To be presented as an Invited Paper for the IEEE International Industrial Automation and Control Conference'95, Taipei, Taiwan, May 1995.

AN OPEN ARCHITECTURE BASED FRAMEWORK FOR AUTOMATION AND INTELLIGENT SYSTEM CONTROL

Hui-Min Huang
John Michaloski
Nicholas Tarnoff
Marilyn Nashman
National Institute of Standards and Technology
Gaithersburg, MD 20899
U.S.A.
{huang, michaloski, tarnoff, or nashman}@cme.nist.gov

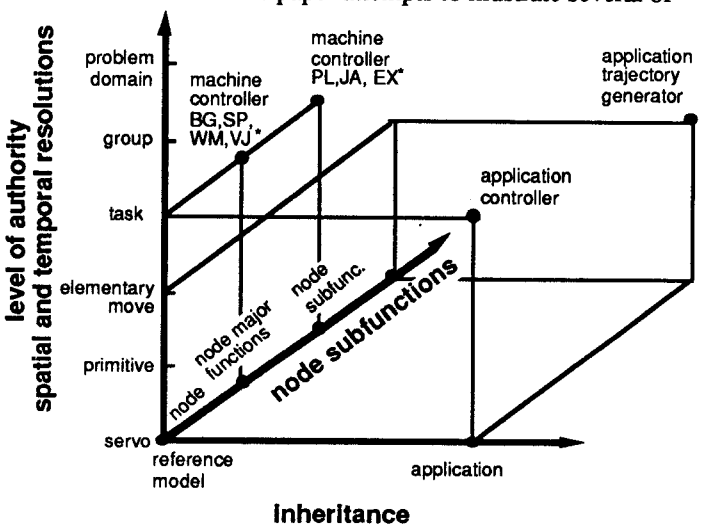
Abstract

This paper conceptualizes a framework that features multiple dimensions for modeling the multiple aspects of complex automation systems. This framework facilitates open and scalable system architecture. Its well-defined structures facilitate efficient processing of system intelligence. Several automation models are used to illustrate the validity of this framework. This work is supported jointly by the National Institute of Standards and Technology (NIST) Systems Integration for Manufacturing Applications (SIMA) Joint Architecture project and the NIST Intelligent Systems Division (ISD) Tools project.

1. Introduction

The major US automotive manufacturers have issued a white paper citing open and modular architecture controllers as resulting in benefits including reduced initiative and lifecycle costs, maximized machine up time, easy maintenance, easy integration, and efficient incorporation of new technologies [1]. In a workshop on advanced machine tool research directions sponsored by the Manufacturing Engineering Laboratory of the NIST, industrial participants expressed their desire for complex and advanced control capabilities that are agile and portable [2]. Sweet et al. [3] identified that large scale, real-time, and distributed are among the key software technologies for the Aerospace Industries Association (AIA). Software is a predominant factor in determining the success and performance of complex control systems. Meanwhile, the fundamental and generic principles of handling software and systems engineering processes are still being searched for within the global engineering community. Large scale automation and intelligent systems impose another layer of difficulty due to their often mission critical and real-time nature. Recent efforts in the areas of computer-aided software engineering environments and architectures include a modular architecture for autonomous robots [4] and a Task Control Architecture (TCA) [5]. However, scalability might be a problem for these architectures as they are not intended to model multiple cooperating agents. Object oriented paradigms [6, 8] are popular for handling the representation problems of software systems. However, literatures [7, 8] have pointed out that these paradigms are not necessary for all the problems.

Researchers in the Intelligent Systems Division (ISD) of Manufacturing Engineering Laboratory of the National Institute of Standards and Technology (NIST) have been developing and applying a particular approach, called Real-time Control Systems (RCS) [9], to this automation and intelligent control problem. RCS covers most of the major critical issues demanded by the manufacturing, control, and software engineering industries. The remainder of the paper attempts to illustrate several of



* See section 3.

Figure 1: A Multiple Dimensional Reference Model Architecture for Intelligent Systems

these critical issues. Our ultimate goal is for RCS to evolve into a unified solution paradigm to the large scale automation and intelligent control problem. In an earlier paper, Huang [10] outlines RCS as a multiple dimensional reference model architecture (Figure 1). This paper further extends this concept, especially toward manufacturing application domains. Several projects that have been developed using different versions of RCS are briefly discussed in terms of how they can fit into this model.

2. Multiple dimensional reference model architecture

The NIST Hierarchical Real-time Control (RCS) is a reference model architecture, under development and being applied since two decades ago. Research results obtained [11, 12, 13] have demonstrated that the concept of reference model architecture is extremely useful since it provides a unified approach and common execution behavior across classes of problems. The RCS architecture prescribes a canonical form based on a generic intelligent machine system model (see section 3). This facilitates the architecture's openness and scalability. RCS is rich because it applies multiple but integrated representation paradigms to model the necessary perspectives of a system. Rich representations are important. Literature has revealed that applying a simplistic modeling paradigm has failed for large systems [8].

The term hierarchy can mean different things to different people. In an object oriented paradigm, a hierarchy can mean a tree describing class derivation. In a functional decomposition paradigm, a hierarchy can mean layers of subfunctions representing a system. Booch [6] describes these two perspectives. In previous research efforts, NIST ISD has successfully explored the task, or level of authority, based hierarchy (see section 4) and a functional view of RCS (see section 3). NIST ISD has also been developing generic software templates and libraries that can be inherited by new RCS applications. As described in [10], a multiple dimensional framework representing the following three paradigms: level of authority, functional decomposition, and inheritance (Figure 1) can be used to model intelligent systems. These features characterize RCS. At the origin of the coordinate system is a reference model built up using a generic controller node. Sections 3 through 5 describe these aspects. Section 6 provides an integrated view. Booch [6] describes two perspectives of a system: algorithmic decomposition and object-oriented decomposition, with the latter being the driving perspective. These perspectives correspond to the functional decomposition and inheritance perspectives of the reference model architecture that this paper describes. The most significant difference in our concept is that the level of authority perspective drives the system design while adhering to the generic reference model.

3. The functional/processing decomposition dimension--an intelligent machine model

As described earlier, a generic controller node resides at the origin of the coordinate system and is functionally described by an intelligent machine model [9]. The functional decomposition notion means that a node is decomposed into finer and finer functions extending farther along this axis.

In Figure 2, the origin of the axis shows a controller node as a whole. It permits interaction from an operator. At the second unit, a node is represented by sensory processing (SP), world modeling (WM), behavior generation (BG), and value judgment (VJ) functions, as described below:

The sensory processing (SP) function samples sensory data, filters and integrates sensory information over space

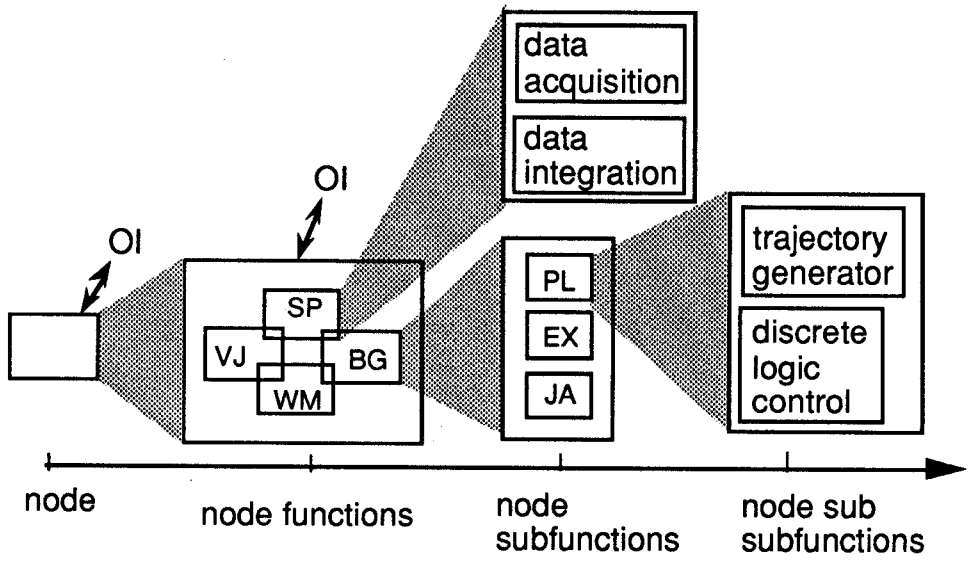


Figure 2: Functional/processing decomposition of a controller node

and time, recognizes patterns, and detects events. In other words, as illustrated at the third unit of Figure 2, SP is decomposed into subfunctions including data acquisition and data integration. The following paragraph provides an example:

Next Generation Inspection System (NGIS) Project: ISD is involved in a manufacturing inspection system project [14] sponsored by the NGIS consortium, which itself is organized by the National Center for Manufacturing Sciences. In the project, the SP includes the integration of vision and touch probe data. This intends to overcome deficiencies in current coordinate measurement machines (CMM) which must be pre-programmed for each part being measured (a time consuming process), and which can not distinguish areas of interest (edges, holes, grooves, etc.) from less interesting areas (flat surfaces).

The world model (WM) function conceptually models the state space of the system, which includes maintaining the knowledge base for a node in real-time. The following is a set of state variables that a node WM maintains:

* The plan information: the set of plans (see section 4.2.2) that a node is capable of performing, the name of the plan that is being executed, and the current state of execution.
* The node status: typical status values are Reset, Executing, Done, Waiting, Error, and Emergency stop.
* The error code: typical errors are mismatch of command or status between senders and receivers, correspondent nodes not responding, time out, etc.
* The performance indices: Timing and frequency of occurrence.
* The data to be shared by other nodes: object position, map with proper resolutions, etc.

The WM also stores system parameters such as inertia matrices and forward and inverse kinematic transformation.

Within an application, the node WMs are distributed throughout the system.

The value judgment (VJ) module computes the costs based on determined criteria (safety, schedule, etc.) for the planning activities (see the Planner below).

The behavior generation (BG) function is responsible for planning and executing the tasks that a node receives from its superior node at the higher authoritative level (see section 4). The output tasks are sent to its subordinate nodes at the lower level as input commands.

The third unit along the axis describes that the above node major functions can be decomposed into subfunctions. BG contains the following subfunctions: the planner (PL), the executor (EX), and the job assigner (JA).

Planning typically involves hypothesizing actions, requesting WM to predict the results of these actions, requesting VJ to compute the costs of the actions, and selecting the best plan. The plan is then executed by EX through servoing the control variables and computing output commands for the next level nodes. JA partitions the output commands generated by EX and sends them to the lower level nodes.

As shown in Figure 2, the fourth unit along the axis yields another layer of functional decomposition. The following paragraph illustrates a decomposition of PL:

Technologies Enabling Agile Manufacturing (TEAM) research effort: The ISD TEAM research effort describes multiple planning functions [15]. A trajectory generator is responsible for the coordinated control of multiple continuous-valued devices -- as opposed to the discrete logic control which is responsible for the coordinated control of discrete-valued devices. The amount of planning sophistication depends on the coordination and the synchronization between the two types of state variables.

A node is subject to operator interaction (OI). An operator may send commands to the BG or request WM data for display.

For detailed descriptions of the node functions, see [9, 12].

4. The level of authority dimension--hierarchical levels

In a complex automation system, multiple controller nodes, as described above, are distributed and are authoritatively connected to form a hierarchical architecture. The dimension described in this section shows how the authority chain is formed.

4.1 levels of authority

RCS predefines the following hierarchical levels as the guidelines for partitioning a hierarchical system:

Level 6 -- problem domain level, also called facility or mission level. This is the highest level. The controller receives overall commands, from the user, for the entire control system. The BG function of this node decomposes these commands and outputs the results to the responsible next level controllers.

Level 5 -- group level. Multiple groups of equipment (see level 4) may exist and they must be coordinated at this level. For example, a manufacturing production line may have multiple workstations. A workstation may have multiple pieces of equipment. The group level, therefore, contains these workstation controllers.

Level 4 -- equipment, or task level. A node at this level typically models a major physical entity, for example, a submarine. Tasks received by the controllers at this level concern how each piece of equipment is expected to perform to accomplish a system goal.

Level 3 -- elementary move (emove) level. The emove level is the kinematic control level. Any task is decomposed into a series of subtasks that are free of kinematic limits, singularities, and obstacles. Sensor data submitted from the primitive level (see below) may be combined to produce surface features, feature distance and relative orientation, etc.

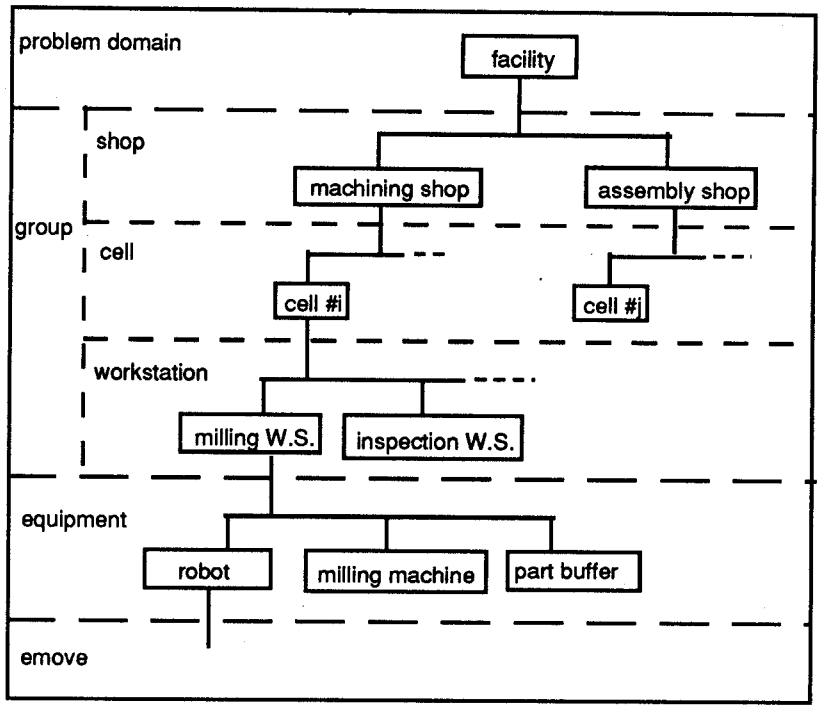


Figure 3: The AMRF Hierarchical Level of Authority Model

Level 2 -- primitive (prim) level. The primitive level is the dynamic control level. The kinematically sound tasks are computed for sub tasks that are dynamically smooth. The SP function integrates and fuses data gathered from individual sensors and produces linear features for objects.

Level 1 -- actuator level. The controller nodes at this level interact with the environment. The BG generates electrical, hydraulic, or mechanical commands to activate the actuators. The SP function for this level is to receive signals from each individual sensor and to process them.

4.2 Tenets of this dimension

4.2.1 Canonical form

The controller nodes repeat and extend themselves in the context of the intelligent machine model to a level sufficiently high to describe a system. A unified execution behavior is exhibited across all the controller nodes at all the levels.

This canonical form also implies the flexibility of the number of nodes at the levels and the number of sublevels within a level. Each level can have none or multiple nodes except for the highest level where there is one node. Some problems may require only on-off types of control and may not require a dynamic (prim) level. When the tasks are complex enough to warrant another level of decomposition between a pair of predefined parent and child levels, multiple sublevels may be needed. The following illustrates this fact:

The NIST Automated Manufacturing Research Facility (AMRF) [16]: As seen in Figure 3, a manufacturing facility contains multiple shops. A shop may contain multiple cells. A cell may have multiple workstations. A workstation may have multiple pieces of equipment. There are three sub levels within the group level.

4.2.2 Execution of system goals via task decomposition

The behavior generation (BG) function of the node(s) at each level receives the commands (tasks) from the level above, decomposes them, and outputs the results to the responsible next level controllers. In other words, controllers at a particular level coordinate the execution of the next lower level controllers. The following project illustrates this activity:

Submarine Automation [17]: See Figure 4. A submarine is encountering a problem while transiting undersea and needs to rise to its periscope depth. The Run_Mission_5 command is given to the command controller (CC). This command is decomposed into Prep_Emergency_Vent, Submerged_Transit, Submerged_Vent, and Emergency_Vent commands. The exact behavior involving these four commands is shown in the state diagram, called a plan in the RCS architecture. Through a series of task decompositions such as this down the hierarchy, each controller receives commands and executes its plans. Actuator control signals are issued at the lowest level to control the turbine, clutches, sail and stern planes, etc.

4.2.3 Resolution, temporal span, and spatial span

Higher levels deal with tasks and data that have less resolution but longer time and wider spatial span. Lower levels deal with tasks and data that have more resolution but shorter time and narrower spatial span.

5. The inheritance dimension--reference model to application

As shown in Figure 5, along this axis, the desired functionality of models at the left of the axis is inherited by models at the right. Kramer et al., [18] introduced a similar concept called multiple tiers of architectures. RCS, being a reference model architecture, implies that the properties of the intelligent machine model, described in section 3, is inherited by any class of problems using the architecture. Manufacturing control systems is one class of problems. Any discrete parts manufacturing control system RCS inherits properties developed for the manufacturing control system RCS. This inheritance relationship can extend to many layers. Each layer can contribute commonly useful software library sets. This fact makes this architectural implementation process efficient and makes the RCS development environment rich. The following paragraph illustrates this:

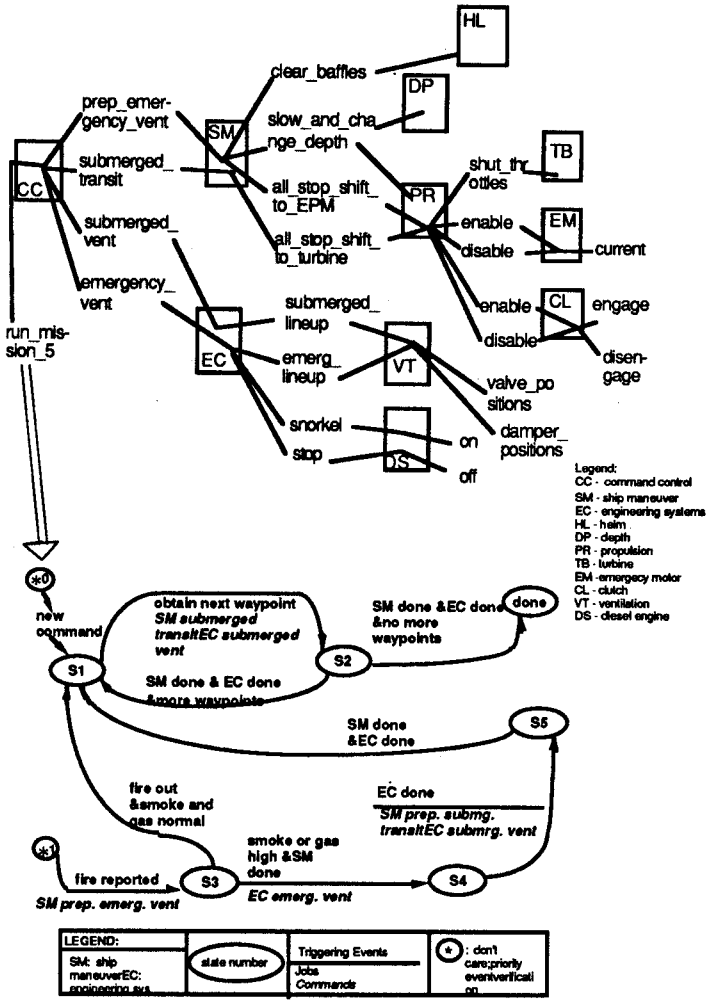


Figure 4: Task Decomposition and Behavior Description

AutoBody Consortium Project: A series of command classes are derived in the object oriented notion. As seen in Figure 6, emove command classes for a Robotics Research Corp.[1] (RRC) robot are developed and common features are inherited [19].

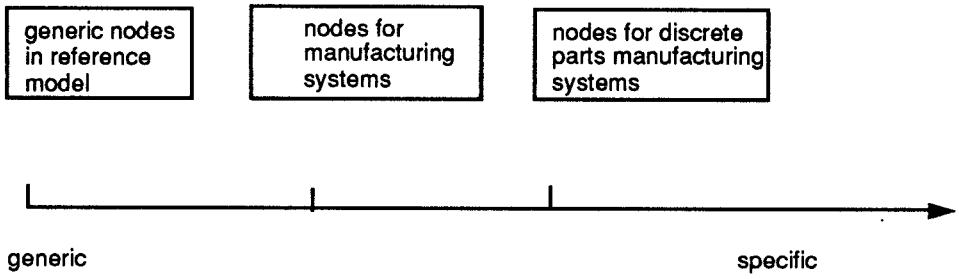


Figure 5: The Inheritance Dimension

[1] References to company or product names are for identification only and do not imply NIST endorsement.

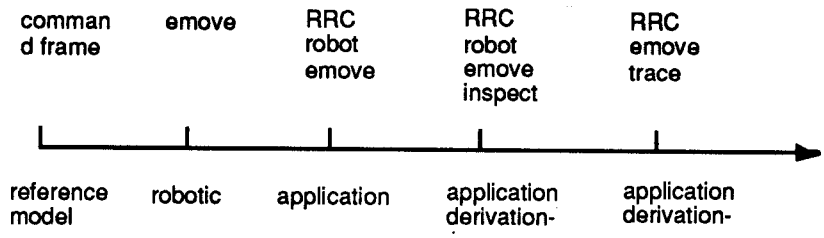


Figure 6: RRC robot emove command classes

6. The Integrated Framework

The described multiple dimensional framework provides an open mechanism and canonical form that are scalable from multiple perspectives. Automation and intelligent control systems of any size can be represented on the framework. The inheritance feature facilitates software reuse. The use of a reference model expedites system development. Therefore, we anticipate that this framework shortens development cycles of the manufacturing systems. Figure 7 illustrates these features. An RCS architectural implementation can be viewed as a hierarchy tree rooted on a notch on the inheritance axis and growing in parallel with the authority axis. The tree leaves represent controller nodes, which can be decomposed along the functional axis.

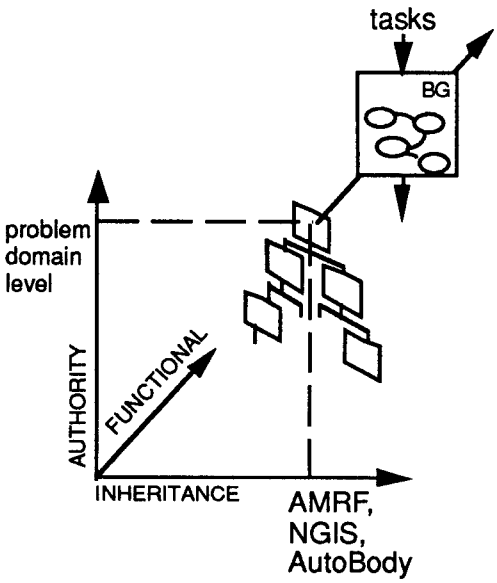


Figure 7: The Integrated Framework

The knowledge base, or world model (WM), of an entire implementation system is formed by the integration of each individual node WM. The controller hierarchy and task structure serve as the references for the organization of the node task knowledge.

The projects concerned with issues related to an integrated framework include:

Systems Integration for Manufacturing Applications (SIMA) Joint Architecture project [18]: The goal of this project is to incorporate manufacturing process planning, control, information management, and communication aspects to obtain, in detail, a generic architecture for discrete parts manufacturing facilities. This project also emphasizes the inheritance aspect.

The NIST ISD Tools Project: A specification for a RCS computer-aided control system development (CACSD) tool is being investigated. The goal is an automated environment facilitating the development of intelligent and automation systems covering the very early conceptualization stages through the final critical mission real-time operation stages.

7. Summary and Future Directions

We have presented a multiple dimensional reference model architecture called RCS to facilitate manufacturing system automation and intelligent system control. RCS has the following advantages:

* An open architecture with rich representations.
* A distributed and efficient structure for integrating and organizing system knowledge.
* A framework facilitating information sharing and software reuse.

We expect to continue exploring this framework. We have demonstrated that various advanced projects possess various features of the framework. Sharing and inheriting common knowledge among projects can be expedited once their relationships on the coordinate system are established. The projects described in this paper are part of NIST's on-going collaborative effort with government, industry, and academia to pursue open-architecture controllers. The objective is to alleviate the problems associated with incorporating proprietary technologies in industrial applications and to reduce product costs.

Architecture concerns additional hardware and software issues that are not covered in this paper, for example, the use of industrial de facto software and hardware standards for the control architecture. We expect to experiment with these additional issues under this framework. Our goal is

a fully described open and scalable architecture that will benefit the manufacturing industry and research and development community.

8. Acknowledgment

Dr. James Albus of NIST and Dr. Anthony Barbera of the Advanced Technology Corporation have been leading the research and application of RCS since they originated it at NIST two decades ago. Harry Scott reviewed the manuscript and provided technical insights.

---

9. References

1 Requirements of Open, Modular Architecture Controllers for Applications in the Automotive Industry, ver. 1.1, a white paper issued by Chrysler, Ford, and GM Corporations, December, 1994.

2 Workshop on Advanced Machine Tool Structures: Research Directions, a workshop report published by the Manufacturing Engineering Laboratory of NIST, Gaithersburg, MD, April, 1994.

3 Sweet, W., et al., Recommendations from the AIA/SEI Workshop on Research Advances Required for Real-Time Software Systems in the 1990s, Special Report SEI-89-SR-18, Cargenie-Mellon University Software Engineering Institute, Pittsburgh, PA, September, 1989.

4 Fleury S., et al., "Design of a Modular Architecture for Autonomous Robot," International Conference on Robotics and Automation, San Diego, CA, May 94.

5 Simmons, R., et al., "Autonomous Task Control for Mobile Robots," Proceedings of the Fifth International Symposium on Intelligent Control, Philadelphia, PA, September, 1990.

6 Booch, G., Object-Oriented Analysis and Design with Applications, 2nd Edition, The Benjamin/Cumming Publishing Company, Inc., Redwood City, California, 1994.

7 Kogut, P. and Clements P., "The Software Architecture Renaissance," Cross Talk--the Monthly Technical Report of the United States Air Force Software Technology Support Center, Hill AFB, Utah, November, 1994.

8 Coad, P. and Yourdon, E., Object Oriented Analysis, Yourdon Press Computing Series, Prentince Hall, Inc., Englewood Cliffs, New Jersey, 1991.

9 Albus, J.S., "Outline for a Theory of Intelligence," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 21, No. 3, May/June 1991

10 Huang, H-M, "Outline of a Multiple Dimensional Reference Model Architecture and a Knowledge Engineering Methodology for Intelligent System Control," Submitted to IEEE Expert Special Track on Intelligent Control, February, 1995.

11 Albus, J.S., McCain, H.G., and Lumia, R., "NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)," NBS Technical Note 1235, National Bureau of Standards, U. S. Department of Commerce, April, 1989

12 Huang, H., Quintero, R., and Albus, J.S., "A Reference Model, Design Approach, and Development Illustration toward Hierarchical Real-Time System Control for Coal Mining Operations," Book Chapter in Control and Dynamic Systems, Advances in Theory and Applications, Volume 46, Academic Press, 1991.

13 Albus, J.S., Juberts, M., Szabo, S., "RCS: A Reference Model Architecture for Intelligent Vehicle and Highway Systems," ISATA 92, Florence, Italy, June 1992.

14 Nashman, N., et al., "An Integrated Vision Touch-Probe System for Dimensional Inspection Tasks,," Submitted for the Workshop for Robot Vision, August 1995.

15 Michaloski, J., Technologies Enabling Agile Manufacturing (TEAM) Document: Application Programming Interfaces (API); Draft NISTIR, Gaithersburg, MD, January, 1995.

16 Albus, J.S., et al., "A Control System for an Automated Manufacturing Research Facility," Robots 8 Conference and Exposition, Detroit, MI, June, 1984.

17 Huang, H., Hira, R., and Quintero, R., "A Submarine Maneuvering System Demonstration Based On The NIST Real-Time Control System Reference Model," Proceedings of The 8th IEEE International Symposium on Intelligent Control, Chicago, Illinois, 1993.

18 Kramer, T. R., et al., "A Reference Architecture for Control of Mechanical Systems;" in proceedings 1994 Tutorial and Workshop on Systems Engineering of Computer-Based Systems; Harold W. Lawson, editor; IEEE Computer Society Press; 1994; pp. 104 - 110.

19 Tarnoff, N., et al., "A Visually Intensive Lifecycle Framework for Robotic Applications," The First World Congress on Intelligent Manufacturing Process and Systems, Puerto Rico, February, 1995.