

# RCS: A cognitive architecture for intelligent multi-agent systems<sup>☆</sup>

James S. Albus<sup>\*</sup>, Anthony J. Barbera

*Intelligent Systems Division, Manufacturing Engineering Laboratory, National Institute of Standards and Technology,  
Building 220, Room B124, Stop 8230, Gaithersburg, MD 20899, USA*

Received 19 August 2004; received in revised form 2 December 2004; accepted 4 December 2004

## Abstract

Real-time control system (RCS) is a cognitive architecture designed to enable any level of intelligent behavior, up to and including human levels of performance. RCS consists of a multi-layered multi-resolutional hierarchy of computational agents each containing elements of sensory processing (SP), world modeling (WM), value judgment (VJ), behavior generation (BG), and a knowledge database (KD). At the lower levels, these agents generate goal-seeking reactive behavior. At higher levels, they enable decision-making, planning, and deliberative behavior. At all levels, the architecture supports signal grounding and abductive inference.

A methodology is described for designing RCS systems, and experimental results are presented from a series of performance tests designed to measure technology readiness.

© 2005 Elsevier Ltd. All rights reserved.

**Keywords:** Cognitive architectures; Unmanned vehicles; Intelligent machines; Knowledge-based control; Knowledge representation; Recursive estimation; Systems engineering; Image processing

## 1. Introduction

Interest in cognitive models and intelligent systems has grown rapidly over the past two decades as a result of a confluence of three important events:

### 1.1. The emergence of a computational theory of intelligence

A fully developed scientific theory of intelligence does not yet exist, but an understanding of how to build intelligent systems is developing faster than most people appreciate. Progress is rapid in many different fields. Recent results from a number of different disciplines, including the neurosciences, cognitive psychology, artificial intelligence, robotics, and intelligent machines, have laid the foundations for a computational theory of intelligence (Newell & Simon,

1972; Anderson, 1983; Laird, Newell, & Rosenbloom, 1987; Albus, 1991; Albus & Meystel, 2001).

### 1.2. The continued exponential growth in computing power

The estimated computational power of the human brain is already rivaled by existing supercomputers. Within the next quarter century, computational power approaching that of the human brain can be expected from a small network of desktop machines (Kurzweil, 1999). This means that serious attempts can be made to model the functional capabilities of the brain in perception, cognition, and behavioral skills. Of course, having the computational power is only part of the challenge, the rest is making proper use of it.

### 1.3. Growth in user interest for military and commercial applications

Potential applications in both civilian and military systems have begun to emerge. In the United States, military interest in unmanned vehicles (air, ground, and sea) has grown rapidly as autonomous vehicle capabilities have

<sup>☆</sup> This paper is a product of U.S. Government employees in the course of their assigned duties, and therefore not subject to copyright.

<sup>\*</sup> Corresponding author.

E-mail addresses: james.albus@nist.gov (J.S. Albus), anthony.barbera@nist.gov (A.J. Barbera).

been demonstrated that far exceed previous expectations (Shoemaker, Bornstein, Myers, & Brendle, 1999). In Japan, Europe, and the U.S., automotive companies are actively pursuing commercial applications of adaptive cruise control, crash warning systems, and collision avoidance technology (Bishop, 2003). The eventual result may be the intelligent automotive autopilot—or even the robot chauffeur.

In subsequent sections, we will discuss the characteristics of cognitive architectures in Section 2, the RCS reference model in Section 3, some of the theoretical problems of computational models in Section 4, the RCS methodology for system design in Section 5, some experimental results in Section 6, and future prospects in Section 7.

## 2. Cognitive architectures

A cognitive architecture can be defined as the organizational structure of functional processes and knowledge representations that enable the modeling of cognitive phenomena. Over the past half-century, several cognitive architectures have been developed. One of the earliest was the ACT architecture (Anderson, 1983). ACT grew out of research on human memory. Over the years, ACT evolved into ACT\* and more recently, ACT-R. ACT-R is being used in several research projects in an Advanced Decision Architectures Collaborative Technology Alliance for the U.S. Army (Gonzalez, 2003). ACT-R is also being used by thousands of schools across the country as an algebra tutor—an instructional system that supports learning-by-doing.

Another well-known and widely used architecture is Soar. Soar grew out of research on human problem solving, and has been used for many academic and military research projects in problem solving, language understanding, computational linguistics, theorem proving, and cognitive modeling (Newell & Simon, 1972; Laird et al., 1987). A recent commercial version of Soar (Tac-Air Soar<sup>1</sup>) has been developed to address a number of simulation and training problems for the U.S. Air Force (SoarTech, 2004).

Other cognitive architectures include Prodigy, ICARUS, IMPRINT, EPIC, Pilot's Associate, and RCS. Like Soar, Prodigy uses search through a problem space to achieve goals cast as first-order expressions (Minton, 1990). ICARUS is a reactive architecture that encodes knowledge as reactive skills (Shapiro & Langley, 1999). IMPRINT is a task description language designed for the Army to capture the procedural specification of tactical behavior scenarios. It contains a dynamic, stochastic, discrete-event network modeling tool designed to help assess the interaction of soldier and system performance throughout the system lifecycle—from concept and design through field testing and system upgrades. IMPRINT has been integrated with ACT-R to model military behaviors (Archer, Lebiere, Warwick, &

Schunk, 2003). EPIC is an architecture that models the detailed timing of human perceptual, cognitive, and motor activity, including the input/output characteristics of the nervous system connecting the higher level cognitive functions to the external world (Kieras & Meyer, 1997). Pilot's Associate is a rule-based system developed to provide cognitive support for military pilots in combat. It is currently being applied to the DARPA/Air Force/Navy Joint Unmanned Combat Air Systems (J-UCAS). RCS is a control system architecture inspired by a theory of cerebellar function (Albus, 1971). RCS models the brain as a hierarchy of goal-directed sensory-interactive intelligent control processes that theoretically could be implemented by neural nets, finite state automata, or production rules (Albus, 1981).

RCS is similar to other cognitive architectures in that it represents procedural knowledge in terms of production rules, and represents declarative knowledge in abstract data structures such as frames, classes, and semantic nets. RCS differs from other cognitive architectures in that it also includes signals, images, and maps in its knowledge database, and maintains a tight real-time coupling between iconic and symbolic data structures in its world model. RCS is also different in: (a) its focus on task decomposition as the fundamental organizing principle; (b) its level of specificity in the assignment of duties and responsibilities to agents and units in the behavior generating hierarchy; and (c) its emphasis on controlling real machines in real-world environments.

RCS evolved from the bottom up as a real-time intelligent control system for real machines operating on real objects in the real world. The first version of RCS was developed as a sensory-interactive goal-directed controller for a laboratory robot (Barbera, Albus, & Fitzgerald, 1979). Over the years, RCS has evolved into an intelligent controller for industrial robots, machine tools, intelligent manufacturing systems, automated general mail facilities, automated stamp distribution systems, automated mining equipment, unmanned underwater vehicles, and unmanned ground vehicles (Barbera, Fitzgerald, Albus, & Haynes, 1984; Albus, 1997). Throughout its development, all symbols in the RCS world model have been grounded to objects and states in the real world.

The most recent version of RCS (4D/RCS) embeds elements of Dickmanns (1992, 1999) 4-D approach to machine vision within the RCS control architecture. 4D/RCS was designed for the U.S. Army Research Lab AUTONAV and Demo III Experimental Unmanned Vehicle programs and has been adopted by the Army Future Combat System program for Autonomous Navigation Systems (Albus & Meystel, 2001; Albus et al., 2002).

## 3. A reference model architecture

A reference model architecture describes the functions, entities, events, relationships, and information flow that

<sup>1</sup> The name of commercial products or vendors does not imply NIST endorsement or that this product is necessarily the best for the purpose.

takes place within and between functional modules. A reference model provides a framework for the specification of functional requirements, the design of software to meet those requirements, and the testing of components and systems. A block diagram of a 4D/RCS reference model architecture is shown in Fig. 1. Each node in the architecture represents an operational unit in an organizational hierarchy. Each node contains a behavior generation (BG), world modeling (WM), sensory processing (SP), and value judgment (VJ) process together with a knowledge database (KD) (included in WM in Fig. 1). Processing nodes are organized such that the BG processes form a command tree. Information in the knowledge database is shared between WM processes in nodes above, below, and at the same level within the same subtree. On the right, are examples of the functional characteristics of the BG processes at each level. On the left, are examples of the scale of maps generated by the SP processes and populated by the WM in the KD at each level. Sensory data paths flowing up the SP hierarchy typically form a graph, not a tree. VJ processes are hidden behind WM processes in the diagram. A control loop may be closed at every node. An operator interface may provide input to, and obtain output from, processes in every node. (Numerical values are representative examples only. Actual numbers depend on parameters of the specific design.)

Fig. 2 shows a first level of detail in 4D/RCS nodes. The internal structure of the BG process consists of a planner and a set of executors (EX). A WM process maintains a KD that is the BG unit's best estimate of the external world. A SP

process operates on input from sensors by windowing (i.e., focusing attention), grouping, computing attributes, filtering, and classifying (e.g., recognizing) entities, events, and situations. A VJ process evaluates expected results of tentative plans, and assigns confidence and worth to entities, events, and situations represented in the KD.

Each node contains both a deliberative and a reactive component. Bottom-up, each node closes a reactive control loop driven by feedback from sensors. Top-down, each node generates and executes plans designed to satisfy task goals, priorities, and constraints conveyed by commands from above. Within each node, deliberative plans are merged with reactive behaviors (Albus et al., 2002).

Each BG process accepts tasks and plans and executes behavior designed to accomplish those tasks. Task command input comes from a higher-level BG process in the 4D/RCS hierarchy. Each input task command is decomposed into a plan consisting of subtasks for subordinate BG processes.

As shown in the upper right of Fig. 2, task commands from a supervisor BG process are input. A planner module decomposes each task into a set of coordinated plans for subordinate BG processes. For each subordinate there is an Executor that issues commands, monitors progress, and compensates for errors between desired plans and observed results. The Executors use feedback to react quickly to emergency conditions with reflexive actions. Predictive capabilities provided by the WM may enable the Executors to generate pre-emptive behavior.

Plans may be generated by any of a great variety of planning algorithms, e.g., case-based reasoning, search-

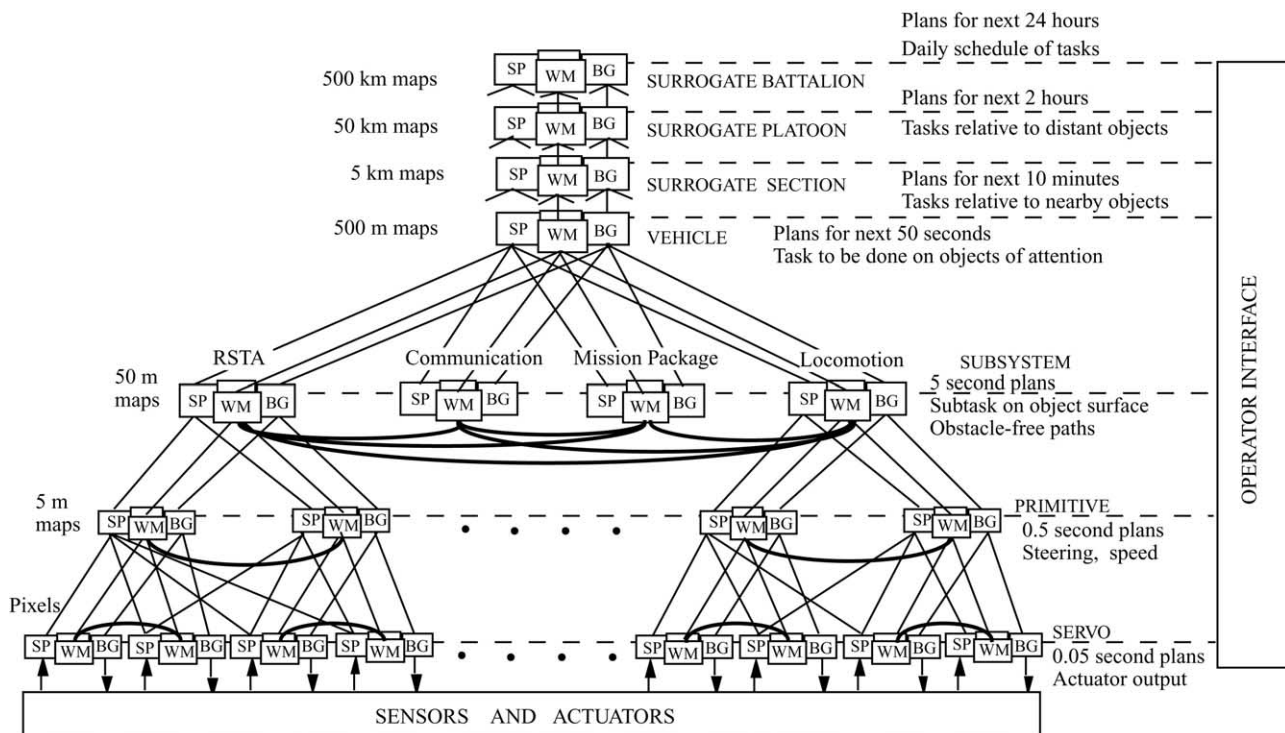


Fig. 1. A 4D/RCS reference model architecture for an autonomous military scout vehicle.

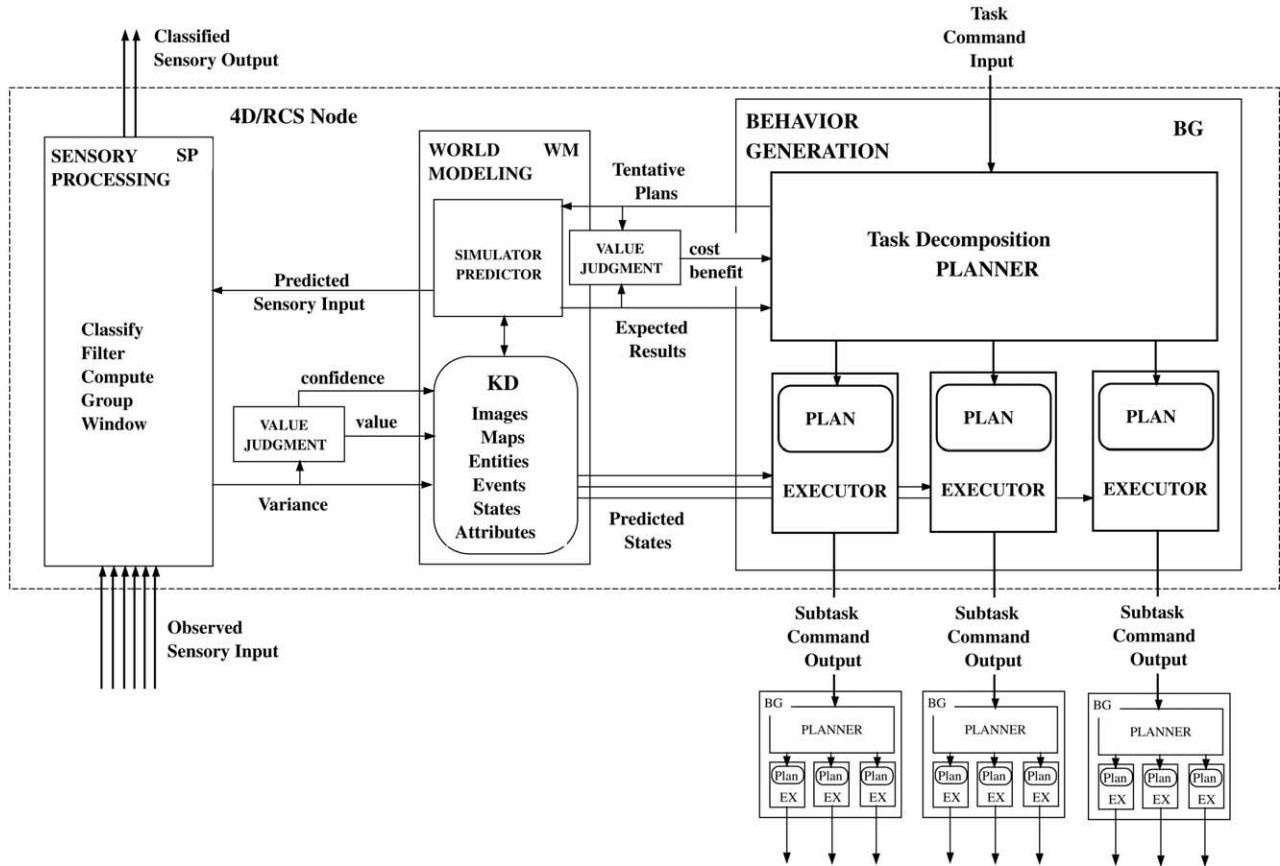


Fig. 2. A typical 4D/RCS computational node.

based optimization, or schema-based scripting. Whatever the methodology, alternative possible futures are subjected to the following procedure:

While the planning period is open

```
{
  (1) the BG planner hypothesizes a tentative_plan;
  (2) the WM predicts the probable_result of the tentative_plan;
  (3) the VJ evaluates the probable_result_value;
  (4) a plan selector within the BG planner checks to see if the proba-
  ble_result_value is greater than the previous probable_result_value of the
  plan already in the current_best_plan_buffer,
  {
    if it is, then the tentative_plan replaces the current plan in the
    current_best_plan_buffer;
    else continue;
  }
};
```

On the next execution clock cycle,

- (1) Move the contents of the current\_best\_plan\_buffer into the executor\_plan\_buffers;
- (2) Begin re-planning immediately, or wait until the next planning cycle triggers;

This entire process occurs concurrently at each level of the 4D/RCS hierarchy at a rate designed to maintain a

current\_best\_plan out to the spatial/temporal planning horizon imposed at that level.

The executor\_plan\_buffer forms an interface between the planner and executor processes in each node. This is the interface between deliberative and reactive processes. The executor\_plan\_buffer enables the planning process to run asynchronously and at a slower repetition rate than the execution cycle of the reactive control loop. As soon as new plans are developed, they are loaded into the executor\_plan\_buffers. The planner is free to run on its own so long as the executor\_plan\_buffers are kept supplied with a current\_best\_plan. In the 4D/RCS architecture this interface between planners and executors exists within every node in the computational hierarchy. Thus, the interface between deliberative and reactive processes is not localized to a particular level, but is distributed throughout the 4D/RCS architecture.

SP and WM processes interact to support windowing (i.e., attention), grouping (i.e., segmentation), recursive estimation (e.g., Kalman filtering), and classification (e.g., detection or recognition). WM processes generate and update images, maps, entities, events, attributes, and states in the KD. Working together, BG, WM, and SP enable deliberative, reactive, and pre-emptive behavior. Coordination between subordinate BG processes is achieved by cross-coupling among plans and sharing of information among Executors via the KD.

At the top, the highest-level task is defined by the highest-level (i.e., mission) goal. At each successive level in the hierarchy, commanded tasks from above are decomposed into subtasks that are sent to subordinates below. Finally, at the bottom, subcommand outputs are sent to actuators to generate forces and movements. Also at the bottom, sensors transform energy into signals that provide sensory input.

#### 4. Discussion

4D/RCS addresses three of the most significant theoretical arguments raised against the possibility of computers achieving human levels of intelligence. These are:

##### 4.1. Abductive inference

Abductive inference is the process of reasoning backward from consequent to antecedent. It has been described by Peirce (1958) as “nothing but guessing.” The inability of local syntactical systems to perform abductive inference is cited by Fodor (2000) as why he believes computational processes cannot produce true intelligence. To Fodor, all computer operations are inherently local and syntactic, and hence fundamentally incapable of context sensitive logic.

But the RCS architecture is driven top–down by high-level mission goals, priorities, and constraints. These provide global context for making gestalt hypotheses (i.e., perceptual guesses) regarding where to focus attention and how to group (or segment) signals and pixels into patterns and regions that correspond to entities and events in the external world. At each level of sensory processing, abductive inferences in the form of gestalt hypotheses are used to segment signals and images. Abductive inferences can be tested by comparing expectations based on hypotheses against observations from sensors. For each hypothesized entity or event, variance between predictions and observations provides a measure of confidence in the hypothesis. When variance is small, confidence is high, and vice versa. If confidence falls below threshold, a hypothesis is rejected and another generated. This supports Peirce’s claims that abduction can be represented in a “perfectly definite logical form”.

##### 4.2. Symbol grounding

Symbol grounding is the establishment of direct correspondence between internal symbolic data and external real world entities, events, and relationships. The inability of local syntactical systems to perform symbol grounding is cited by Searle (1992) as why he believes computational processes can never be really intelligent. To Searle, computer operations are without semantic meaning because the symbols they manipulate are never grounded in the real world.

But the 4D/RCS architecture establishes and maintains a direct link between the internal world model and the external real world. A RCS attention process directs sensors toward regions of the world that are important. A RCS segmentation process applies context-sensitive gestalt grouping hypotheses to patterns of signals from sensors. As a result of segmentation, spatial and temporal groupings are linked to named symbolic data structures (such as C structs, or C++ objects and classes) that represent hypothesized entities and events. Geometric and temporal attributes of hypothesized groups are computed, and relationships (represented as pointers) between entities, events, and their constituent elements are established and maintained. Finally, entities and events are classified and recognized by comparing observed attributes to stored attributes of class prototypes. This entire process is repeated at each stage of sensory processing at a rate fast enough to capture the dynamics of the entities and events being attended to.

This recursive two-way interaction between model-based expectations and sensory-based observations provides symbol grounding. Expectations based on attributes and class membership of entities and events in the world model are constantly compared against observations derived from sensors monitoring corresponding entities and events in the real world. In this way, symbolic representations of entities, events, and relationships in the 4D/RCS world model are grounded to entities, events, and relationships in the real world.

It should be noted that many other researchers in the field of robotics and autonomous systems have begun to address the symbol grounding problem. A recent special edition of *Robotics and Autonomous Systems* consists of a collection of articles that address the “perceptual anchoring” of symbolic representations to objects in the real world (Coradeschi & Saffiotti, 2003).

##### 4.3. The frame problem

The frame problem is the problem of predicting what in the world changes as the result of an action, and what stays the same. The frame problem results from attempting to model the world entirely in terms of logical propositions (Pylyshyn, 1987). Many important features about the world (in particular geometry and dynamics) are not easily modeled by logical propositions. For example, as I write this, I am pondering the difficulties of using logical propositions to model a bookcase in my office that is filled with books, papers, boxes, folders, and assorted junk and trash. As I ponder this, a fly maneuvers at high speed (in terms of body lengths per second) around my bookcase without danger of collision. Apparently, the fly’s internal model of the world enables it to fly between shelves and stacks of books and papers without collision, and to land on the tip of a straw in an old soda can without difficulty. Surely the fly’s brain does not represent my bookcase in terms of logical propositions.



It has been said that a picture is worth a thousand words. I would venture that a 4-D representation (3 spatial + 1 temporal) of a complex scenario in a dynamic environment may be worth a million logical propositions.

On the other hand, the location and direction of motion of objects in the world are easily represented in an image or map, and distinguishing what changes from what does not in a dynamic environment can be easily determined by simple comparison between one frame and the next. That is why 4D/RCS representations include iconic formats such as visual images and maps in addition to symbolic data structures such as frames, objects, classes, and rules. Both iconic and symbolic formats are linked together by pointers in a real-time relational database that is updated in each node at a rate that is commensurate with the requirements of the planning and control processes within that node.

At the lowest level, signals and images from sensors are sampled many times per second and compared with expectations generated from world model predictions. At all levels, differences between observations and expectations are used to update the internal model by a process of recursive estimation. Predictions from the world model are projected back into sensor coordinates, overlaid on, and cross-correlated with images and maps of the external world. The result is that symbols in the world model are linked to, and can be projected back onto, pixels and regions in images and maps. In this way, the internal world model is effectively served to the external real world, and symbolic data structures in the world model are grounded to entities and events in the real world.

This is illustrated in Fig. 3 where a thread through a single chain of command in the 4D/RCS hierarchy of Fig. 1 is shown. On the far-right are Planner and Executor modules. In the center-right are maps for representing terrain features, road, bridges, vehicles, friendly/enemy positions, and the cost and risk of traversing various regions. On the left are Sensory Processing, World Modeling, and Value Judgment functions that generate and maintain symbolic representations of entities and events, and segmented images and maps with labeled regions in the Knowledge Database. The coordinate transforms in the middle use range information to assign labeled regions in the entity image hierarchy on the center-left to locations on planning maps on the center-right. Note that the perspective viewpoint of images and entity classes on the left is orthogonal to that of planning maps used by BG processes on the right. Planning maps are multi-scale bird's-eye views of terrain, whereas images are sensor-based views roughly tangential to the terrain surface.

At each echelon of the Behavior Generation hierarchy, the range and resolution of knowledge about the world is defined by the requirements of the task. At each BG echelon, the world model enables 4D/RCS behavior generation processes to plan tasks and paths that optimize a cost function that is defined by the global context of task goals, intent, priorities, and constraints. At each level in the Sensory Processing hierarchy, an attention process is driven by task priorities and differences

between observations and predictions to select those regions of the world that are important to the task.

## 5. Engineering methodology

Over the past 35 years, as many different applications have been implemented using the RCS reference model architecture, a RCS software engineering methodology has been developed.

The RCS methodology begins with an in-depth analysis of the task or mission the system is intended to perform. This is followed by the encoding of task knowledge (i.e., procedural knowledge) in the form of a task decomposition tree (e.g., AND/OR graph) that represents the decomposition of tasks into sequences of simpler and simpler subtasks. This task decomposition framework is then mapped onto a hierarchy of organizational units that have the knowledge, skills, and abilities to perform the required task decomposition. The world model knowledge needed to support this task decomposition is then determined. Finally, the sensors and sensory processing capabilities required to maintain the world knowledge are specified.

The fundamental premise is that at each point in time, and within each organizational unit, the state of the task defines the requirements for all of the support processing. In particular, the task state determines what needs to be sensed, what world objects, events, and situations need to be analyzed, what plans need to be generated, and what task knowledge is required to do so (Barbera, Horst, Schlenoff, Wallace, & Aha, 2003; Barbera, Albus, Messina, Schlenoff, & Horst, 2004).

In Fig. 4, an example of the RCS methodology for designing a control system for autonomous on-road driving under everyday traffic conditions is summarized in six steps.

Step 1 consists of an intensive analysis of domain knowledge from training manuals and subject matter experts. Scenarios are developed and analyzed for each task and subtask. The result of this step is a structuring of procedural knowledge into a task decomposition tree with simpler and simpler tasks at each echelon. At each echelon, a vocabulary of commands (action verbs with goal states, parameters, and constraints) is defined to evoke task behavior at each echelon.

Step 2 defines a hierarchical structure of organizational units that will execute the commands defined in step 1. For each unit, its duties and responsibilities in response to each command are specified. This is analogous to establishing a work breakdown structure for a development project, or defining an organizational chart for a business or military operation.

Step 3 specifies the processing that is triggered within each unit in response to input commands, task state, and feedback from the world model. For each input command, a state-graph (or state-table or extended finite state automaton) is defined that defines a plan (or specifies a procedure for making a plan) for accomplishing the commanded

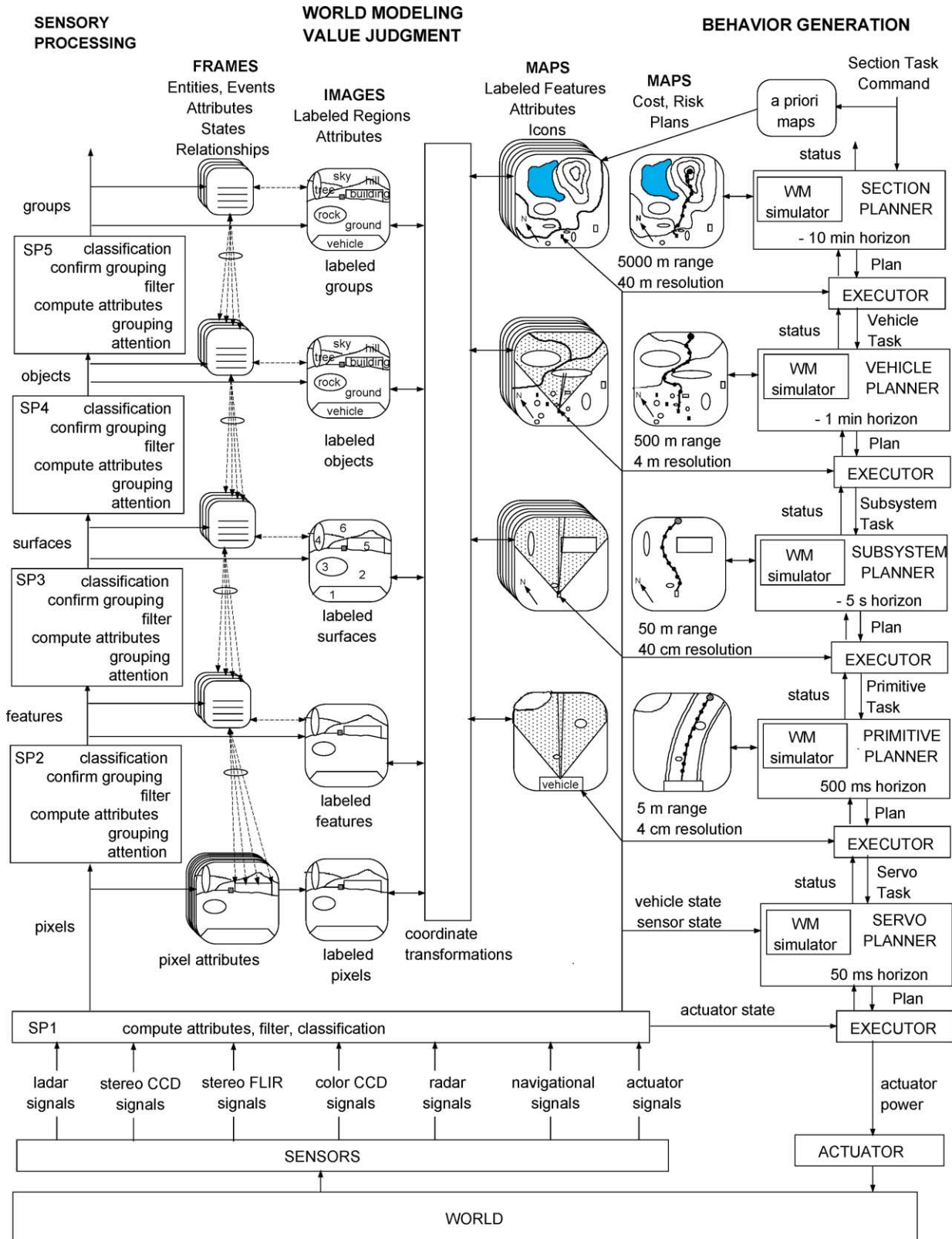


Fig. 3. Five levels of the 4D/RCS architecture developed for Demo III.

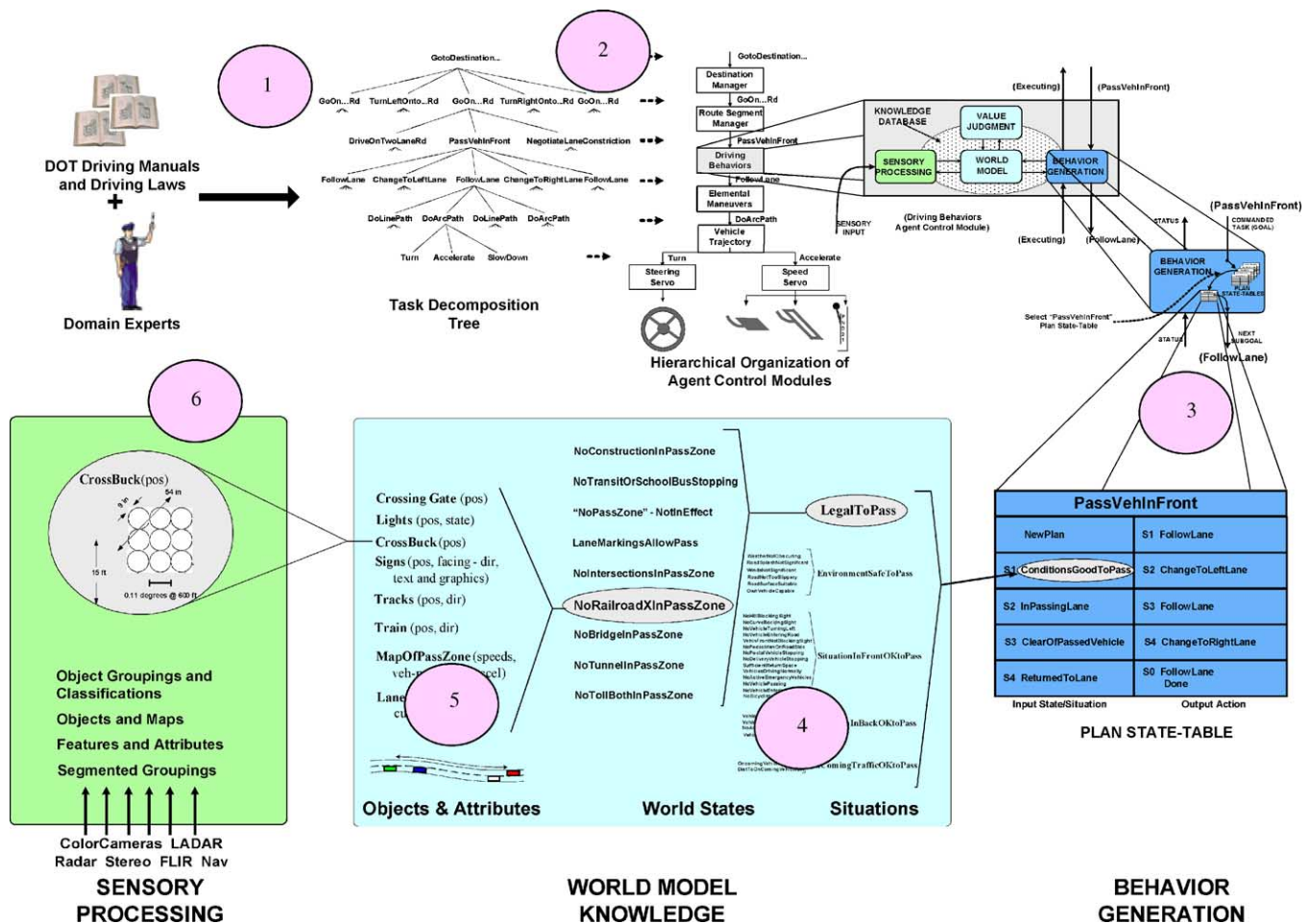


Fig. 4. The six steps of the RCS design methodology.

task. The input command selects (or causes to be generated) an appropriate state-table, the execution of which generates a series of output commands to units at the next lower echelon.

The library of state-tables contains a set of state-sensitive procedural rules that identify all the task branching conditions and specify the corresponding state transition and output command parameters.

The result of step 3 is that each organizational unit has for each input command a state-table of ordered production rules, each suitable for execution by an extended finite state automaton (FSA). The sequence of output subcommands required to accomplish the input command is generated by situations (i.e., branching conditions) that cause the FSA to transition from one output subcommand to the next.

In step 4, each of the situations that are defined in step 3 are analyzed to reveal their dependencies on world and task states. This step identifies the detailed relationships between entities, events, and states of the world that cause a particular situation to be true.

In step 5, we identify and name all of the objects and entities together with their particular features and attributes that are relevant to detecting the above world states and situations.

In step 6, we use the context of the particular task activities to establish times and distances, and therefore, the resolutions in time and space with which the relevant objects and entities must be measured and recognized by the sensory processing component. This establishes a set of requirements and specifications for the sensor system to support each subtask activity.

## 6. Experimental results

Experimental validation of the 4D/RCS architecture has been provided by the performance of the Demo III experimental unmanned ground vehicles (XUVs) in an extended series of demonstrations and field tests during the winter of 2002–2003. These tests were designed to answer two key questions posed by the Lead System Integrator for the Army Future Combat System: First, “what is the current level of maturity of autonomous mobility technology?” Second, “if the operator must assist the vehicle, what impact will those activities have upon the soldier’s workload? The operational concept was that a military unmanned ground vehicles (UGVs) would normally be expected to maneuver autonomously from point A to point B. However, when a UGV runs



across obstacles to forward movement that it cannot negotiate, it is able to call upon an operator for assistance. The UGV is expected to remain in relatively close proximity to its operator (typically between 500 and 1000 m).

The XUVs under test weigh about 1750 kg, have 4-wheel drive 4-wheel steering, and are powered by 78 hp diesel engines with hydrostatic transmissions. The XUVs are capable of operation in two modes: autonomous and teleoperated. For the field tests, each XUV was equipped with an inertial reference system, a commercial grade GPS receiver (accurate to about  $\pm 20$  m), a LADAR camera with a frame rate of 10 frames per second (for autonomous driving), a set of color CCD cameras (for teleoperation), military radios, low-power wireless local area network, and a variety of internal sensors. The LADAR had a field of view  $90^\circ$  wide and  $20^\circ$  high with resolution of about  $1/2^\circ$  per pixel. It was mounted on a pan/tilt head that enabled it to look in the direction that the XUV planned to drive. The LADAR was able to detect the ground out to a range of about 20 m, and detect vertical surfaces (such as trees) out to a range of about 60 m.

The SUV software was developed in accordance with the 4D/RCS reference model architecture. However, only the first three levels of the 4D/RCS architecture were populated with software because the tests were primarily designed to measure the technology readiness level of the subsystem echelon autonomous mobility behaviors only. Section and vehicle echelon tactical behaviors were supplied by a human operator

via the Operator Control Unit, using waypoints about 100 m apart laid out on aerial photographs by human experts.

Tests were conducted on three types of terrain: rolling/arid, rolling/vegetated, and urban. Tests on rolling/arid terrain were conducted at Tooele Army Depot, Utah on open terrain with sagebrush up to 1 m high. Obstacles to mobility included deep gullies, erosion features, rocks, and breastworks of an abandoned dam with steep slopes. Tests on rolling/vegetated terrain were conducted on maneuver ranges at Ft. Indiantown Gap, Pennsylvania through woods of varying density, open terrain with tall grass and brush, dirt roads, creeks, lakes, and trails with constriction points such as bridges. Urban tests were conducted among Army barracks at Ft. Indiantown Gap where there are buildings, phone poles, phone booths, parked cars, dumpsters, and culverts. For the tests, rubble was dumped on the streets, along with abandoned cars, and human mannequins.

Routes for XUV missions were laid out on a terrain map by trained Army scouts using aerial photographs. This information was conveyed to the XUVs in the form of GPS waypoints spaced about 100 m apart. Two test courses were laid out at each test site with varying levels of terrain complexity. Test missions of three different lengths, 500, 1000, and 2000 m were defined, and randomized with regard to start points and various other parameters. Two teams ran a total of 144 missions over a period of 6 days at each of the test sites.

The tests were designed to test the XUVs ability to traverse cross-country and through urban areas. The XUVs operated



Fig. 5. A Demo III experimental unmanned ground vehicle (XUV) with a 4D/RCS control system architecture traveling autonomously through the woods on a test mission during a snowstorm.

completely autonomously until they got into trouble and called for help. At that point, an operator (without line-of-sight) was alerted and asked to teleoperate the vehicle out of difficulty. During these teleoperations, data was collected on the cause of the difficulty, the type of operator intervention required to extract the XUV, the work load on the operator, and the time required before the XUV was returned to autonomous mode. Typical reasons for calling for help were that the XUV was unable to proceed because of some terrain condition or obstacle (such as soft sand on a steep slope, or dense woods), and was unable to find an acceptable path plan after several attempts at backing up and heading a different direction.

During three major experiments designed to determine the technology readiness of autonomous driving, the Demo III experimental unmanned vehicles were driven a total of 550 km, over rough terrain: (1) in the desert; (2) in the woods, through rolling fields of weeds and tall grass, and on dirt roads and trails; and (3) through an urban environment with narrow streets cluttered with rubble, parked cars, dumpsters, culverts, telephone poles, and manikins. Tests were conducted under various conditions including night, day, clear weather, rain, and falling snow. Fig. 5 shows an XUV traveling through the woods at Ft. Indiantown Gap during a snowstorm. The unmanned vehicles operated over 90% of both time and distance without any operator assistance. A detailed report of these experiments has been published (Camden et al., 2003), along with high resolution ground truth data describing the terrain where the XUVs experienced difficulties (Witzgall, Cheok, & Gilsinn, 2003). A photograph of an XUV maneuvering through the woods in a snowstorm is shown in Fig. 5.

It should be noted that the Demo III technology readiness tests were limited to environments devoid of moving objects such as on-coming traffic, pedestrians, or other vehicles. It should also be noted that only the lower three levels of the 4D/RCS architecture were involved in these tests. All of the higher-level capabilities were provided by the human operator, and by subject matter experts. Both of these limitations must be overcome before UGVs can play a major role in Army field operations. However, the results of the technology readiness tests were sufficiently positive that the Demo III technology has been incorporated into the system development and demonstration phase of the Army Future Combat System procurement, and enough of the 4D/RCS reference model has been implemented to demonstrate that the fundamental concept is valid and the more advanced features are feasible. As a result, 4D/RCS has been adopted by the Army as the reference model architecture for the Autonomous Navigation System that will be deployed on all the new ground vehicles in the Future Combat System.

## 7. Future prospects

Current research is focused on operating in environments that contain moving objects and implementing the upper

echelons of the 4D/RCS architecture. Recently, NIST analyzed the set of tasks required for driving in normal traffic, and begun to develop an ontology of knowledge that is required for autonomous driving. (Barbera et al., 2004) NIST, the Robotics Technology Alliance, and a number of private contractors are currently developing sensors and algorithms for autonomous driving on normal roads and streets, e.g., driving on country roads and city streets with on-coming traffic, negotiating intersections with traffic signals and pedestrians, and maneuvering in and out of parking spaces. Researchers are analyzing the sensing capabilities and perception algorithms necessary to detect and track moving objects under various conditions in normal traffic. Schlenoff, Madhavan, and Barbera (2004) are developing world modeling algorithms for representing moving objects. Schlenoff and colleagues are also developing ontologies and tools for representing all the objects, events, situations, and relationships that are required for competent driving behavior. ARL sponsored research at NIST currently includes work on tactical behaviors for teams of real and virtual, manned and un-manned, military ground and air vehicles.

Based on recent progress and technological developments, we believe that autonomous driving is an excellent topic for future research. There are a number of reasons for this conclusion.

First, it is a problem domain for which there is a large potential user base, both in the military and civilian sectors. This translates into research funding.

Second, it is a problem domain where physical actuators and power systems are readily available. Wheeled and tracked vehicle technology is mature, inexpensive, and widely deployed.

Third, it is a problem domain for which the technology is ready. The invention of real-time LADAR imaging makes it possible to capture the 3-D geometry and dynamics of the world. This has broken the perception barrier. The continued exponential growth rate in computing power per dollar cost has brought the necessary computational power within the realm of economic viability. This has broken the computational barrier. Cognitive modeling and intelligent control theory has advanced to the point where the engineering of intelligent systems is feasible. This has broken the technology barrier. Given the knowledge that we now have, and making reasonable assumptions regarding the growth of computational power and expected levels of funding over the next two decades, we estimate that autonomous driving with safety and efficiency comparable to human performance will be both technologically and economically feasible by the year 2025.

Finally, autonomous driving is problem domain of fundamental scientific interest. Locomotion is perhaps the most basic of all behaviors in the biological world. Locomotion is essential to finding food and evading predators throughout the animal kingdom. The brains of all animate creatures have evolved under the pressures of natural selection in rewarding successful locomotion

behavior. It is therefore, not unreasonable to suspect that building truly intelligent mobility systems will reveal fundamental new insights into the mysteries of how the mechanisms of brain give rise to the phenomena of intelligence, consciousness, and mind.

## 8. Conclusions

In many ways, 4D/RCS is a superset of Soar, ACT-R, ICARUS, IMPRINT, Pilot's Associate, Dickmanns 4-D approach, and even behaviorist architectures such as Subsumption (Brooks, 1986) and its many derivatives (Arkin, 1998; Mataric, 1993). 4D/RCS represents a conscious attempt to incorporate and integrate many different and diverse concepts and approaches into a harmonious whole. It is hierarchical but distributed, deliberative yet reactive. It spans the space between the cognitive and reflexive, between planning and feedback control. It bridges the gap between spatial distances ranging from kilometres to millimetres, and between time intervals ranging from months to milliseconds. And it does so in small regular steps, each of which can be easily understood and readily accomplished through well-understood computational processes.

Each organizational unit in 4D/RCS refines tasks with about an order of magnitude increase in detail, and an order of magnitude decrease in scale, both in time and space. At the upper levels, most of the computational power is spent on cognitive tasks, such as analyzing the past, understanding the present, and planning for the future. At the lower levels, most of the computational power is spent in motor control, and the early stages of perception.

However, at every level, the computational infrastructure is fundamentally the same (except for scale in time and space). Computational modules accept inputs and produce outputs. This I/O transfer function can be implemented in a variety of different ways, including neural nets, finite state automata, search based algorithms, means-ends analysis, fuzzy control, or program function calls. Knowledge is represented in matrices, strings, pointers, frames, rules, and differential equations. At various levels and in many different ways, computational modules process sensory data, model the world, and decompose higher-level intentions into lower-level tasks. Within each module, this process is both limited in complexity and finite in scope. Perhaps most important, 4D/RCS makes the processes of intelligent behavior understandable in terms of computational theory. Thus, it can be engineered into practical machines.

4D/RCS is an attempt to address the full range of complexity inherent in embodied cognitive systems, from sensing to cognition to action in real-world environments. The 4D/RCS architecture suggests how high level goals can be decomposed and merged with sensory feedback to generate behaviors that are both goal directed and sensory

interactive. The 4D/RCS design methodology defines a series of steps for engineering intelligent systems that conform to the reference model architecture. This methodology consists of: (1) task analysis and decomposition; (2) mapping tasks onto organizational units; (3) developing state graph and state machine representations of procedural knowledge required to accomplish tasks; (4) defining the transition conditions that sequence subtask activities; (5) defining the objects, events, and relationships that define the transition conditions, and developing the data structures required to support planning and control of intelligent behavior; and (6) developing specifications for sensors to gather information about the world, and for sensory processing algorithms that provide symbol grounding.

There are, of course, many unresolved issues. For example, the claim is often made that any rule-based system such as 4D/RCS architecture must be brittle. The premise is that it is not possible to write a rule for every possible situation. And this is true—but irrelevant. It is only necessary to write a rule for every class of situations. And the number of classes need be only as large as the number of different types of action that the system is able to perform. Grouping hypotheses (abductive inferences) within the perception system segment the infinite number of possible situations into a finite number of classes, each of which can have attributes and parameters that can be used for generating a gradation of activity. Thus, the problem lies not in building a control system with an infinite number of rules. The problem lies in building a perception system that can segment the infinite number of possible situations into a finite number of classes. In large measure, the degree of intelligence depends on the number of classes that the perception system can distinguish. More intelligent systems typically can distinguish a larger number of classes. But, even for humans, the number of classes is not infinite.

It is also often argued that a system based on finite-state automata, or production rules, cannot be adaptive, and therefore cannot learn. But 4D/RCS integrates many concepts other than production rules. For example it supports recursive estimation, image understanding, 3-D geometry, search-based algorithms, and differential equations. 4D/RCS provides links between symbolic entities or events and regions in maps or images. And, 4D/RCS supports a wide variety of learning approaches. 4D/RCS supports map learning, and remembering where landmarks are located in the environment. 4D/RCS supports learning from human instructors by adding or modifying rules. 4D/RCS also supports reinforcement learning for adjusting control parameters to optimize dynamic response. In fact, the first version of RCS was based on the CMAC neural net, and every functional module in the 4D/RCS architecture could, in principle, be implemented by a neural net (Albus, 1975a, 1975b).

4D/RCS is often criticized for being too complicated—and for simple applications this may be true. All of the features



of 4D/RCS are not required for every problem. 4D/RCS is designed to enable intelligent systems that rival human levels of performance. Building systems that accomplish complex missions in complex dynamic and uncertain environments requires complexity. The human brain contains on the order of  $10^{11}$  neurons with perhaps as many as  $10^{15}$  synapses. These are organized into hundreds of computational modules interconnected by thousands of communication pathways. The brain consumes approximately 20% of the body's energy supply. If this were not required for intelligent behavior, nature would have simplified the human brain long ago. High degrees of intelligence require complexity.

Finally, there is the issue of scaling up. Many promising approaches have failed when confronted by large real-world problems. Certainly 4D/RCS has not yet been fully implemented in a really large problem domain, such as an Army scout platoon-echelon route reconnaissance or assault mission against an opposing force of skilled human warriors. Before this can be achieved, considerable investments must be made in software engineering tools and software development environments that enable the knowledge skills and abilities of human experts to be encoded in software algorithms to be embedded in the control architecture.

There has been some progress along these lines. Proctor and Shackelford have developed a Neutral Messaging Language (NML) that can be used for establishing communications between RCS modules on a single machine, or over a network of multiple machines. (Gazi et al., 2001) NML is publicly available over the internet at <http://isd.mel.nist.gov/projects/rcslib>. Other tools, some specifically designed for generating RCS code, are available commercially. Two introductory textbooks and other documents are also publicly available. (Albus & Meystel, 2001; Meystel & Albus, 2002; Albus et al., 2002) But much more remains to be done before this technology is ready for full-scale deployment.

## Acknowledgements

This work was partly supported by the Army Research Laboratory, Charles Shoemaker, Program Manager, and by DARPA, Douglas Gage, Program Manager. It was also supported by the Intelligent Systems Division of NIST through funding for Research and Engineering of Intelligent Systems.

## References

- Albus, J. S. (1971). A theory of cerebellar function. *Mathematical Biosciences*, 10, 25–61.
- Albus, J. S. (1975a). A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control*, September, 220–227.
- Albus, J. S. (1975b). Data storage in the cerebellar model articulation controller (CMAC). *Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control*, September, 228–233.
- Albus, J. (1981). *Brains, Behavior, and Robotics*. Peterborough, NH: BYTE/McGraw Hill.
- Albus, J. S. (1991, May/June). Outline for a theory of intelligence. *IEEE Transactions on Systems, Man and Cybernetics*, 21(3), 473–509.
- Albus, J. S. (1997). The NIST real-time control system (RCS): An approach to intelligent systems research. *Journal of Experimental and Theoretical Artificial Intelligence*, 9, 157–174.
- Albus, J. et al. (2002). *4D/RCS: A Reference Model Architecture for Unmanned Vehicle Systems, Version 2.0, NISTIR 6910*. Gaithersburg, MD: National Institute of Standards and Technology.
- Albus, J., & Meystel, A. (2001). *Engineering of Mind: An Introduction to the Science of Intelligent Systems*. N.Y. Wiley.
- Anderson, J. (1983). *The Architecture of Cognition*. Mahwah, NJ: Lawrence Erlbaum Associates.
- Archer, R., Lebiere, C., Warwick, W., & Schunk, D. (2003). Integration of task network and cognitive models to support system design. In *Proceedings of the Collaborative Technology Alliances Conference 2003 Advanced Decision Architectures*.
- Arkin, R. C. (1998). *Behavior-Based Robotics*. Cambridge, MA: MIT Press.
- Barbera, A. J., Albus, J. S., & Fitzgerald, M. L. (1979). Hierarchical control of robots using microcomputers. In *Proceedings of the Ninth International Symposium on Industrial Robots*.
- Barbera, A. J., Fitzgerald, M. L., Albus, J. S., & Haynes, L. S. (1984). RCS: The NBS real-time control system. In *Proceedings of the Robots 8 Conference and Exposition*.
- Barbera, A., Horst, J., Schlenoff, C., Wallace, E., & Aha, D. (2003). Developing world model data specifications as metrics for sensory processing for on-road driving tasks. In *Proceedings of the 2003 PerMIS Workshop*.
- Barbera, A., Albus, J., Messina, E., Schlenoff, C., & Horst, J. (2004). How task analysis can be used to derive and organize the knowledge for the control of autonomous vehicles. In *Proceedings of the Knowledge Representation and Ontology for Autonomous Systems Workshop of the 2004 AAAI Spring Symposium*.
- Bishop, R. (2003). *Presentation to a NIST/DARPA Workshop on Autonomous Driving*. Tysons Corner, VA: DARPA Mobile Autonomous Robot Software Workshop.
- Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2, 14–23.
- Camden, R., Bodt, B., Schipani, S., Bornstein, J., Phelps, R., Runyon, T., French, F., Shoemaker, C. (2003). Autonomous Mobility Technology Assessment: Interim Report—February 2003. ARL-MR-565, Army Research Laboratory, ATTN: AMSRL-WM-RP, Aberdeen Proving Ground MD 21005-5066.
- Coradeschi, S., & Saffiotti, A. (Eds.). (2003). Special issue on perceptual anchoring. *Robotics and Autonomous Systems*, 43, 2–3.
- Dickmanns, E. D. (1992). A general dynamic vision architecture for UGV and UAV. *Journal of Applied Intelligence*, 2, 251–270.
- Dickmanns, E. D. (1999). An expectation-based, multi-focal, saccadic (EMS) vision system for vehicle guidance. In *Proceedings of the Ninth International Symposium on Robotics Research (ISRR'99)*.
- Fodor, J. (2000). *The Mind does not Work that Way*. Cambridge, MA: MIT Press.
- Gazi, V., Moore, M. L., Passino, K. M., Shackelford, W. P., Proctor, F. M., & Albus, J. S. (2001). *The RCS Handbook: Tools for Real Time Control Systems Software Development*. NY: Wiley.
- Gonzalez, C. (2003). ACT-R implementation of an instance-based decision making theory. In *Proceedings of the Collaborative Technology Alliances Conference 2003 Advanced Decision Architectures*.
- J-UCAS, <http://www.darpa.mil/j-ucas/>.
- Kieras, D., & Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction*, 12, 391–438.



- Kurzweil, R. (1999). *The Age of Spiritual Machines*. New York, NY: Penguin Books.
- Laird, J., Newell, A., & Rosenbloom, P. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33, 1–64.
- Mataric, M. (1993). Designing emergent behaviors: From local interactions to collective intelligence. In J. A. Meyer, H. L. Roitblat, & S. W. Wilson (Eds.), *From Animals to Animats: Proceedings of the Second International Conference on Simulation of Adaptive Behavior* (pp. 432–441). Cambridge, MA: MIT Press.
- Meystel, A., & Albus, J. (2002). *Intelligent Systems: Architecture, Design and Control*. NY: Wiley.
- Minton, S. N. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42, 363–391.
- Newell, A., & Simon, H. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.
- Peirce, Charles Sanders (1958). *Collected Papers, Band VII*, (Hrsg.). Arthur W. Burks.
- Pylyshyn, Z. (Ed.). (1987). *The Robot's Dilemma: The Frame Problem in Artificial Intelligence*. Norwood, NJ: Ablex.
- Searle, J. (1992). *The Rediscovery of the Mind*. Cambridge, MA: MIT Press.
- Schlenhoff, C., Madhavan, R., & Barbera, T. (2004). A hierarchical, multi-resolutional moving object prediction approach for autonomous on-road driving. In *Proceedings of the 2004 ICRA Conference*.
- Shapiro, D., & Langley, P. (1999). Controlling physical agents through reactive logic programming. In *Proceedings of the Third International Conference on Autonomous Agents* 386–387.
- Shoemaker, C., Bornstein, J., Myers, S., & Brendle, B. (1999, April). Demo III: Department of Defense testbed for unmanned ground mobility. *SPIE Conference on Unmanned Ground Vehicle Technology*, SPIE, vol. 3693.
- SoarTech (2004). <http://www.soartech.com/htmlonly/projects.php>.
- Witzgall, C., Cheok, G. S., & Gilsinn, D. E. (2003). Terrain characterization from ground-based LADAR. In *Proceedings of PerMIS '03 Workshop*.

**James Albus** founded and led the Intelligent Systems Division at the National Institute of Standards and Technology for 20 years. He is currently a Senior NIST Fellow pursuing research in autonomous vehicles, perception, knowledge representation, and intelligent behavior. During the 1960s he designed electro-optical systems for more than 15 NASA spacecraft. During the 1970s, he developed a model of the cerebellum that is still widely used by neurophysiologists today. Based on that model, he invented the Cerebellar Model Arithmetic Controller (CMAC) neural net, and co-invented the real-time control system (RCS) that has been used for developing a wide variety of intelligent systems for both government and commercial applications. He is the inventor of the NIST RoboCrane. Dr. Albus is the author of five books and more than 150 scientific papers, journal articles, and official government studies on intelligent systems and robotics, and has lectured extensively throughout the world.

**Anthony Barbera** is the other co-inventor of the RCS reference model architecture and developer of the RCS software development methodology. For more than 30 years, Dr. Barbera has been building intelligent systems using RCS. He has developed RCS controllers for laboratory robots, factory automation systems, machine tools, nuclear submarine automation systems, postal general mail facility automation, stamp distribution automation, water-jet cutting controllers, and autonomous ground and air vehicles. He is currently designing 4D/RCS software for military tactical behaviors for platoon level operations.