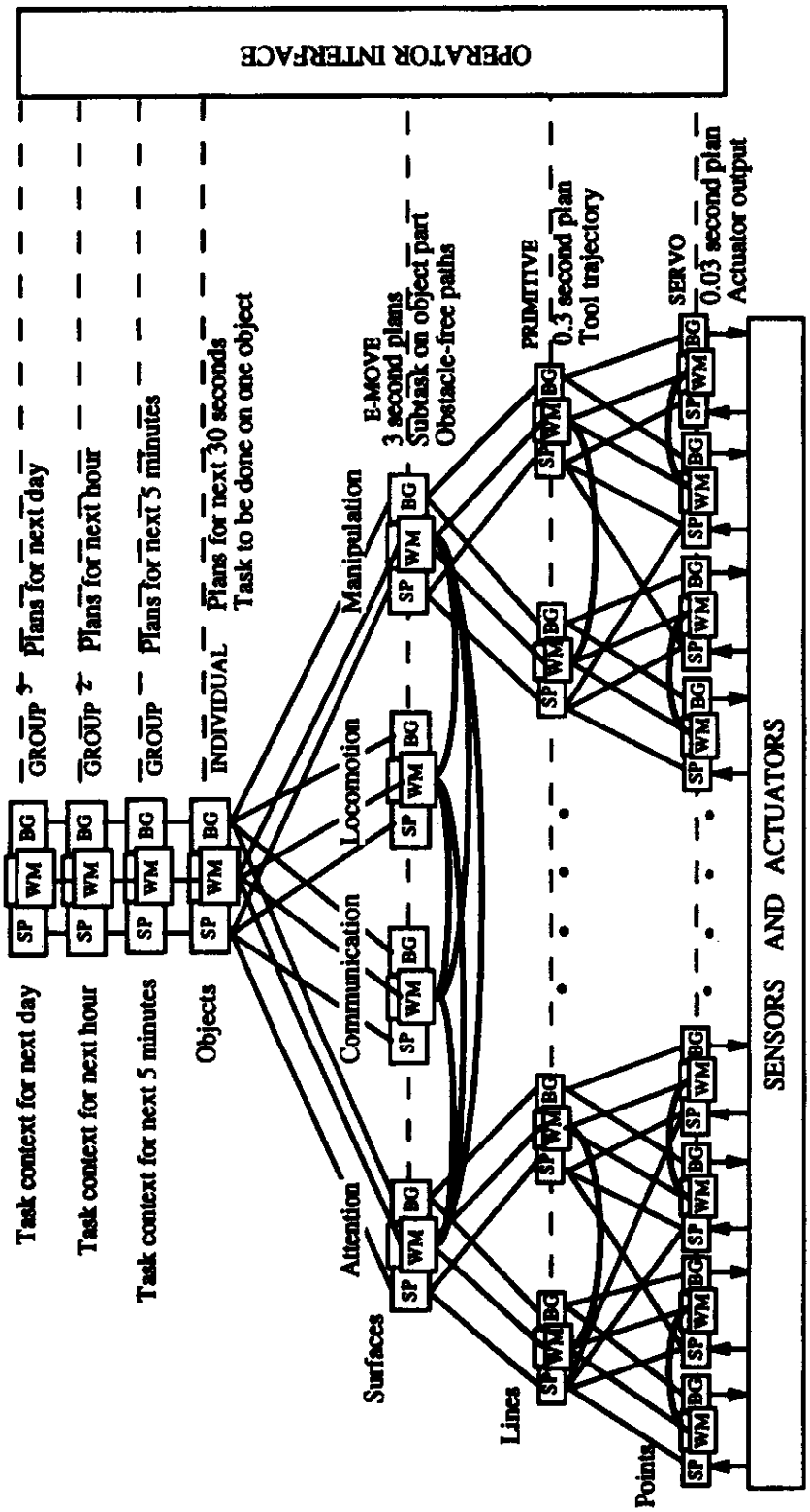

Figure 1. RCS reference model architecture for intelligent systems. Processing nodes are organized such that the BG modules form a command tree. Information in the knowledge database is shared between WM modules in nodes within the same subtree. On the right are examples of the functional characteristics 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.

Long-term goals and deliberative strategies often cannot be successfully carried out without quick reaction to rapidly changing events in a competitive and unpredictable world environment.

Therefore, the central issue in understanding intelligence is not in choosing between deliberative OR reflexive systems, but in understanding how to combine and blend both deliberative AND reflexive systems in a single integrated architecture that can both plan for the future and react to unexpected events in the present so as to produce a seamless continuum of behaviour. Truly intelligent behaviour consists of both deliberative mechanisms that generate plans and strategies for looking ahead so as to avoid danger and achieve desirable goals, and reflexive mechanisms that deal with immediate reactions to sensed conditions. Human intelligence is a sophisticated combination of goal-driven and reactive behaviours—a mixture of deliberative and reflexive mechanisms. As species ascend the evolutionary ladder, the deliberative does not replace the reflexive, but is added to, and overlaid on, the reflexive. Intelligence evolves and grows as the capacity to plan and imagine the future is combined with the more primitive capabilities to react and respond. The central issue in designing intelligent systems is how to integrate both deliberative and reflexive capabilities into a single efficient control system architecture.

2. Bridging the gap

The Real-time Control System (RCS) (Albus 1981, 1991, 1993) developed at NIST and elsewhere over the past two decades provides, among other things, a model for bridging the gap between the deliberative and the reflexive. RCS is a reference model architecture for intelligent systems design that consists of a hierarchically layered set of processing nodes connected together by a network of communications pathways as shown in Figure 1. At each layer of the RCS hierarchy sensory data are processed, entities are recognized, world model representations are maintained. At each level there are both deliberative and reflexive elements. Tasks are deliberately decomposed into plans for 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

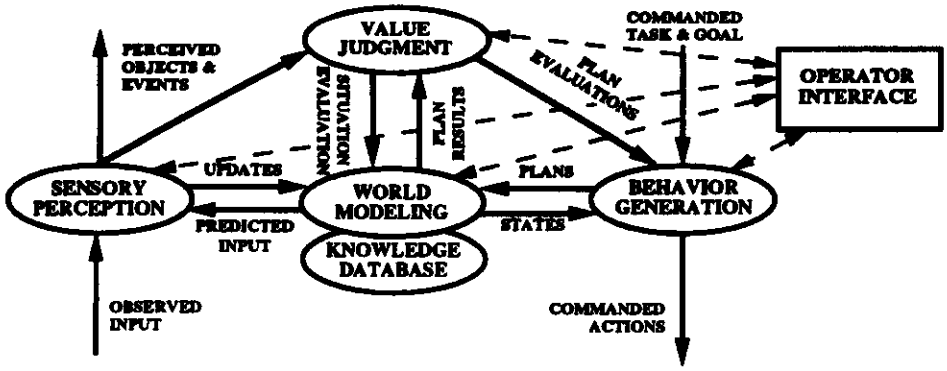

Figure 2. A typical node in the RCS architecture.

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: Behaviour Generating (BG), World Modelling (WM), Sensory Processing (SP), and Value Judgement (VJ); plus a Knowledge Database (KD) and Operator Interface module. These are shown in Figure 2. The nodes are serviced by a Communications system that communicates information between nodes, modules, and submodules.

3. Behaviour Generation (BG) modules

Each BG module contains a job assignor (JA), a set of schedulers (SC), a plan selector (PS), and a set of executor (EX) functions (or submodules) as shown in Figure 3. The JA, SC, and PS submodules perform deliberative planning functions. The EX submodules perform reactive plan execution and servoing functions.

Within the BG modules, each executor together with its supporting scheduler (plus associated world modelling, knowledge database, and sensory perception modules) acts as an agent. The set of executors and schedulers in the BG module make up a set of cooperating agents. Tasks arriving at the input of the BG module are decomposed into job assignments for the set of agents. The job assigned to each agent is further decomposed into a series of subtasks by the corresponding scheduler (possibly in cooperation with schedulers of other agents). Together the job assignor and schedulers function so as to generate one or more tentative plans for accomplishing the task. Several plans may be able to achieve the task goal. The plan selector selects the best of the possible plans for execution and places it in a coordinated agents plan buffer. Each Executor then executes the piece of the selected plan prepared for it by its own scheduler.

The JA, SC, and PS functions are planning functions that deliberatively select or generate a plan to achieve the task goal. The EX functions are reactive functions that reflexively respond to current sensory feedback.

3.1. *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 algorithms for multi-agent cooperating or competitive groups.

Plans may be selected or synthesized by the job assignor (JA) and scheduler (SC) submodules. JA functions distribute jobs and resources to agents, and transform coordinate systems from task to subtask coordinates (e.g. from end-point, or tool, coordinates to joint actuator coordinates). SC functions compute a temporal schedule of subtasks for each agent and coordinate schedules between cooperating agents (e.g. coordinate joint actuator trajectories 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 a plan.

Tentative plans are sent to the world model (shown in Figures 2 and 4) where expected results are simulated. Expected results are sent to the value judgement (VJ) module for cost/benefit evaluation, and evaluations are returned to the plan section (PS) submodule for a decision as to the best plan of action. By iteration through this planning loop, the space of possible plans can be searched, with the PS function selecting for execution the coordinated agents plan receiving the best VJ evaluation.

RCS can accommodate a variety of planning algorithms from case-based plan selection from a library of pre-computed scripts, to an A* search of configuration space. Plans may be computed off-line long before execution, or in real-time as execution is proceeding. Anytime planning algorithms can be accommodated.

In all cases, plans must be synthesized prior to execution. 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

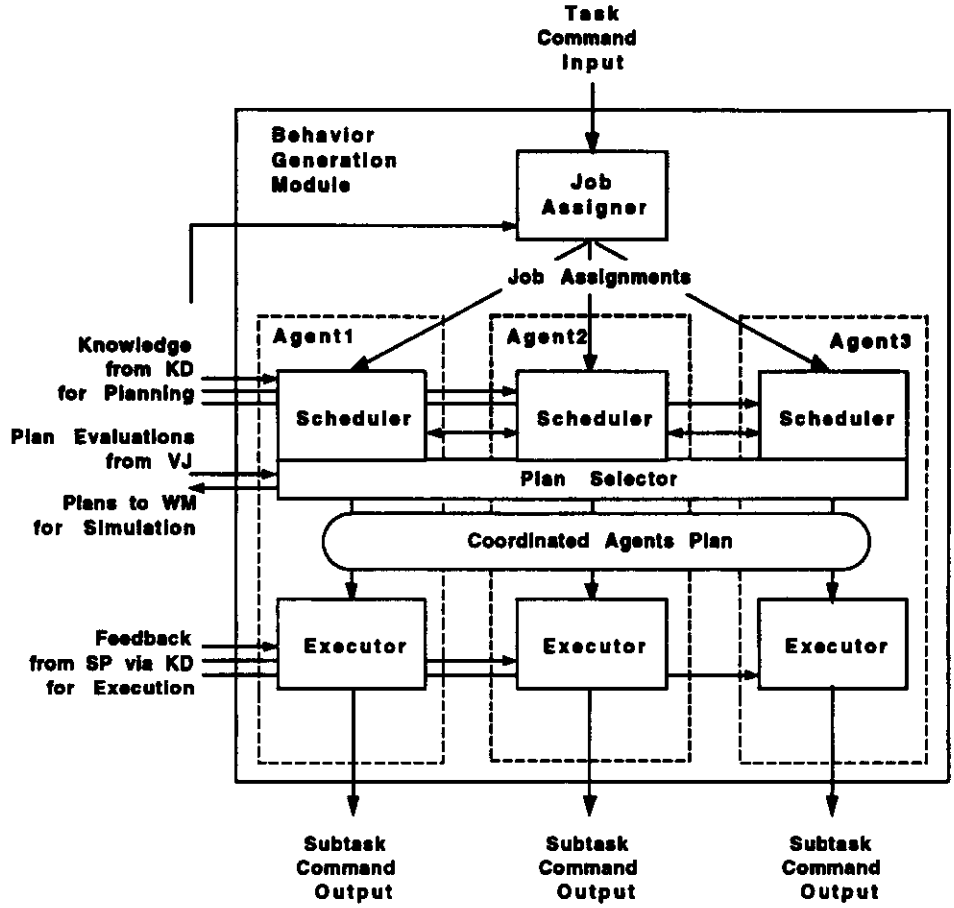

Figure 3. Behaviour generation (BG) module showing a job assignor that allocates jobs and resources to three agents, each consisting of a scheduler and an executor. A plan selector selects the best of several alternative plans generated by the job assignor and schedulers for execution by the executors. The executors with their supporting schedulers, world modelling, knowledge database, and sensory perception modules comprise agents.

mode computing environment, once a day, or once a week. However, this produces plans that often become obsolete shortly after execution begins. As the uncertainty in

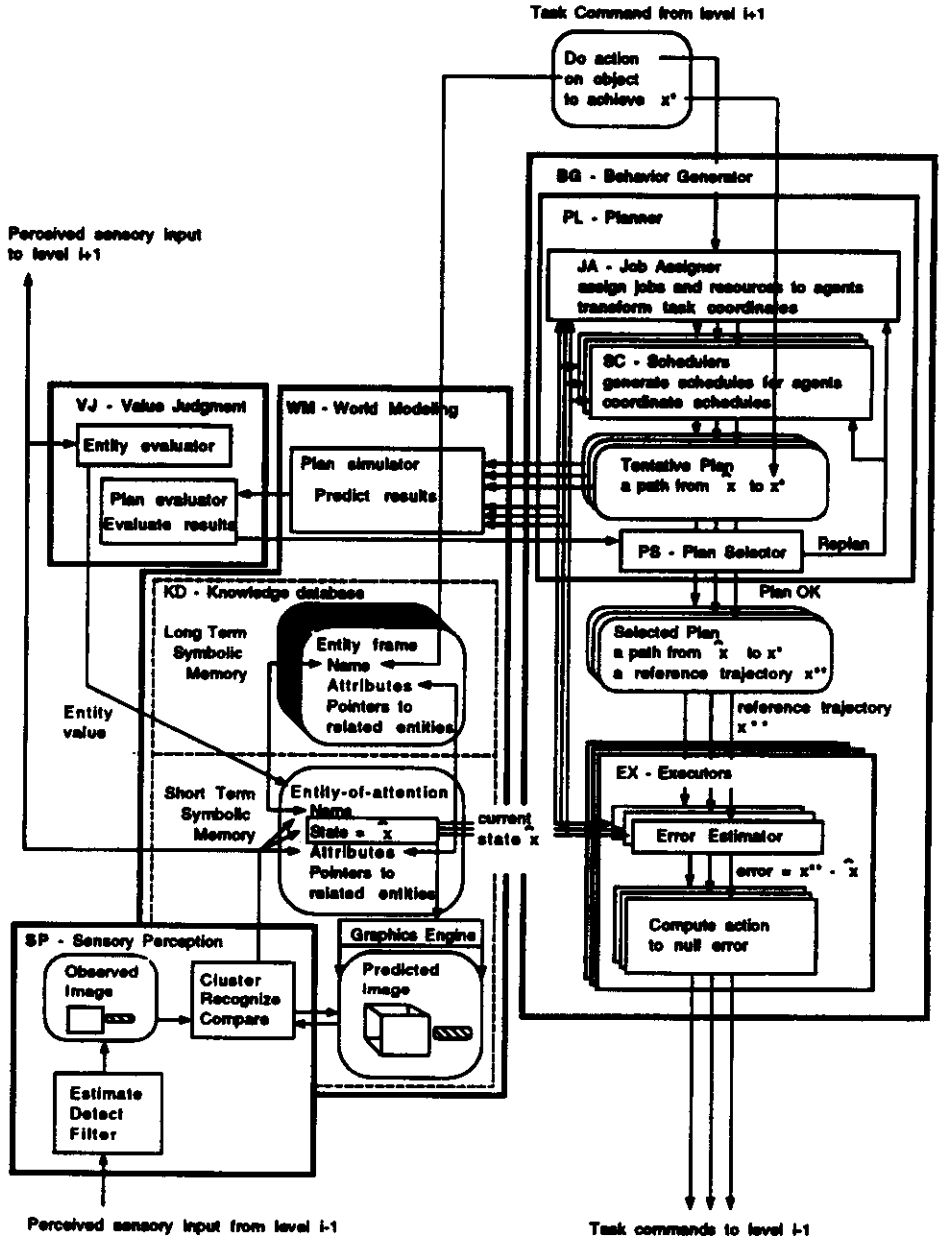

Figure 4. Relationships within a single node of the RCS architecture. The behaviour generating (BG) modules contain job assignor (JA), scheduler (SC), plan selector (PS) and executor (EX) submodules. The world modelling (WM) module contains a plan simulator and mechanisms for updating the knowledge database (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

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.

For uncertain environments, plans must be recomputed frequently, either on demand, or at repetitive cyclical intervals. The response time 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, it is best if the planner generates a new plan before the executor completes the first step or two in the current plan.

At each RCS level, plans are computed to a given planning horizon. The length of the planning horizon is a distinguishing characteristic of a RCS level (see Figure 1). The resolution of plans is such that at each level, plans contain on the order of ten sequential subtasks for each agent. On average, planning horizons shrink by about an order of magnitude at each lower level, and the number of subtasks to the horizon remains constant. Thus, the temporal resolution of subtasks increases about an order of magnitude at each lower level. Planning horizons at high levels may span months or years, while the planning horizon at the bottom level typically is 30 milliseconds or less. Therefore, 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.

At each hierarchical level, plans are expressed in a vocabulary of task commands that can be accepted as input by the executor (EX) submodules at that level.

3.2. *Execution*

For each agent at each level, there is an executor (EX) function (or submodule). The EX function compares the desired subgoal of the current plan subtask with the current estimated state of the world from the world model knowledge database (KD). The EX function then executes a control law designed to reduce the difference between the subgoal of the current plan subtask and the current estimated state of the world. The set of schedules of subtasks in the plan corresponds to a set of reference trajectories for the corresponding EX submodules. The estimated state of the world in the KD corresponds to a feedback signal. Each EX module thus functions as a servo-mechanism, steering its agent to follow the reference trajectory synthesized by the JA and SC submodules and selected by the PS submodule.

The repetition rate of the EX cycle corresponds to the bandwidth of the control loop 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 a 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.

---

by the WM module with perceived input from sensors. It has algorithms for recognizing entities and clustering entities into higher level entities. The value judgement (VJ) module evaluates plans and places values on entities recognized in the observed sensory input.

The EX functions are reactive, or reflexive, elements that generate sensory interactive behaviour. They are concerned with the immediate moment. The planning functions performed by job assignment JA, scheduling SC, WM simulation, VJ evaluations, and plan selection PS functions are deliberative elements. They are preoccupied with the future. 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 reflexive execution elements predominate and the planning elements are relatively simple. At higher levels, the reverse is true – deliberative planning elements consume most of the computing resources, and reflexive execution elements decrease in relative importance.

4. World Modelling (WM) modules

The WM modules perform four basic functions:

(a) WM functions maintain the knowledge database (KD), keeping it current and consistent. They update state estimates in the knowledge database based on correlations and variance between world model predictions and sensory observations at each node. 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.

(b) WM functions 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.

(c) WM functions respond to queries from the BG molecules 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.

(d) 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.

5. Sensory Processing (SP) modules

SP modules process data from visual, auditory, tactile, proprioceptive, 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 (shown in Figure 4) 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, colour, and range discontinuities, optical flow, and stereo disparity. They may utilize a variety of signal detection and pattern recognition algorithms to analyse scenes, compute attributes of entities, and provide the information needed for manipulation, locomotion, communication, attention tracking, and spatial–temporal reasoning.

6. 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. For example, VJ modules may compute whether an object or event is worthy of attention, or of storage in long-term memory. VJ modules may also 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.

7. Knowledge Database (KD) modules

KD modules store the data that support the BG, WM, SP, and VJ processing modules in each node. KD modules consist of data structures that contain state variables. Types of data structures include scalars, vectors, matrices, iconic image arrays, and symbolic characters, strings, lists, frames, and graphs. Information in the KD includes knowledge about entities and events, and about how the world behaves, both logically and dynamically. The KD contains both short-term (dynamic) and long-term (static) memory elements.

Short-term memory consists of both symbolic and iconic representations:

(a) Short-term symbolic representations consist of current entities-of-attention that have either been specified by the current task, or are particularly noteworthy entities observed in current sensory input. Entities are represented by symbolic lists that contain entity attributes and pointers. Attributes describe properties of the entities. The pointers define relationships between entities, and indicate correspondence between current entities-of-attention and long-term symbolic entities stored in long-term memory.

(b) Short-term iconic representations can consist of attribute images generated directly, or by recursive estimation, from sensory observations. Short-term iconic images can also be generated by internal imagination from short-term symbolic representations. Short-term iconic images can be used to mask or window incoming data, or to fuse incoming sensory observations with internally imagined images generated from current entities-of-attention. Short-term iconic images persist in memory only so long as they are refreshed by incoming sensory data or by internally generated images.

Long-term memory consists entirely of symbolic data structures. Long-term symbolic representations include the entire dictionary of entities that the intelligent system knows about. Attributes from long-term symbolic entity 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 symbolic entity can be added to the attribute list of the short-term entity-of-attention. Alternatively, if a match is recognized between a current entity-of-attention and a long-term symbolic 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 typically implemented in a distributed fashion, with representations at each node corresponding to the requirements of those task decomposition and sensory processing functions being carried out in each node.

8. Communication system

The communication system provides a network of pathways that transmit messages between processing and database modules. The communications system conveys commands from a BG module to its subordinates, and returns status. It conveys tentative plans from the BG planner to the WM simulator. It transmits simulation results to the VJ evaluator, and plan evaluations to the BG plan selector. It moves sensory data from sensors to filters, transfers WM predictions to SP comparators. It sets windows on SP spatial integrators and thresholds on 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 communication system enables the various processing and data modules in the RCS architecture to act as 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. RCS requires that any particular state variable have only one functional source, or writer, but it may have many destinations, or readers.

9. Comparison of RCS with subsumption

The subsumption architecture is a layered collection of behaviours, or behaviour generating modules, that tightly connect perception to action. In control terms, this simply means that each behaviour defines a control loop, with a control function that maps sensory feedback into actuator output. The subsumption behaviour modules are implemented as finite state automata, so that the system maintains a first-order Markov type of memory of past history, and this memory can be used to select among a variety of possible behaviours, or control functions. There can be message passing, suppression and inhibition between processes within a behaviour, or between behaviours. When a behaviour inhibits another, and substitutes itself instead, the inhibitor can be said to have subsumed the inhibited (Brooks 1990).

In RCS, the EX submodules are essentially identical to the behaviour modules in the subsumption architecture. In RCS, plans can be expressed as state graphs (or state tables), and EX submodules function as finite state automata that execute the state tables. A substantial difference between RCS and the subsumption architecture lies in the manner in which a choice is made between alternative behaviours. In RCS, the choice is made by the plan selection (PS) submodule based on plan simulation by the WM module, and evaluation of predicted results by the VJ module prior to execution. In subsumption, the choice is made between competing behaviours at execution time by the subsumption mechanism of inhibition and substitution. Thus, although the subsumption architecture behaviour modules perform essentially the same function as the RCS planning submodules, they do so without the mechanism of forethought.

The principal difference between RCS and subsumption lies not in the execution or even the planning of behaviour, but in the representation of knowledge in the world

model. RCS develops rich and sophisticated representations of the world in the knowledge database that are capable of supporting cognitive reasoning and imagination. RCS also supports sophisticated sensory perception mechanisms that are capable of keeping the world model current and up-to-date and in spatial and temporal correspondence with the state of the real world. Subsumption denies the need for complex world models or sophisticated internal representations. It is fundamentally reactive, not deliberative, in its methodology.

10. Implementation mechanisms

Each submodule in the RCS system is implemented as a cyclically executing process. RCS processes typically are triggered by a clock at a fixed repetition rate, but may be triggered by events. Each process consists of an augmented finite-state machine surrounded by a set of input and output data buffers. At the beginning of each cycle, the submodule reads from its input buffers and processes the inputs into a form suitable for a state-table that encodes a set of state-dependent production rules of a form that are common in expert systems. The processed input is compared with the rules in the stable-table. The rule that matches causes the process to go to a next state, possibly execute a procedure, and compute an appropriate output. Each submodule also computes a set of diagnostic functions, such as the time required for the process to complete, the maximum time the process has taken, and the average time taken. Each submodule has an interface for an operator that provides the operator the ability to halt, single-step, and/or display the value of any variable and the state of any process at any time during execution. A process may be halted, parameters examined by a programmer, variables changed, and execution resumed. Communications between processes in the RCS system are designed so that all processes cycle independently and run completely asynchronously.

Programming tools and software templates have been built that provide the system developer an easy way to configure an RCS system. The template automatically generates all the required utilities and diagnostic features. Software templates are implemented in C++. A graphical design tool enables a programmer to define a RCS submodule with the click of a mouse, and interconnect submodules by click and draw techniques. These programming tools have enabled RCS systems with hundreds of submodules to be built in a few months.

RCS has been implemented on a variety of platforms, including Sun workstations, 486 PC computers, VME systems, and MacIntosh machines using a number of different operating systems, including Forth, pSOS, DOS, VxWorks, and Lynx. The overhead for a RCS template running on a 486 class machine is about five microseconds. The cycle time for a typical low-level RCS submodule is 30 milliseconds.

11. Applications

The RCS architecture has been used in the implementation of a number of experimental projects. These include:

11.1. *A horizontal machining workstation*

This RCS project was part of the NBS Automated Manufacturing Research Facility (AMRF) (Albus *et al.* 1982). 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 behaviours 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 (Wavering and Fiala 1987).

11.2. *A cleaning and deburring workstation*

This RCS 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 vice, and turnings 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 (Murphy *et al.* 1988).

11.3. *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 grind precision chamfers on complex parts manufactured from hard materials such as aircraft jet engine components. The workstation consists of a grinding tool mounted on a micro-positioner with computer controlled force and stiffness parameters, integrated with a six-degree-of-freedom robot, and an indexing table for part fixturing. 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. Under a cooperative research and development agreement, a prototype production cell is currently being tested at Pratt & Whitney's East Hartford, CT site (Stouffer *et al.* 1993).

11.4. *NBS/NASA standard reference model architecture for the Space Station Telerobotic Servicer (NASREM)*

This project was funded by the NASA Goddard Space Flight Center. NASREM was used by Martin Marietta to develop the control system for the space station telerobotic servicer. Algorithms were developed for force servoing, impedance control, and real-time image processing of telerobotic systems at NIST, Martin Marietta, Lockheed, Goddard, and in a number of university and industry labs in the United States and Europe (Albus *et al.* 1987).

11.5. *Coal mining automation*

This project transferred RCS architecture and methodology to a team of researchers in the US 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 (Huang *et al.* 1991).

11.6. *A nuclear submarine manoeuvring system*

This ARPA sponsored RCS 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 maneuvering 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 (Albus 1975) 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 a lubrication oil fire by reconfiguring ventilation systems, rising in depth to snorkel level, and engaging the diesel engines for emergency propulsion (Huang *et al.* 1993).

11.7. *A US Postal Service automated stamp distribution centre*

This RCS 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 centre 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.

11.8. *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. Tests were conducted in Lake Winnipasaki during the fall of 1987 (Herman and Albus 1988).

11.9. *An unmanned ground vehicle*

Two versions of RCS have been implemented on an Army HMMWV light truck. One version 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 (Szabo *et al.* 1990).

A second version 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. A world model estimate of the lane markings is compared to observed edges in the image, and a new estimate is computed every 15 milliseconds, with pipeline latency of less than 150 milliseconds. 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 (Schneiderman and Nashman 1994).

11.10. *Planning and control 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 (Cleveland and Meystel 1990).

11.11. *An autonomous mobile vehicle*
An autonomous vehicle was assembled and tested by Drexel University over the period 1984–1987. The goal of the effort was to investigate the RCS architecture with four levels of resolution 'Planer–Navigator–Pilot' on top of the lower level control of steering and propulsion. The results of this research are described in Meystel (1991).

11.12. *An open architecture enhanced 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 (Proctor and Michaloski 1993). The EMC is a testbed for evaluating open architecture interface specifications. 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) programme, EMC functional modules have been defined, and application programming interfaces (APIs) are being specified for sending messages between the functional modules. A prototype EMC has been installed and is being evaluated in the General Motors Powertrain prototype production facility in Pontiac Michigan as part of a DoE–TEAM/NIST–EMC government/industry consortium. The goal of this effort is to develop API standards for open architecture controllers.

The EMC implemented at the GM Powertrain plant is on a four-axis horizontal matching centre with a tool changer and pallet shuttle system. It contains a commercial motion control board which closes the control loop on the X, Y, Z axes every 300 microseconds. At this rate, the output commands to the motor drives of the machine tool are indistinguishable from continuous control signals.

Higher-level nodes in the EMC controller have a control cycle that runs every 20 milliseconds. These nodes provide input to the trajectory generator and spindle controller on the motion control board, as well as continuous motion output to the B axis motor drive, and discrete control signals to the tool changer, pallet shuttle, and miscellaneous actuators. The machining centre has more than 100 discrete input/output points which must be sequenced precisely in order to effect proper loading and unloading of tools and materials. For these processes, 20 milliseconds is short compared to the dynamics of the system being controlled. Complete tool path motions, tool change operations, and pallet shuttle operations require many seconds to complete. At higher levels, events occur even less frequently. Machining tasks may take a number of minutes to execute. Therefore, to the machine, it appears as if the EMC is a continuous controller.

Yet the EMC can switch control modes on any 20 millisecond cycle boundary.

During each computation cycle, the controller examines the input, matches it with the state-transition conditions in a state table, and if a match occurs, the system switches to a new state. In principle, each state could implement a different control algorithm. An operator can interrupt the system, switch control modes from automatic to manual, or give a new feed rate override command every 20 milliseconds. The RCS system is thus a hybrid controller in which all events, both continuous and discrete, are handled at rates that make the distinction between discrete events and continuous processes essentially disappear.

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 implementation 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 (Senehi *et al.* 1994). When complete, this joint architecture will define a reference model architecture for manufacturing 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 (Quintero and Barbera 1993). Commercial versions of the EMC are expected to appear on the market within a few months.

12. Summary 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. RCS modules are arranged so as to process sensory feedback through a variety of filters and integrators with different time intervals so as to create servo-loop bandwidths at a hierarchy of levels. All the functional modules in RCS are designed as independent concurrently executing analogue, or cyclically sampled 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. Just as neurons continuously monitor their synaptic inputs and compute axonal outputs, so RCS modules sample their inputs and compute their outputs on a clock cycle that is short compared to the bandwidth of the process being controlled.

RCS contains both deliberative and reactive elements at each level. This makes it ideal as a research platform for comparing and contrasting research results from minimalist (emergent and reactive) systems with results from deliberative (intentional and planning) systems. The current RCS architecture is a canonical system that can serve as a framework for integrating concepts from a variety of fields such as intelligent control, artificial intelligence, neural nets, machine vision, robotics, computer science,

operations research, game theory, signal processing, filtering, and communications theory. The goal is to provide a reference model architecture for the design, engineering, and testing of intelligent systems.

References

Albus, J. S. (1975) New approach to manipulator control: the cerebellar model articulation controller (CMAC), and Data storage in the cerebellar model articulation controller (CMAC), *Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control*, September: 220–233.

Albus, J. S. (1981) *Brains, Behavior, and Robotics* Byte (Peterborough, NH: McGraw-Hill).

Albus, J. S. (1991) Outline for a theory of intelligence, *IEEE Transactions on Systems, Man and Cybernetics*, **21**: 473–509.

Albus, J. S. (1993) A reference model architecture for intelligent systems design. In P. J. Antsaklis and K. M. Passino (eds). *An Introduction to Intelligent and Autonomous Control* (Boston, MA: Kluwer).

Albus, J. S., McLean, C. R., Barbera, A. J. and Fitzgerald, M. L. (1982) Architecture for real-time sensory-interactive control of robots in a manufacturing facility. In *Proceedings of the Fourth IFAC/IFIP Symposium–Information Control Problems in Manufacturing Technology*, Gaithersburg, MD, 26–29 October.

Albus, J. S., McCain, H. G. and Lumia, R. 'NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM). NISTTN 1235, 1989 edn, National Institute of Standards and Technology, Gaithersburg, MD, April 1989 (supersedes NBS Technical Note 1235, July 1987).

Brooks, R. A. (1990) Elephants don't play chess, *Robotics and Autonomous Systems*, **6**: 3–15.

Cleveland, B. and Meystel, A. (1990) Predictive planing + fuzzy compensation = intelligent control. In *Proceedings of the 5th IEEE International Symposium on Intelligent Control*, Philadelphia, PA, September.

Flanagan, O. (1991) *The Science of the Mind*, (Cambridge, MA: MIT Press).

Herman, M. and Albus, J. S. (1988) Overview of the multiple autonomous underwater vehicles (MAUV) project. *1988 IEEE International Conference on Robotics and Automation*, Philadelphia, PA, April.

Huang, H. M., Quintero, R. and Albus, J. S. (1991) A reference model, design approach, and development illustration toward hierarchical real-time system control for coal mining operations. In *Manufacturing and automation systems: techniques and technologies* (San Diego, CA: Academic Press) pp. 173–254.

Huang, H. M., Hira, R. and Quintero, R. (1993) A submarine maneuvering system demonstration based on the NIST real-time control system reference model. In *Proceedings of the 8th IEEE International Symposium on Intelligent Control*, Chicago, IL, 24–27 August.

Meystel, A. (1991) *Autonomous Mobile Robots: Vehicles with Cognitive Control* (Singapore: World Scientific).

Murphy, K. N., Norcross, R. J. and Proctor, F. M. (1988) CAD directed robotic deburring. In *Proceedings of the Second International Symposium on Robotics and Manufacturing Research, Education, and Applications*, Albuquerque, NM, 16–18 November.

Proctor, F. and Michaloski, J. (1993) Enhanced machine controller architecture overview. NISTIR 5331, National Institute of Standards and Technology, Gaithersburg, MD, December.

Quintero, R. and Barbera, A. J. (1993) A software template approach to building complex large-scale intelligent control systems. In *Proceedings of the 8th IEEE International Symposium on Intelligent Control*, Chicago, IL, 25–27 September.

Senehi, M. K., Kramer, T. J., Michaloski, J., Quintero, R., Ray, S. R., Rippey, W. G. and Wallace, S. (1994) Reference architecture for machine control systems integration: interim report. NISTIR 5517, National Institute of Standards and Technology, Gaithersburg, MD.

Schneiderman, H. and Nashman, M. (1994) Visual tracking for autonomous driving, *IEEE Transactions on Robotics and Automation*, **10**: 769–775.

Skinner, B. F. (1953) *Science and Human Behavior* (New York: Buntam).

Stouffer, K., Michaloski, J., Russell, R. and Proctor, F. (1995) ADACS—an automated system for part finishing. NISTIR 5171, National Institute of Standards and Technology, Gaithersburg, MD, April 1993, and *Proceedings of the IECON '93 International Conference on Industrial Electronics, Control and Instrumentation*, Maui, Hawaii, 15–19 November.

Szabo, S., Scott, H. A., Murphy, K. N. and Legowik, S. A. (1990) Control system architecture for a remotely operated unmanned land vehicle. In *Proceedings of the 5th IEEE International Symposium on Intelligent Control*, Philadelphia, PA, September 1990.

Wavering, A. J. and Fiala, J. C. (1987) Real-time control system of the horizontal workstation robot. NBSIR 88-3692, National Institute of Standards and Technology, Gaithersburg, MD, December.