DEVELOPMENT OF A REAL-TIME CONTROL SYSTEM ARCHITECTURE FOR AUTOMATED STEEL CONSTRUCTION

Kamel S. Saidi, Robert Bunch, Alan M. Lytle
Construction Metrology and Automation Group
National Institute of Standards and Technology
100 Bureau Drive, MS 8611, Gaithersburg, MD 20899-8611
kamel.saidi@nist.gov / robert.bunch@nist.gov / alan.lytle@nist.gov

Fredrick Proctor
Control Systems Group
National Institute of Standards and Technology
100 Bureau Drive, MS 8230, Gaithersburg, MD 20899-8230
fredrick.proctor@nist.gov

Abstract: The National Institute of Standards and Technology (NIST) is developing a real-time control system for a robotic crane (the NIST RoboCrane) that will perform automated structural steel pick-and-place operations. The control system architecture is based on the NIST Real-time Control Systems (RCS) reference model, which defines a system development methodology and a hierarchical control architecture in which system tasks and associated information are decomposed and organized into more easily manageable components or subsets. The task of picking and placing a structural steel beam was decomposed into multiple sub-task levels that must be performed in order to complete the pick-and-place operation. Tasks were decomposed from high-level operator input, such as "Install Beam A" down through the controller to the sensor and actuator level on the crane, at which, for example, voltages are computed and output to individual motors. Each level was organized into a series of control nodes each responsible for executing the sub-tasks at that given level of control. The control nodes share a common generic node model for sensor information processing, world modeling, and behavior generation. The control system development effort and its implementation on RoboCrane are presented in this paper.

Keywords: construction automation, real-time control architecture, robocrane, robotic crane, steel construction, task decomposition.

1. INTRODUCTION

The Construction Metrology and Automation Group (CMAG) at the National Institute of Standards and Technology (NIST) has been conducting ongoing research in autonomous construction since 2002. The goal of this research is to provide standards, methodologies, and performance metrics that will enable the development of advanced systems to automate construction tasks. CMAG's initial focus has been on the performance of autonomous structural steel erection, and in particular the steel beam pick-and-place operation. These efforts are also aimed at developing an Automated Construction Testbed (ACT) through which to test innovative construction technologies.

CMAG has implemented new capabilities into the NIST RoboCrane – a robotic crane developed at NIST in the early 1980's [1, 2, 3] – in order to develop and demonstrate autonomous steel construction processes. Chief among the desired capabilities is improved picking and placement of steel beams. The pick-and-place capability has been implemented into the ACT as a scaled, yet representative construction task. Specifically, CMAG recently demonstrated autonomous picking of a 7-foot structural steel beam and placement into a specially designed holder using prototype drop-in connections[1] (see Figure 1). A detailed description of the work that was conducted to give RoboCrane this capability can be found in [4].

[1] A gravity-load-only shear connection originally designed at the Lehigh University Advanced Technology for Large Structural Systems (ATLSS) Center.

This paper describes the work that is underway to develop a new controller for RoboCrane. A brief description of RoboCrane is presented next.

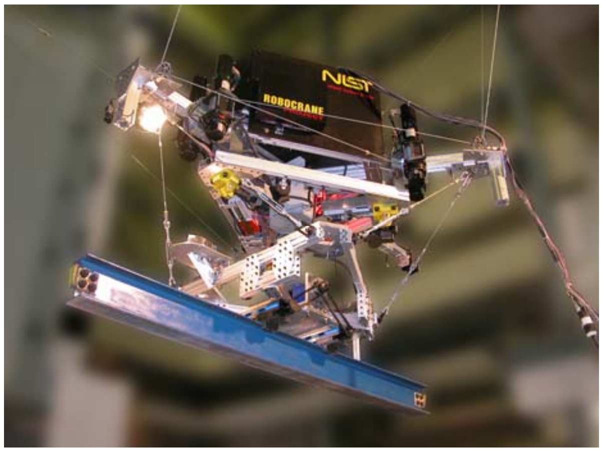

Figure 1. The autonomous RoboCrane transporting a 7 foot structural steel beam within the Automated Construction Testbed at NIST.

2. THE NIST ROBOCRANE

2.1 General Description

RoboCrane was first developed by the NIST Manufacturing Engineering Laboratory's (MEL) Intelligent Systems Division (ISD) in the late 1980's as part of a Defense Advanced Research Project Agency (DARPA) contract to stabilize crane loads. The functional design was further developed and adapted for specialized applications including manufacturing, construction, hazardous waste remediation, aircraft paint stripping, and shipbuilding [5, 6, 7].

The basic RoboCrane is a six degree of freedom (DOF), parallel, kinematic machine (an inverted Stewart-Gough platform) actuated through a cable-based support system. The suspended moveable platform is kinematically constrained by maintaining tension in all six support cables (due to gravity) which terminate in pairs at the vertices of the overhead support.

This arrangement provides enhanced load stability over traditional lift systems (or cranes) and improved control of the position and orientation of the load. The suspended moveable platform and the overhead support typically form two opposing equilateral triangles, and are often referred to as the "lower triangle" and "upper triangle," respectively. In the version of RoboCrane used in this project, the Tetrahedral Robotic Apparatus (TETRA), all the winches, amplifiers, and motor controllers are located on the moveable platform.

2.2 Developments to Date and Current Limitations

The current RoboCrane is capable of autonomous assembly of a multi-component, simple, steel structure using pose tracking provided by a laser-based site measurement system (SMS) and assembly scripts generated from a commercial 4D-CAD package. An automated gripper mechanism was also designed and implemented. All of the above was achieved without altering RoboCrane's original teleoperated control system. Instead, a high-level autonomous control system was developed to simulate the human operator by sending identical 6-DOF joystick[2] commands through and reading the same encoder data from the original controller's interface.

Although autonomous operation was demonstrated using the current RoboCrane controller, the current RoboCrane controller does not have the necessary functionality and flexibility for use as a general development/testing platform. More sophisticated capabilities, such as real-time collision detection and the ability to combine various position sensing systems for pose estimation are not possible without altering the existing low-level control system. Thus, we decided to upgrade some of RoboCrane's control hardware and develop new control software that could more easily accommodate the various improvements required to operate efficiently as a versatile intelligent construction test platform for the ACT. The new control system is being developed following the NIST Real-time Control System (RCS) methodology.

[2] A 3D input device that allows a user to simultaneously command 6 degrees of freedom with one hand.

3. THE NIST REAL-TIME CONTROL SYSTEM

The NIST RCS methodology describes how to build control systems using a hierarchy of cyclically executing control modules. At the lowest level of the hierarchy, each control module processes input from sensors, builds a world model, and generates outputs to actuators in response to commands from its supervisory control module. These functional components of a control module are termed sensory processing (SP), world modeling (WM) and behavior generation (BG), respectively. The servo control of a motor is a common example of a control module at the lowest level. Here, the sensor may be a motor shaft position encoder, the actuator is the motor shaft, the command is a desired setpoint for the shaft position, and the behavior may be the execution of a simple proportional-integral-derivative (PID) control algorithm. The SP function may simply be reading and scaling input from the encoder device, and the WM function may be maintaining a filtered estimate of the shaft position. Typical cycle times for such control modules are on the order of a millisecond.

One or more of these lowest-level control modules may be subordinate to a control module at the next level up in the hierarchy, termed the supervisor. In our example, the SP function at this level may simply provide each motor shaft position to the WM function, which would compute the overall position and orientation of the device's controlled point, perhaps the tool on a robot. The BG function may smoothly transform goal points to motor trajectories based on speed, acceleration and jerk. Here, goal points may arrive at variable intervals from the higher-level supervisor, one that may be reading them from a program file. Cycle times increase by about an order of magnitude for control modules one level higher in the hierarchy. For this trajectory planner, the cycle time would be about 10 ms.

A full RCS hierarchy would include additional lowest-level control modules for individual tools, and control modules at higher levels of the hierarchy may coordinate the actions of many robots and auxiliary equipment. RCS has found its richest application in the area of mobile robotics. Here the SP functions include not just motors but cameras, 3D imaging systems (e.g. laser scanner), GPS and other navigation sensors. WM functions build maps of various resolutions and maintain symbolic representations of the world. BG functions reason on the symbolic representations, planning optimal paths around known features and reacting to sensed obstacles.

An RCS design differs from functional design or object-oriented design in that it begins with a task analysis of the system to be controlled. Here the designer identifies the tasks to be performed at the top level, and then breaks each task down into subtasks that are performed by the subordinates. Usually the designer does not have complete freedom to determine the task breakdown, as some of the components that make up the system may have been reused from prior projects. In this case, the tasks must be expressed in terms of the available subtasks. Task analyses are helped enormously by considering scenarios that include system startup, shutdown, normal use and changes between various modes of operation. Often these scenarios bring to light the need for tasks that are not apparent from the original conception of the system.

An example of a comprehensive task analysis for the design of an automatic road vehicle controller can be found in [8]. The designers considered hundreds of scenarios listed in a manual of military driving, including lane changes, passing and intersection rules. What is made obvious by this analysis is that the top- and bottom-level tasks are relatively simple, while the tasks in the middle are the most complex. Other examples of task analyses for unmanned vehicle systems can be found in the latest version of RCS (known as 4D/RCS) [9].

Implementation of RCS control modules is done conceptually using state tables, which can then be programmed in any general-purpose computer language using conditionals or switch statements. The NIST RCS Library [10] documents the software tools available for programming in C++ or Java. A detailed handbook [11] covers the entire RCS analysis, design and programming using several examples and the RCS Library tools.

4. RCS CONTROLLER DESIGN FOR AUTONOMOUS STRUCTURAL STEEL ERECTION

Designing a new RCS-based controller for RoboCrane began by first identifying the requirements of the controller. Although the initial focus was on structural steel erection, we considered it important to design the new controller to be expandable to handle more generic construction tasks. Therefore, the overall goal of the RoboCrane controller was defined as follows: *to plan and execute tasks required for automated construction-material handling and/or building construction.*

4.1 Controller Requirements

The requirements for the new control system were defined next. In order to accomplish everything we intended to do with RoboCrane we decided that the controller should provide the following:

- Autonomous, semi-automated, and teleoperated modes of operation
- RoboCrane tool-point (i.e., platform) position and velocity control modes
- RoboCrane tool-point motion in joint, Cartesian, as well as other user-definable coordinate systems
- Cross-platform code portability (but still dependent on the real-time operating system)
- Adaptability to other robot/crane hardware
- Sensor-based collision avoidance

4.2 System Scope

Although the motivation for developing a new controller was to be able to use it to control various cable-driven robots and to accomplish various tasks, the initial scope of the controller was limited to the following:

- Smooth and stable motion of the NIST RoboCrane
- Perform the steel beam pick and place task
- Construct a structure whose shape is limited by RoboCrane's current range of motion
- Connect the beam to the holder using drop-in connectors
- Carry beams whose size falls within RoboCrane's current load-carrying capabilities
- Communicate to RoboCrane using the current field bus architecture
- Operate under a real-time Linux operating system
- Use the built-in incremental winch motor encoders as well as the laser-based positioning system to determine RoboCrane's pose, but include the ability to add other sensors for pose determination in the future
- Acquire the steel beam and holder poses using the current laser-based positioning system

4.3 Task Decomposition

The next step in the RCS controller design process was to conduct a task decomposition of the controller's overall goal. Through a series of internal brainstorming meetings, we divided the overall goal into several subtasks, which were consequently also broken down into smaller tasks. This process continued until the lowest level tasks involved sending commands to the RoboCrane hardware (e.g., setting motor voltages). This is the lowest level of control that the new controller can provide.

Figure 2 shows a sample task tree diagram resulting from the task decomposition process. In this figure the physical task of picking and placing a steel beam (as part of a steel erection sequence) is decomposed into 3 levels of subtasks. In keeping with the RCS architecture, each sublevel is responsible for planning and executing a smaller portion of the overall pick-and-place task. The lowest level is responsible for maintaining a commanded joint (or motor) velocity (or position). The next level up is responsible for generating and executing a series of *n* waypoints (i.e., positions and orientations in time) for the RoboCrane platform. The next higher level generates and executes the necessary commands to accomplish a segment of the pick-and-place operation. Finally, the highest level in Figure 2 is responsible for coordinating the execution of the segments that make up the overall pick-and-place task. This highest level also receives commands from higher levels (not shown in Figure 2) which coordinate the pick-and-place task with other tasks such as attaching a beam to a structure, picking and placing a column, and etc.

In addition to the physical tasks represented in the task tree diagram of Figure 2, other non-physical tasks are required in order to accomplish a pick-and-place operation. These include tasks such as detecting obstacles, calculating collision free paths, etc. These tasks were also captured and broken down into 3 levels of subtasks, but are not included in Figure 2.

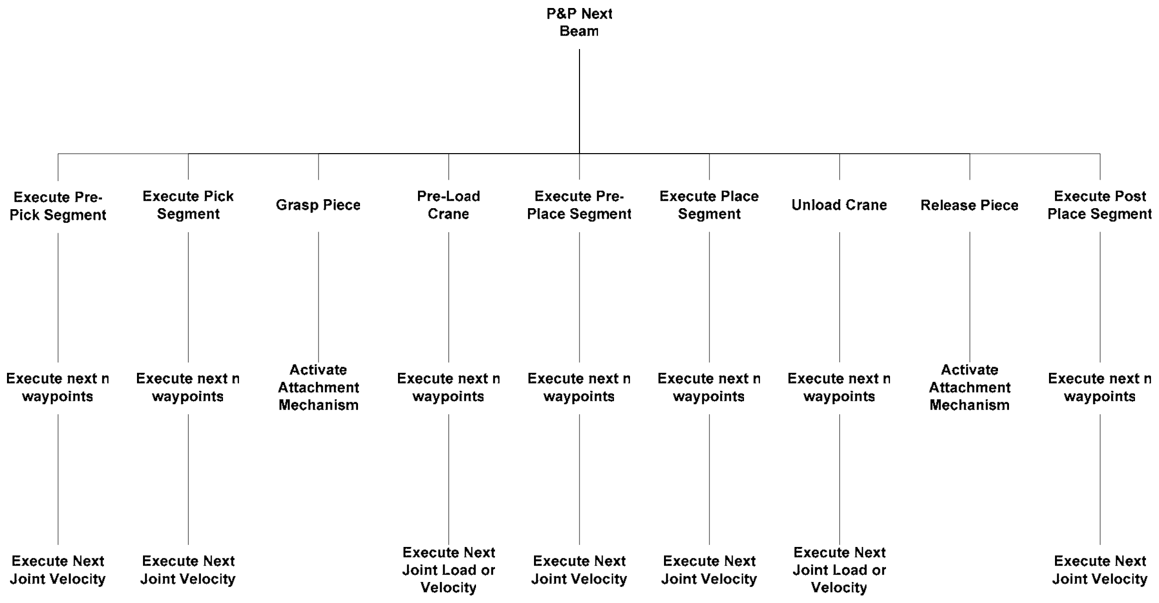

Figure 2. Task tree diagram for the *pick-and-place next beam* task.

4.4 State Tables

Following the task decomposition process, the commands going into and out of each task represented in the task tree diagram of Figure 2 are listed in a state table format. A state table (or state transition table) describes all possible input and output states (and actions) of a finite state machine. Figure 3 shows a state table for the *pick and place next beam* task. The command that starts the execution of this task has the same name as the task itself and is also the title of the state table. The state table columns (from left to right) represent the input state numbers, the conditions that must be met to change the state, the output state numbers, and the output commands that are sent to lower level tasks, respectively.

Pick and Place Next Beam			
S0	New Command	S1	Hold – Status=Executing
S1	Conditions Good to Move to Pre-Pick Pose	S2	Move to Pre-Pick Pose
S1	Timed out	S0	Hold – Status=Error
S2	Conditions Good to Move to Pick Pose	S3	Move to Pick Pose
S3	Conditions Good to Grasp	S4	Grasp Beam
S4	Conditions Good to Pre-Load Crane	S5	Pre-Load Crane
S5	Conditions Good to Move to Pre-Place Pose	S6	Move to Pre-Place Pose
S6	Conditions Good to Move to Place Pose	S7	Move to Place Pose
S7	Conditions Good to Unload Crane	S8	Unload Crane
S8	Conditions Good to Release	S9	Release Beam
S9	Conditions Good to Move to Post Place Pose	S10	Move to Post Place Pose
S10	At Post Place Pose	S0	Hold - Status=Done

Figure 3. State table for the pick and place next beam task.

When the *pick and place next beam* command is issued by a higher level task, the controller examines the state table shown in Figure 3. The initial state of the *pick and place next beam* task is S0 and the first condition that is checked is whether the received command is new. If it is a new command, the state of the task is changed to S1 and the status of the task is changed to indicate that it is executing.

The next time the above state table is checked (during the next execution cycle of its corresponding control module) the new state of the task is S1, and the conditions that must be met are whether it is acceptable to move RoboCrane to the beam's pre-pick pose, or whether enough time has elapsed that something must be wrong. There may be one or more sub-conditions that must be satisfied in order to determine whether it is acceptable to proceed, but these can be aggregated into one description in the state table. If the conditions are met, the state of the task is changed to S2 and the command to move to the pre-pick pose is sent to a lower-level task. If time has expired, the state of the task is changed to S0 and an error is reported. Each lower level task that receives an output command reports its status back to the higher level task that issued the command until it finishes executing or encounters an error. This process continues until all of the commands in the state table have been executed, at which point the *pick and place next beam* task is considered completed and the state of the table is reset to S0. For brevity, only a single timeout condition is shown in Figure 3. In practice, numerous checks of this sort are made throughout the state table. Once the state tables for all of the tasks identified through the task decomposition process are completed they are organized into control modules as described next and implemented in software following the RCS guidelines.

4.5 Control Modules

As described in the RCS description in Section 3, the tasks in the task tree diagram of Figure 2 are organized into multiple levels. Each level's tasks may be grouped together into one or more modules responsible for coordinating and executing the tasks within it. Some of the critical modules (such as the servo algorithms) run as real-time processes within the operating system, while other less critical modules (such as long term path planning) run as non-deterministic processes.

Figure 4 shows the control architecture for the new RoboCrane controller. The four levels above the software/hardware demarcation line in Figure 4 correspond to the four levels of Figure 2. The tasks have been grouped into the control modules shown. For example, the bottom level tasks of Figure 2 are grouped into the six "Servo" modules in Figure 4. Each of these modules are responsible for executing a servo algorithm which accepts the actual and desired positions (or velocity) of a winch motor as inputs and calculates a command voltage which maintains the desired position (or velocity). An alternate configuration would be to group the six servo modules into one.

Figure 4 also shows that the RoboCrane controller is part of a larger control architecture which includes four higher-level modules. For example, at the level above the RoboCrane controller would be a Pick-and-Place Manager that would actually command RoboCrane to perform the pick-and-place operation. The commands sent down by each module to a lower-level module are shown in the light gray boxes on the right. Some of the functions (or non-physical tasks) that each module performs are also shown in the light gray boxes on the left. The control modules above the Pick-and-Place Manager are included in the figure, but are out of the scope of the current controller development effort.

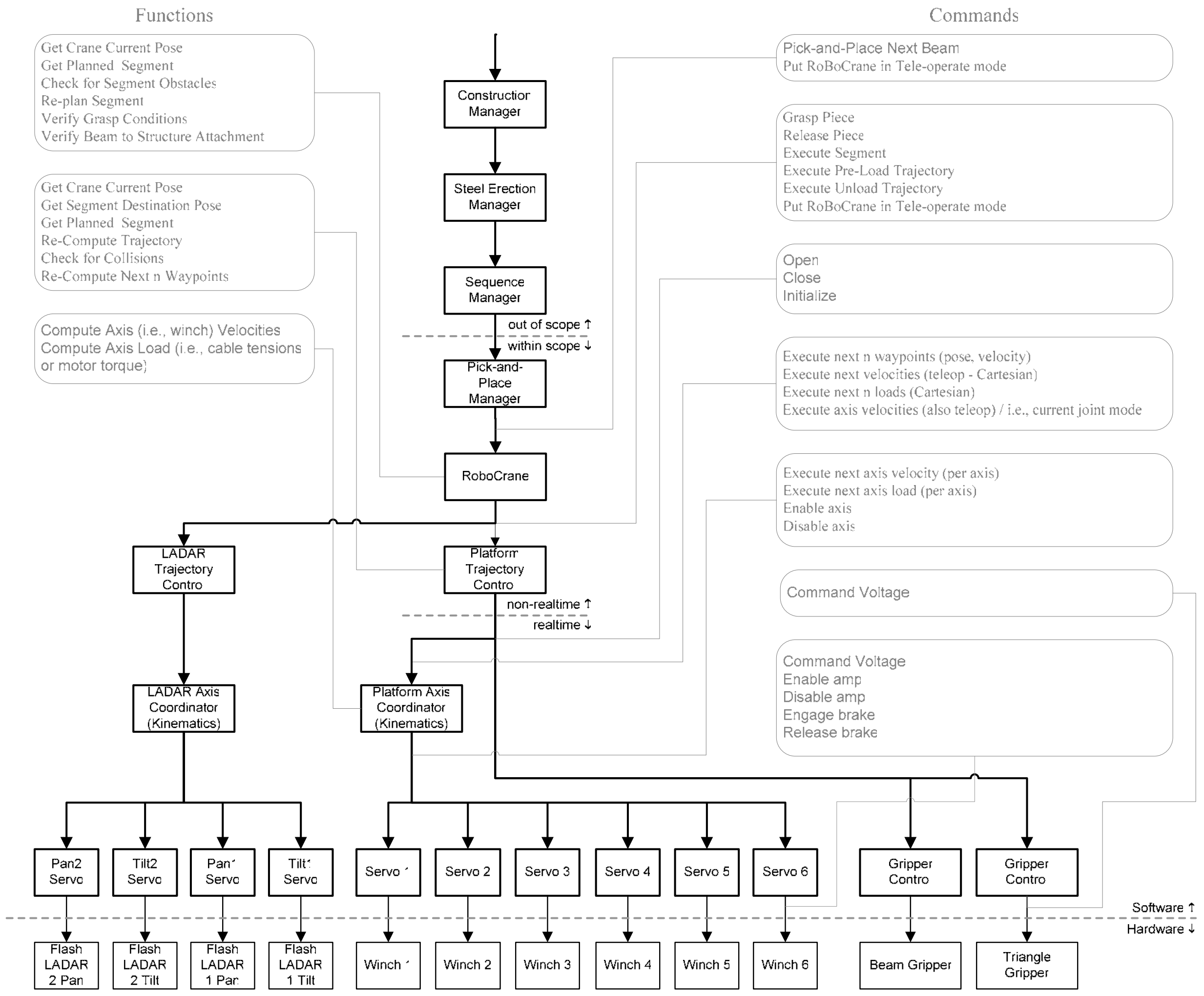

Figure 4. New RoboCrane controller architecture diagram.

Finally, Figure 4 also includes modules for controlling the 3D imaging systems. These modules are responsible for coordinating the sensor orientations with the RoboCrane platform's motion in order to maintain a desired part of RoboCrane's environment within the combined sensors' field of view.

5. RESULTS AND CONCLUSIONS

Researchers from the Construction Metrology and Automation Group at NIST are developing a new controller for the RoboCrane robot, which is based on the NIST Real-time Control System methodology. The new controller will enhance RoboCrane's performance and enable more complex functionality for the NIST Automated Construction Testbed. These efforts are initially focused on developing standards, test methods, and performance metrics for robotic structural steel erection.

To date, the lowest level tasks identified through the RCS task decomposition process (Figure 2) have been implemented and tested as part of the new RoboCrane servo control module. In addition, the software interface between the servo module and the RoboCrane fieldbus communication hardware has also been completed and tested as part of the servo control module development effort. The servo algorithm has been tuned and tested with RoboCrane's winch motors using position feedback from the motor encoders and a PID loop to calculate the required motor voltages. The servo control module runs at a 5 ms frequency within a real-time task. This control cycle time was determined through testing and is capable of controlling the winches at their maximum rate of rotation.

6. FUTURE WORK

Future work will focus on implementing the higher level tasks from the RCS task decomposition process as a near-term goal. This will also involve developing new world modeling and sensor processing capabilities for RoboCrane.

7. REFERENCES

[1] Dagalakis, N.G., Albus, J.S., Goodwin, K.R., Lee, J.D., Tsai, T., Abrishamian, H., and Bostelman, R.V., "Robot Crane Technology Program – Final Report," *NIST Technical Note 1267*, July 1989.

[2] Albus, J., Bostelman, R., and Dagalakis, N., "The NIST RoboCrane, A Robot Crane," *Journal of Research of the National Institute of Standards and Technology*, Vol. 97, No. 3, May-June 1992, pp. 373-385.

[3] Bostelman, R.V., Jacoff, A., Dagalakis, N.G., Albus, J.S., "RCS-Based RoboCrane Integration," *Proceedings of the International Conference on Intelligent Systems: A Semiotic Perspective*, Gaithersburg, MD, October 20-23, 1996.

[4] Saidi, K.S., Lytle, A.M., and Scott, N.A., "Developments towards Automated Construction," *NISTIR 7264*, National Institute of Standards and Technology, Gaithersburg, MD, 2005.

[5] Bostelman, R.V., Albus, J.S., Dagalakis, N.G., Jacoff, A., and Gross, J., "Applications of the NIST RoboCrane," *Proceedings of the 5th International Symposium on Robotics and Manufacturing*, Maui, HI, August 14-18, 1994, (1994)

[6] Bostelman, R., Albus, J., and Stone, W. C., "Toward Next-Generation Construction Machines," *American Nuclear Society Proceedings*, Seattle, WA, March 4-8, 2001.

[7] Bostelman, R.V., Proctor, F.M., Shackleford, W., Lytle, A., Albus, J.S., "The Flying Carpet: A Tool to Improve Ship Repair Efficiency," *American Society of Naval Engineers Symposium, Manufacturing Technology for Ship Construction and Repair*, Bremerton, WA, September 10-12, 2002.

[8] Barbera, A., Albus, J., Messina, E., Schlenoff, C., and Horst, J., "How Task Analysis Can Be Used to Derive and Organize the Knowledge for the Control of Autonomous Vehicles," *Proceedings of the AAAI Sprint Symposium Series on Knowledge Representation and Ontology for Autonomous Systems*, Palo Alto, CA, March 22-24, 2004.

[9] Albus J. and et al., "4D/RCS: A Reference Model Architecture for Unmanned Systems, Version 2.0," in *NISTIR 6910*, National Institute of Standards and Technology, Gaithersburg, MD, 2002.

[10] RCS, "Real-Time Control Systems Library – Software and Documentation," available: http://www.isd.mel.nist.gov/projects/rcslib.

[11] Gazi, V., Moore, M.L., Passino, K.M., Shackleford, W.P., Proctor, F.M. and Albus, J.S., *The RCS Handbook – Tools for Real-Time Control Systems Software Development*, John Wiley & Sons, 2001. ISBN 0-471-43565-1.