The Real-time Control System (RCS) architecture developed at NIST and elsewhere over the past two decades (Barbera, Albus, & Fitzgerald 1979; Albus et al. 1982; Albus 1991;) defines a canonical form for a nested intelligent control system. The RCS architecture consists of a hierarchically layered set of processing modules connected together by a network of communications pathways. The primary distinguishing feature of the layers is the bandwidth of the control loops. The characteristic bandwidth of each level is determined by the spatial and temporal integration window of filters, the temporal frequency of signals and events, the spatial frequency of patterns, and the planning horizon and granularity of the planners that operate at each level. At each level, tasks are decomposed into sequential subtasks, to be performed by cooperating sets of subordinate agents. Signals from sensors are filtered and correlated with spatial and temporal features that are relevant to the control function being implemented at that level.

The four basic types of processing modules from which the RCS architecture is built are:

1. **Behavior Generating (BG) modules**. BG modules contain job assignment, planning, and control algorithms. These embody knowledge about how to perform tasks - i.e. how to decompose tasks into subtasks that subordinate agents know how to execute. BG modules can accommodate a variety of planning algorithms, from simple table look-up of pre-computed plans, to real-time search of configuration space, or game theoretic algorithms for multi-agent cooperating and competitive groups. Planning horizons at high levels may span months or years, while planning horizons at the bottom level typically are less than 50 milliseconds. Control loop bandwidth at each level is typically at least ten times the reciprocal of the planning horizon at that level.

2. **World Modeling (WM) modules**. The WM modules model the state space of the problem domain. They contain information storage and retrieval mechanisms, as well as algorithms for transforming information from one coordinate system to another. WM modules use dynamic models to generate expectations, and predict the results of current and future actions. WM modules may contain recursive estimation algorithms and processes that compute lists of attributes from images, graphics engines that generate images from symbolic lists, and recognition and detection algorithms that perform pattern matching operations necessary to verify the identification of features, surfaces, objects, and groups. The WM module maintains a knowledge database (KD), acts as a question answering system, and uses information from the KD to predict or simulate the future.

3. **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 and pattern recognition algorithms. Interactions between WM and SP modules can generate a variety of filtering and detection processes such as Kalman filtering and recursive estimation, Fourier transforms, and phase lock loops. Vision system SP modules process images to detect brightness, color, and range discontinuities, optical flow, stereo disparity, and utilize a variety of signal detection and pattern recognition algorithms to analyze scenes and compute information needed for manipulation, locomotion, and spatio-temporal reasoning.

4. **Value Judgment (VJ) modules**. VJ modules contain algorithms for computing cost, risk, and benefit, for evaluating states and situations, for estimating the reliability of state estimations, and for assigning cost-benefit values to objects and events. VJ modules may compute Bayes and Dempster-Schafer
statistics on information about the world based on the correlation and variance between observations and predictions.

5. Knowledge Database (KD) modules KD modules consist of data structures that contain state variables, iconic images, and symbolic frames containing lists of attributes. 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 and long term memory elements. The KD is typically implemented in a distributed fashion, suitable for real-time data retrieval and update.

6. A communication system that conveys messages between the various modules. The communication system provides a network of pathways that transmits messages between the various processing and database modules. The communications system richly, but not completely, interconnects the modules.

The various modules in the RCS architecture act as a collection of intelligent agents (or software objects), sending and receiving messages to and from each other. These messages convey commands and requests, and return status.

The RCS architecture has evolved over the past two decades from a rather simple robot control schema to a reference model architecture for intelligent system design. From the beginning, RCS has represented a conscious attempt to emulate the function and structure of the neurological machinery in the brain. Each RCS module has properties that are known, or hypothesized, to exist in the brain. For example, RCS modules may be constructed from neural nets such as CMAC (Albus 1975a; 1975b) that compute arithmetic and/or logical functions on a set of inputs to produce a set of output state variables. These can be carried over communications pathways to other functional modules that may use them to perform further functional computations, or to generate addresses, or to store information in memory for latter use. RCS functional modules may add, subtract, multiply, differentiate, integrate, compute correlation functions, recognize patterns, generate names or addresses of symbolic representations, or to perform planning functions at a hierarchy of levels. In its most complete theoretical form, the RCS reference model architecture provides a framework for integrating concepts from artificial intelligence, machine vision, robotics, computer science, control theory, operations research, game theory, signal processing, filtering, and communications theory.

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

1. A Horizontal Machining Workstation This project was part of the NBS Automated Manufacturing Research Facility (AMRF). It included an integrated sensory-interactive real-time control system for a robot with a structured light machine vision system, a machine tool, an automatic fixturing system, and a pallet shuttle. The robot included a quick change wrist, a part handling gripper with tactile sensors, and a tool handling gripper for loading and unloading tools in the machine tool magazine. Plans were represented as state-tables, and a wide variety of sensory interactive behaviors were demonstrated. These included locating and recognizing parts and part orientation of unoriented parts presented in trays, and automatically generating part handling sequences for part and tool loading and unloading. (Albus et al. 1983; Barbera et al. 1984; Wavering & Fiala 1987)

2. A Cleaning and Deburring Workstation This project was also part of the AMRF. It included two robots, a set of buffing wheels, a part washing/drying machine, and a variety of abrasive brushes. Part geometry was input from a CAD database. Deburring tool paths were automatically planned from knowledge of the part geometry plus operator input indicating which edges were to be deburred. Deburring parameters such as forces and feed rates were also selected from a menu by the operator. Part handling sequences were planned automatically for loading parts in a vise, and turning parts over to permit tool and gripper access. Force sensors and force control algorithms were used during task execution to modify the planned paths so as to compensate for inaccuracies in robot kinematics and dynamics. (Murphy, Norcross, & Proctor 1988; Norcross 1989; Proctor, Murphy, & Norcross 1989)

3. An Advanced Deburring and Chamfering System This project is currently underway. It is designed for high speed force controlled precision deburring and chamfering of complex parts made of hard metal, such as jet engine turbine blades. It consists of a robot and a deburring tool mounted on a micro-positioner with force sensitive end point control. Impedance control of the tool endpoint is performed in real-time through an algorithm that independently controls the stiffness and damping coefficients normal-to and tangential-to the part edge at the tool contact point. Part geometry is derived
from standard IGES CAD data formats. Edge selection is done by a human operator. Tool speeds, feeds, and force parameters are automatically generated by formula using parameters from a materials machinability database. (Stouffer et al. 1993; Stouffer 1995)

4. NBS/NASA Standard Reference Model Architecture for the Space Station Telerobotic Servicer (NASREM) This project was funded by NASA Goddard Space Flight Center. NASREM was used by Martin Marietta to develop the control system for the space station 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, McCain, & Lumia 1989; Albus et al. 1989a).

5. An architecture for Coal Mining Automation This project effectively transferred the RCS architecture and methodology to a large team of researchers in the US Bureau of Mines who are tasked with developing prototype coal mining automation sensors and systems and transferring such systems, 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 design, built, and demonstrated. The same control system was later demonstrated with an actual mining machine and sensors. (Albus et al. 1989b; Huang, Quintero, & Albus 1991; Horst & Barbera 1983; Huang, Horst, & Quintero 1992; Horst 1993; 1994)

6. An nuclear submarine maneuvering system This project demonstrated the design and implementation in simulation of maneuvering and engineering support systems for a 637 class nuclear submarine. The maneuvering system involves an automatic steering, trim, speed, and depth control system. The system demonstrated the ability to execute a lengthy and complex mission involving transit of the Bering Straits under ice. Ice avoidance sonar signals were integrated into a local map using a CMAC 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 lub oil fire by reconfiguring ventilation systems, rising in depth to snorkel level, and engaging the diesel engines for emergency propulsion. (Huang, Hira, & Feldman 1992b; 1992a; Huang et al. 1993; Huang, Hira, & Quintero 1993)

7. A control system for a U.S. Postal Service Automated Stamp Distribution Center This system demonstrated the ability to route packages through a series of carossels, conveyors, and storage gins, to maintain precise inventory control, provide security, and generate maintenance diagnostics in the case of system failure. The distribution center was designed and tested first in simulation, and then implemented as a full scale system. The system contained over 220 actuators, 300 sensors, and ten operator workstations. An even larger and more complex RCS system for controlling a general mail facility is still under development. (Advanced Technology Research Corporation 1991)

8. A control system for Multiple Autonomous Undersea Vehicles This system was developed for controlling a pair of experimental vehicles designed and built by the University of New Hampshire. The RCS control system included a real-time path planner for obstacle avoidance, and a real-time map builder for constructing a topological map of the bottom. A series of tests was conducted in Lake Winnipesaukee during the fall of 1987. (Albus & Blidberg 1987; Herman & Albus 1987; 1988; Albus 1988)

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

A second version of this RCS system has demonstrated the ability to drive the HMMWV automatically using TV images processed through a machine vision system with a real-time model matching algorithm for tracking lane markings. The RCS real-time vision processing system has enabled this vehicle to drive automatically at speeds up to sixty miles per hour on the highway, and at speeds up to thirty-five miles per hour on a winding test track used by the county police for driver-training. (Herman, Albus,
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. 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 program, EMC functional modules have been defined, and Application Programming Interfaces (APIs) are being defined for sending messages between the functional modules. A prototype machine tool controller is being installed in a General Motors plant as part of the DoE-TEAM/NIST-EMC government/industry consortium. The goal of this effort is to define a set of standard application programming interfaces for open architecture controllers. (Proctor & Michaloski 1993; Albus et al. 1992; Proctor et al. 1993).

All of the projects listed above that have used the RCS architecture have implemented only a subset of the features of most advanced theoretical form of the RCS reference model architecture (Albus 1994). This is because the RCS theoretical development has remained well advanced over what it has been possible to implement, given programatic limitations in funding.

Current work at NIST and elsewhere is pursuing more complex implementations of RCS. For example, efforts to incorporate human operator interfaces into the RCS architecture that began with NASREM have continued with the Air Force/JPL/NIST Universal Telerobotic Aircraft Project (UTAP) (Michaloski, Backes, & Lumia 1995; Lumia et al. 1994), and the NIST RoboCrane. Work is also under way to integrate the RCS architecture with the Manufacturing Systems Integration (MSI) factory control architecture, and the Quality In Automation (QIA) architecture. (Kramer & Senehi 1993; 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 (Michaloski & Wheatley 1990; Quintero & Barbera 1992; 1993).

References


Albus, J. 1975a. Data storage in the cerebellar model articulation controller (cmac). Transactions of the


Michaloski, J. L., and Wheatley, T. 1990. Design principles for a real-time robot control system,. In
Proceedings of the IEEE International Conference on Systems Engineering.


