NISTIR 5643

Outline of a Multiple Dimensional Reference Model Architecture and a Knowledge Engineering Methodology for Intelligent Systems Control

Hui-Min Huang
Unmanned Systems Group
Intelligent Systems Division

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Bldg. 220 Rm. B124
Gaithersburg, MD 20899

NIST

NISTIR 5643

Outline of a Multiple Dimensional Reference Model Architecture and a Knowledge Engineering Methodology for Intelligent Systems Control

Hui-Min Huang
Unmanned Systems Group
Intelligent Systems Division

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Bldg. 220 Rm. B124
Gaithersburg, MD 20899

April 1995

U.S. DEPARTMENT OF COMMERCE
Ronald H. Brown, Secretary

TECHNOLOGY ADMINISTRATION
Mary L. Good, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Arati Prabhakar, Director

Submitted to the IEEE Expert Magazine, February 1995.

OUTLINE OF A MULTIPLE DIMENSIONAL REFERENCE MODEL ARCHITECTURE AND A KNOWLEDGE ENGINEERING METHODOLOGY FOR INTELLIGENT SYSTEM CONTROL

Hui-Min Huang
Mechanical Engineer
National Institute of Standards and Technology
Gaithersburg, MD 20899
email: huang@cme.nist.gov

Abstract

We outline a multiple dimensional reference model architecture and a methodology for representing and developing intelligent systems. The reference model architecture features multiple dimensions enabling modeling the multiple aspects of complex systems. The canonical form within this framework facilitates open and scalable system architecture. The well-defined structures facilitate efficient knowledge engineering processes. We describe a submarine automation model performing real-time control to illustrate the application of this reference model architecture.

1. Introduction

Large scale intelligent control systems pose unique challenges in computer software and hardware technologies for researchers. These systems often conduct critical missions. They commonly require the capability of real-time access to knowledge bases to meet the millisecond level control cycle requirements. Researchers have begun to address some aspects of this complex problem domain. Antsaklis [1] pointed out that intelligent systems typically involve hierarchical architectures and that certain levels exist in the hierarchies to handle predefined functions. Sweet et al. [2] identified that large scale, real-time, and distributed are among the key software technologies for the Aerospace Industries Association (AIA). Strassmann [3] also pointed out that, "The rapid deployment of information systems in the future under unpredictable and often hostile conditions calls for easily repairable software that is constructed from reliable standard components." In the areas of computer-aided software engineering environments and architectures, Simmons [4] describes a Task Control Architecture (TCA). However, scalability might be a problem for this architecture as it is not intended to model multiple cooperating agents. Object oriented paradigms [5, 6] are becoming popular for handling the representation problems of software systems. As Coad [6] pointed out, however, they are not suitable for all problems. Schneider, et al. [7] developed a flexible object oriented real-time software implementation tool called ControlShell[1]. However, this tool does not intend to address the architecture issue and it seems as if a reference model architecture can complement the capability of ControlShell.

Intelligent system control has been the research focus of the Intelligent Systems Division (ISD) of the National Institute of Standards and Technology (NIST). NIST ISD proposes that a comprehensive approach toward this intelligent control system problem should cover all of the following critical issues:

* A scalable and open architecture.
* A rich and representative reference model.
* A distributed and efficient structure for

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

organizing system knowledge.
* A rigorous knowledge engineering process and modeling paradigm.
* A comprehensive computer-aided rapid development and deployment environment.
* Real-time control and operator interaction capability.

The particular approach that the NIST ISD has been using is called the Real-time Control System (RCS) reference model architecture [8]. Researchers in NIST ISD and elsewhere have been applying RCS to various large scale intelligent control systems, including [9, 10, 11] since two decades ago. The ultimate goal for NIST ISD is for RCS to evolve into a unified solution paradigm to the problem domain of intelligent system control.

This paper attempts to outline RCS as a multiple dimension reference model architecture for use with intelligent systems. This paper also outlines an application development methodology for RCS which emphasizes knowledge engineering and task analysis. A submarine automation model provides an illustration.

2. Multiple Dimensional Reference Model Architecture

The NIST RCS is a reference model architecture. Research results [9, 10, 11] 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 2.1.1). 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 [6].

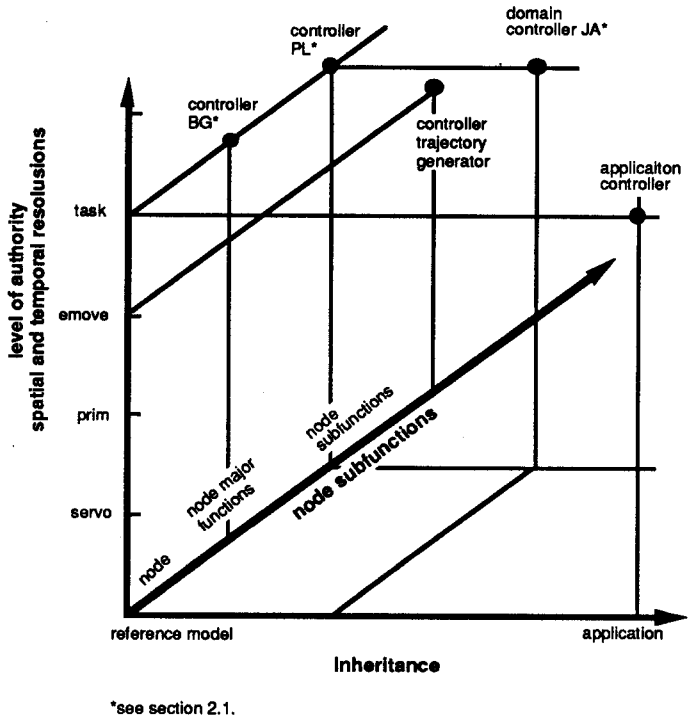

Figure 1: A Multiple Dimension Reference Model Architecture for Intelligent Systems

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 [5] describes these two perspectives. In previous research efforts, NIST ISD has successfully explored the task, or level of authority, based hierarchy (see section 2.2) and a functional view of RCS (see section 2.1.1). NIST ISD has also been developing generic software templates and libraries that can be inherited by new RCS applications. Efforts are required to integrate all these aspects together to form an integrated view. This is a major issue that we intend to address in this paper. Our result is shown in Figure 1, which

describes that the three paradigms of: level of authority, functional decomposition, and inheritance form a multiple dimensional reference model architecture. These features characterize RCS. At the origin of the coordinate system is a generic controller node, which serves as the building block of RCS and is described by an intelligent machine model. Sections 2.1 through 2.3 describe these aspects. Section 2.4 describes how an implementation architecture is identified within the reference model architecture. Booch [5] 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 referencing the generic reference model.

2.1.1 An intelligent machine model

In Figure 2, the origin shows that a node 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 and time, recognizes patterns, and detects events.

The world model (WM) function conceptually models the state space of the system, including maintaining the knowledge base for a node in real-time. In this paper, the terms knowledge base, state space, and world model are used interchangeably. WM also estimates and predicts world states for the planning and sensory processing purposes. See section 2.1.2 for more detail.

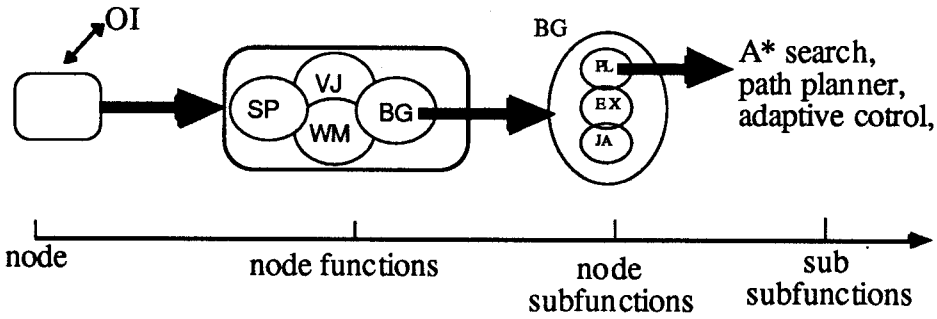

Figure 2: Functional Decomposition of a Controller Node

2.1 The functional 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 [8]. The functional decomposition notion means that a node is decomposed into finer and finer functions extending farther along this axis.

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 2.2). The output tasks are sent to its subordinate nodes at the lower level as input commands.

The value judgment (VJ) module determines the costs during the planning stage (see the Planner below).

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). Planning typically requires the evaluation of alternative hypothetical sequences of planned subtasks. The planner hypothesizes some action or series of actions. WM predicts the results of the action(s). VJ computes the costs of the action(s).

The executor (EX). EX executes the plans prepared by the planner by servoing the state variables and computing output commands for the next level nodes.

The job assignment manager (JA). JA partitions the output commands generated by EX and sends them to the lower level nodes.

The fourth unit along the axis yields another layer of functional decomposition. For example, particular path planners or search algorithms may be included in a PL.

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 [8, 10].

2.1.2 The state space

The following information is a typical set of knowledge maintained by WMs in a distributed fashion, but available system-wide.

Node state space:

* The plan information: the set of plans 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 performance is typically the most critical index. A node can maintain the following timing information: last cycle execution time, maximal execution time, minimal execution time, averaged execution time (moving average), and execution time trend data.

* The data to be shared by other nodes: object position, fuel level, etc.

Control system state space:

The integration of each individual node state space constitutes the control system state space. The controller hierarchy and task structure serve as the references for the organization of the node state space information. The submarine automation model illustrates this concept in section 3.

2.2 The level of authority dimension--hierarchical levels

RCS is a hierarchical architecture. Controller nodes are distributed and are authoritatively connected across the levels along this axis. Sections 2.4 and 3.4 provide an example.

2.2.1 Levels of authority

In RCS, the following levels are predefined 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 (E-move) Level. The e-move 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 the next paragraph) may be combined to produce surface features, feature distance and relative orientation, etc.

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.

2.2.2 Tenets of this dimension

Flexibility of the number of nodes at the levels: 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. On the other hand, some problems may need multiple sublevels within a predefined level. This may happen when the tasks are complex enough to warrant another level of decomposition between a pair of predefined parent and child levels. It may also happen when the physical environment contains multiple layers of natural boundaries between a pair of predefined parent and child levels. For example, a manufacturing facility may have multiple production lines. A production line may have multiple workstations. A workstation may have multiple pieces of equipment. There are two sub levels within the group level.

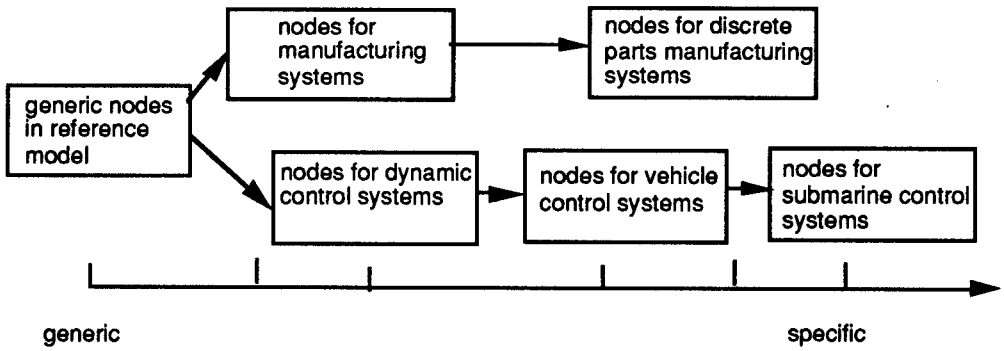

Figure 3: The Inheritance Dimension

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 exhibits across all the controller nodes at all the levels.

Execution of system goals via task decomposition: The behavior generation (BG) function of the node(s) at each level receives the commands 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.

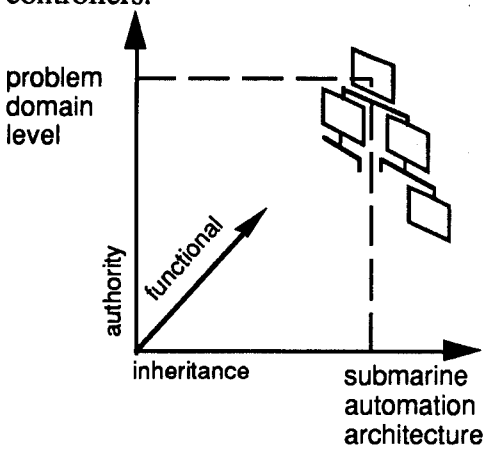

Figure 4: Identifying an Architectural Implementation

Resolution, temporal span, and spatial span: High levels deal with tasks and data that have less detail but longer time and wider spatial span. Lower levels deal with tasks and data that have more detail but shorter time and narrower spatial span.

2.3 The inheritance dimension-- reference model to application

The concept along this axis is that the desired functionality of models at the left of the axis is inherited by models at the right, shown in Figure 3. Kramer et al., [12] 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 2.1, is inherited by any class of problems using the architecture, for example, the dynamic control system class or manufacturing control system class. Any vehicle control RCS inherits properties developed for the dynamic 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.

2.4 Identifying an architectural implementation within the reference model

An RCS architectural implementation, see Figure 4, 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.

3. RCS Methodology: A Task Oriented Knowledge Engineering Process

3.1 The knowledge evolution process

Architectural implementation involves understanding, sorting, assimilating, and integrating domain knowledge and systems, computing, and control knowledge using a systematic approach. RCS prescribes a task oriented knowledge engineering process, which is highlighted in Figure 5 as a knowledge refinement process. Note that, the multiple dimensional reference model architecture (section 2) does not become involved until the middle stages of the process. Knowledge for a problem domain and knowledge in the computing, systems, and control fields are, to a large extent, unrelated in their raw forms, shown as the two separate branches at the upper half of the drawing. In the second box of the left branch, the RCS methodology calls for interactions with domain experts to identify a subset of the knowledge that is within the scope of project requirements. This exercise is important in the sense that it may result in clarifications or modifications of project requirements. Another step of knowledge refinement process further deduces the domain knowledge to a set of

inclusive written operational scenarios. The developers perform a task analysis based on the scenario descriptions. Quintero [13] also describes these aspects in detail.

In the submarine project, the NIST ISD researchers collaborated with a retired submarine commander. Based on the pre-established goals of the simulated

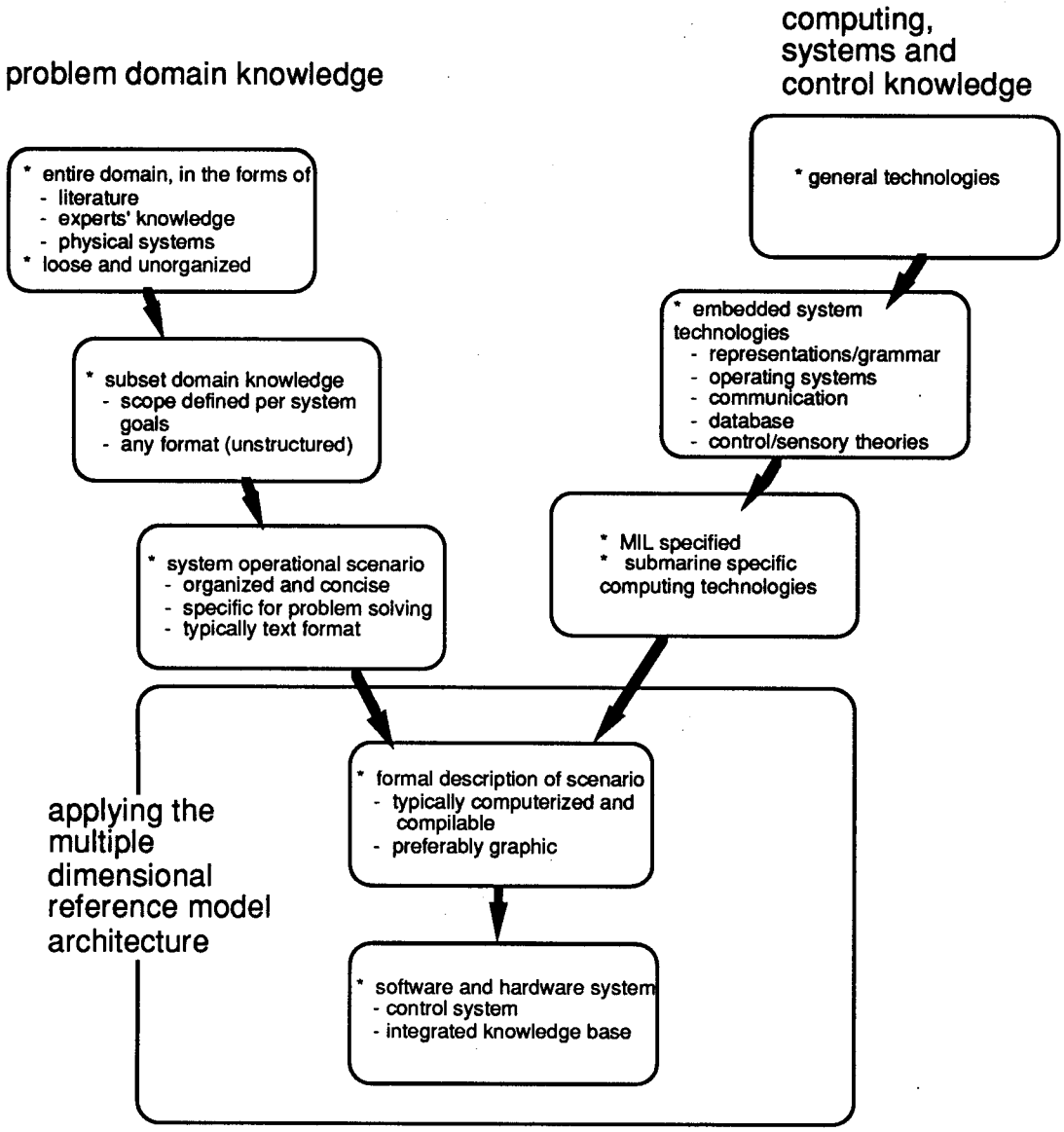

Figure 5: A Knowledge Evolution Process

submarine, he described to us, in detail and in every aspect, how a submarine would operate to achieve the specified goals. The following are some of the methods used to maximize the relevant knowledge output from the domain expert. ISD researchers did not impose any structural constraints during his description. The commander was encouraged to use the submarine terminology. The main role of the ISD researchers at this stage of interaction was to be goal driven, to be intelligent listeners, and to ask stimulating questions sparsely to catalyze his stories.

At the right branch of Figure 5, a separate knowledge refinement process occurs. Note, we have not yet fully explored how the task oriented method is applied to this

particular area. Appropriate computer platforms (for both development and operations), modeling and implementation languages, operating systems, CASE tools, MIL and other standards or specifications, etc., are to be selected for use with project implementation.

Section 3.4 continues the description of this knowledge evolution process.

3.2 An illustrative scenario for the submarine automation model

A set of scenarios was developed for this project. The following concise description provides a flavor of one scenario within the set:

A submarine is conducting a submerged transit of the open ocean at its standard speed (15 knots or nautical miles/hour, equivalent to 7.7 meters/second) and at a keel depth of 200 meters. A watchstander reports that there is a lube oil fire in the lower level Engine Room. The Officer of the Deck (OOD) directs the Ballast Control Panel (BCP) operator to pass the word on the general announcing system. The OOD completes the following actions for coming to periscope depth: Clearing baffles, Checking for sonar contacts and close contacts, Slowing and changing depth, and Raising the periscope.

The damage control party fights the fire in the engine room. On indication of decreasing main lube oil pressure the OOD orders "All stop, shift propulsion to the EPM (emergency propulsion motor)." The shaft rotation is stopped and the clutch is used to disengage the shaft from the turbines and the EPM circuit breaker is closed. The Engineering Officer of the Watch (EOOW) reports to the OOD that he is prepared to answer bells on the EPM. The OOD orders "Ahead two thirds" which maintains enough speed for depth and steering control.

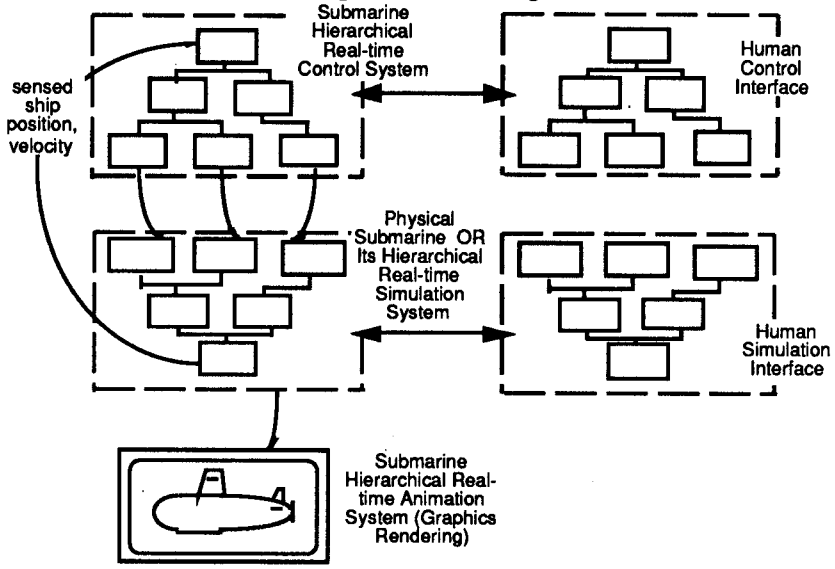

Figure 6: An Analysis and Implementation Software Structure

The damage control party reports that the fire is out. The BCP selects the ventilation lineup and sets it to emergency ventilate the engine room using the diesel engine. When the lineup is proper, the OOD directs "Commence snorkeling."

3.3 A proposed analysis and implementation structure

In our methodology, a generic and comprehensive logical structure is proposed to facilitate the analysis and implementation of this multiple dimension RCS reference model architecture. This logic structure is considered a software technology specific to RCS and, therefore, belongs to the lower portion of Figure 5. This structure identifies and integrates the following five areas (see Figure 6):

* Control hierarchy. An architectural implementation, described in section 2.4, produces a control hierarchy to perform real-time system control to accomplish missions. Figure 9 is an

illustration of this hierarchy. The lowest level nodes send electrical, hydraulic, or mechanical control signals to either physical actuators or simulators (see below).

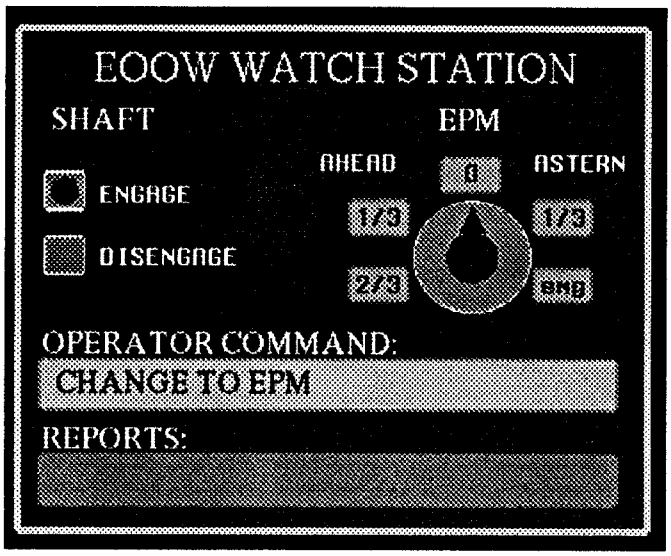

Figure 7: A Graphic Operator Interface Panel

* Simulation hierarchy. This hierarchy is required to facilitate conceptualizing and testing a control hierarchy. The lowest level nodes, the actuator simulators, receive control signals and compute for the mechanical movements of each individual actuator. The higher level nodes integrate the simulated actuator movements, compute the ship level dynamics, and obtain the ship movements. The results are fed back to the above control hierarchy via simulated sensors. The dynamics of the relevant environmental objects, for example, sea water salinity and air contamination in the submarine engine room, is also simulated. See section 4 for a detailed illustration.

* Animation hierarchy. This hierarchy performs graphic rendering only, at or close to, real-time, based on the simulation results.

* Control operator interface. As shown in Figure 2, nodes can be interacted by operators. This hierarchy is proposed for the interaction purposes. This setup facilitates multiple control modes: manual, autonomous, and hybrid. It also allows emergency control overrides from an operator.

Figure 7 is a graphic operator interface

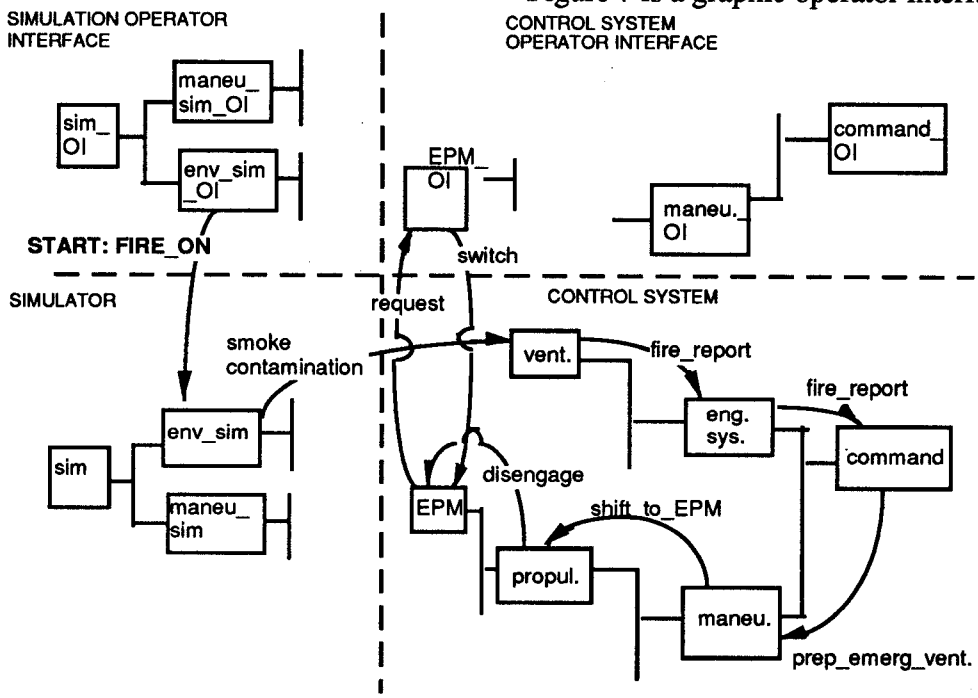

Figure 8: Task Analysis Diagram

panel implemented for the submarine automation model. An operator, called Engineering Officer of the Watch (EOOW), can switch the propulsion

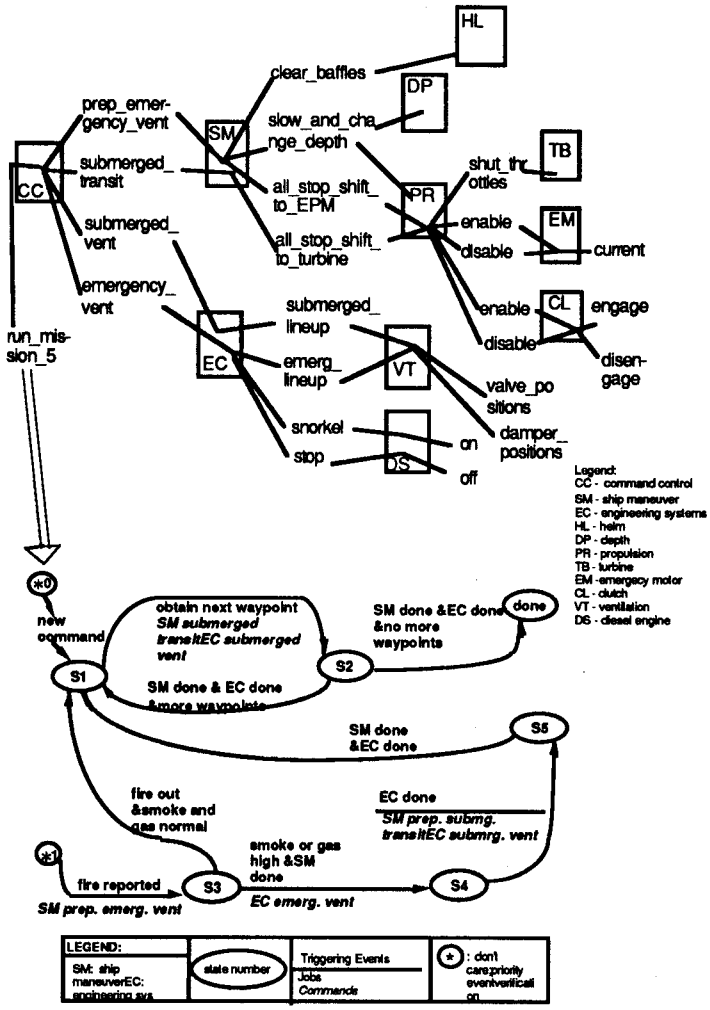

Figure 9: Illustrative Task Tree and Task Plans

control to Emergency Propulsion Motor (EPM) under certain circumstances. See section 3.2 for an operational scenario.

* Simulation operator interface. This is a different kind of operator interface. Simulated events, such as the occurrence of a lube oil fire, are injected via this hierarchy.

Current results indicate the five hierarchies might exhibit one-to-one correspondence to each other. Section 4 illustrates this effect.

3.4 Task analysis

After the operational scenarios are obtained, the two knowledge areas merge, as shown in Figure 5. The developers perform task analysis, based on the scenario descriptions, to further formalize the system knowledge. The researchers first lay out the four quadrants of a task analysis diagram (Figure 8), corresponding to the four hierarchies described in Figure 6. The researchers then identify the control system tasks and simulation system events (described in detail later in this section), which are essentially the verbs in the scenario descriptions. A formal modeling representation, namely finite state machine, is then used to describe control system behaviors--the sentences that the verbs are associated with in the scenario descriptions. The state diagrams are then organized in the RCS hierarchy, as illustrated in Figure 9. Finally, a C language controller node software template [14] is used to code the software. These activities are shown in the bottom two boxes in Figure 5. The templates are a part of the NIST ISD effort to explore the inheritance dimension of the multiple dimensional reference model architecture (Figure 1).

Figure 8 illustrates this task analysis activity. Refer to Figure 6 for the four quadrants. Arrows show how we identify and sequence a series of tasks, nodes, and events. Note that a control system is to be developed to replace the manual operations described in the scenario. On the upper left corner, a fire_on event is identified per scenario. During simulation, this event is to be injected through a simulation operator

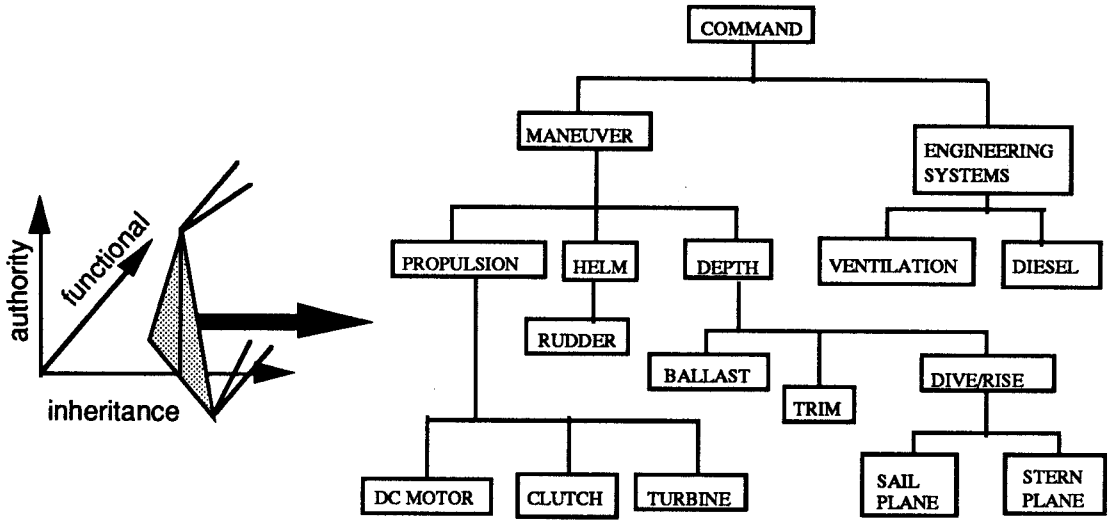

Figure 10: An Architectural Implementation

interaction node. Therefore, the env_sim_OI node is identified. The corresponding simulator node computes the changes in the air constituents (see smoke_contamination in figure) which must be detected by a simulated sensor installed in the vent controller. This illustrates how developers realized that the particular sensors and ventilation controller are required. The fire report is sent up the controller hierarchy to the highest level controller which has the responsibility of coordinating the engineering system and maneuvering controllers. From the scenario, one of the commands Maneuver will receive is prep_emerg_vent. A propulsion controller is required to handle the switch to the EPM control. In our control system configuration, this particular switching operation is to be done manually. Therefore, a request is sent to the controller operator interface hierarchy. The operator reads the message (see Figure 7), disengages the main shaft, and reports when it is done. Many iterations of these event, node, and task identification activities are required to finalize the multiple hierarchies.

Figure 10 shows the control hierarchy of our submarine model in the context of the multiple dimension reference model architecture described in section 2.

Figure 9 proceeds another step further by describing the behavior for each of the identified tasks. Note that analytical algorithms can be integrated into these discrete event algorithms as processing jobs.

4. Real-time Control, Simulation, and Operator Interaction

As described earlier, our RCS methodology provides a behavior oriented analysis method that allows describing the internal model of a system to any sufficient level of detail. This analysis produces a structure that is described via an organization hierarchy, a task tree, and behavior diagrams, as described in section 3. Once the structure is in place, the necessary supporting data, algorithms, simulation, sensors, and operator interface can be identified. The same concept is extended to the development of the simulation structure. The result is a hierarchical simulator. Such a process facilitates sensory data analysis for RCS controller units. It also enables incremental testing of the control hierarchy.

The depth control and simulation for the submarine is described here as an illustration.

In Figure 11, a mission command given to the commander includes the depth requirements. They must be converted to the electrical signals for the sail and stern planes. Intermediate levels are needed to provide smooth transitions between these two extremes. The intermediate levels facilitate human understanding, computation efficiency, and control stability. The command controller passes the depth requirements of the mission, through a series of way points, down the hierarchy to the Maneuver controller. Maneuver computes the required ship depths accordingly and passes them down for the Depth controller. Depth computes a series of bubble angles required for controlling the ship to the specified depths. The Dive/Rise controller computes required plane angle maneuvers, for the Sail and Stern Plane controllers, to achieve the required bubble angles. The plane controllers generate required electrical signals for the control valves to move the planes to the commanded angles.

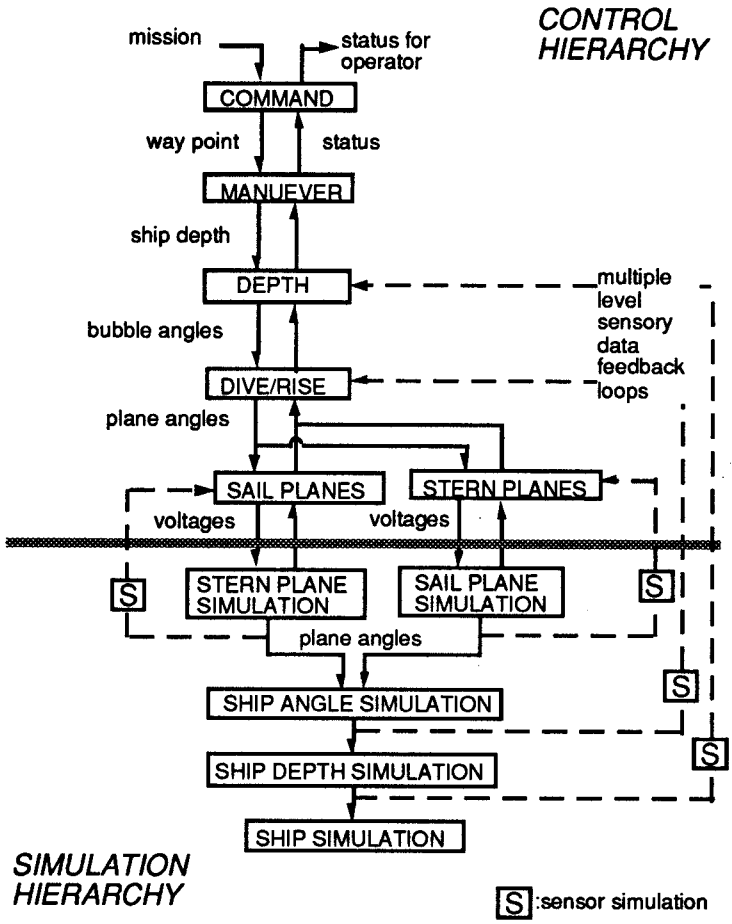

Figure 11: Nested Depth Control and Simulation in Submarine Automation

The submarine depth simulator is developed as a reverse of the control hierarchy, as seen in the lower portion of Figure 11. The only input that the simulator receives from the controllers is the commanded electrical signals. The hydrodynamic model for the submarine is decomposed and distributed in the simulator hierarchy. At the lowest level (shown at the top of the simulator), the electrical signals are used to compute for the simulated plane angles, which are integrated at the next level to form simulated ship bubble angles. At the next level, the dynamic model uses the ship angles to compute the ship depth. All these intermediate results may be used as sensor input to feed back to the appropriate controllers.

5. Conclusion and Future Directions

A multiple dimension reference model architecture has been presented. Multiple dimensions in the architecture facilitate the descriptions of complex systems from multiple perspectives. This provides a framework for open and scalable system architecture. A knowledge engineering methodology has been described and illustrated. This process describes systematic structures to organize system knowledge and describes a series of smooth transformations to assimilate system knowledge from raw forms to computer executable forms. A submarine automation model is used to illustrate the application of this RCS reference model architecture and its knowledge engineering methodology. The simulation demonstrated the model

performing intelligent tasks by using the well-structured operational knowledge.

Work has begun at NIST ISD and with outside researchers to automate this process. A Joint Architecture project has been ongoing to describe, in detail, a generic architecture for discrete parts manufacturing facilities. A specification for a RCS computer-aided control system development (CACSD) tool is being investigated. The achievement of such a tool would greatly enhance the inheritance aspect of the architecture. Software templates have been generated to speed up implementations. The ultimate goal is an automated environment facilitating the development of intelligent systems covering from the very early conceptualization stages to the final critical mission real-time operation stages.

6. Acknowledgments

Dr. James Albus and Dr. Anthony Barbera have been leading the research and application of RCS since they originated it at NIST two decades ago. Captain Robert Lowell and Dr. Ed Carapezza of ARPA have been the sponsors of the submarine project at various stages and provided valuable technical insights to the problem. Keith Young was our principal submarine domain expert. Richard Quintero, Ron Hira, Ross Tabachow, and Will Shackleford of NIST and M. L. Fitzgerald, Nat Frampton, Philip Feldman, and Clyde Findley of the Advanced Technology Corporation have participated in various stages of the submarine project.

7. References

1 Antsaklis, P.J., Lemmon, M., Stiver, J.A., Hybrid System Modeling and Event Identification, Technical Report of the ISIS Group, ISIS-93-002, University of Notre Dame, Notre Dame, IN, 1993.

2 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.

3 Strassmann, P.A., "The Use of the Ada Computer Language: The DoD Context," Cross Talk, the Monthly Technical Report of the United States Air Force Software Technology Support Center, Hill AFB, Utah, February, 1992.

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

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

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

7 Schneider, S. A, et al., "ControlShell: A Real-Time Software Framework," AIAA Conference on Intelligent Robots in Field, Factory, Service, and Space, March, 1994.

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

9 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.

10 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.

11 Szabo, S., Scott, H.A., Murphy, K.N., Legowik, S.A., Bostelman, R.V., "High-Level Mobility Controller for a Remotely Operated Unmanned Land Vehicle," Journal of Intelligent and Robotic Systems, 5: 63-77, 1992

12 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.

13 Quintero, R. and Barbera, A.J., A Real-Time Control System Methodology for Developing Intelligent Control Systems, NISTIR 4936, 1992

14 Huang, H., Hira, R., Quintero, Q., and Barbera A., "Applying the NIST Real-time Control System Reference Model to Submarine Automation: a Maneuvering System Demonstration," NISTIR 5126, 1993.