Proceedings of the Performance Metrics for Intelligent Systems (PerMIS) Workshop, Gaithersburg, MD, August 13-15, 2002.

A Simulation Framework for Evaluating Mobile Robots

Stephen Balakirsky and Elena Messina
National Institute of Standards and Technology
Intelligent Systems Division
Gaithersburg, MD 20899-8230
Email: stephen@cme.nist.gov, elena.messina@nist.gov [*][†]

Abstract

As robotic technologies mature, we are moving from simple systems that roam our laboratories to heterogeneous groups of systems that operate in complex non-structured environments. The novel and extremely complex nature of these autonomous systems generates a great deal of subsystem interdependencies that makes team, individual system, and subsystem validation and performance measurement difficult. Simple simulations or laboratory experimentation are no longer sufficient. To assist in evaluating these components and making design decisions, we are developing an integrated real-virtual environment. It is our hope that this will greatly facilitate the design, development, and understanding of how to configure and use multi-robot teams and will accelerate the robots' deployment.

Keywords:

simulation, architectures, 4D/RCS, mobile robots, algorithm validation

1 Introduction

There have been many recent successes in the field of mobile robotics. These range from single robot systems such as MINERVA that has been designed to give guided tours of museums [8], Predator and Global Hawk that have been designed for military air applications, and Demo III [7] [5] and Perceptor that have been designed for military ground applications to multi-robot systems such as the multiplicity of robot teams involved in the Robocup soccer league [3].

As these systems become more complex and attempt to perform more ambitious tasks, the knowledge and resources (both hardware and software) that are necessary to make contributions to the field dramatically increases. As a result, informal (or formal) code sharing now takes place between many universities and research institutions. For example, the code used in the Robocup competitions is published and freely available to anyone who wishes to download and use it, and the mobility and planning software used by the Perceptor program is heavily based upon the code from Demo III. While this code reuse allows researchers to gain quick entry into the various mobile robot arenas, it raises some interesting questions. If multiple sources of algorithms that provide a solution to a particular problem exist, which one is better? Will the given algorithm work in the new proposed environment? Are there any unintended consequences on the rest of the system (or systems) of integrating in this new component?

Code and component sharing also allows researchers to perform research in a specific area of robotics without becoming an expert in every aspect of robotics. For example, it should be possible on today's computer hardware to develop a planning system capable of creating plans to navigate through complex city traffic. However, in order to do this, an image processing system must exist that can detect and predict the location of traffic, road lanes, and traffic signs. It is feasible that a planning researcher can reuse a base technology such as the image processing subsystem rather then creating one from scratch. However, is there a solution to this problem when no such base technology or algorithm is available? Will the new

[*]No approval or endorsement of any commercial product by the National Institute of Standards and Technology is intended or implied. Certain commercial equipment, instruments, or materials are identified in this report in order to facilitate understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

[†]This work was sponsored in large part by grants from the Army Research Laboratories and the DARPA Mobile Autonomous Robot Software Program.

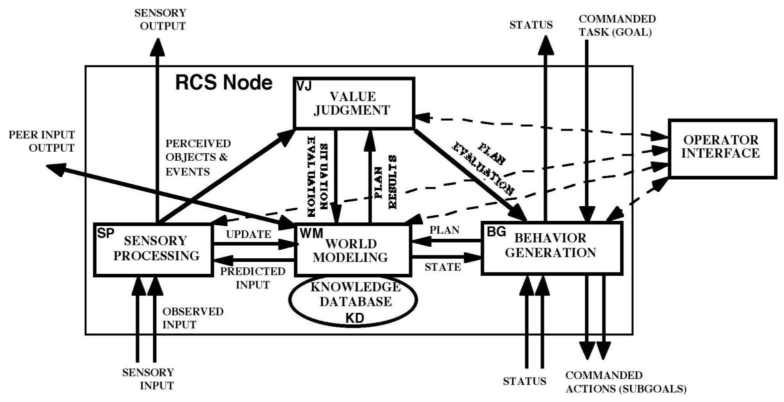


Figure 1: Model for RCS Control Node.

planning approach have to wait to be proven because of dependencies on as yet undeveloped algorithms or hardware?

This paper will suggest a means to address all of the above questions by describing the development of an integrated real-virtual simulation environment. The objective of this environment is to provide a standard architecture and set of interfaces through which real and virtual systems may be seamlessly coupled together. It will be shown that through this coupling, components ranging from individual algorithms to groups of vehicles may be developed, debugged, and evaluated.

2 An Architecture for Intelligent Autonomous Vehicles

One of the key decisions to be made in building any kind of complex system is how to organize the hardware and software. The Demo III Program and some of the teams competing for the Future Combat Systems contract have selected the 4D/RCS reference architecture for their autonomous vehicles [1]. Rather than starting from scratch, the simulation framework will be built upon the existing 4D/RCS architecture and will take advantage of existing interfaces and components.

The 4D/RCS architecture consists of a hierarchy of computational nodes, each of which contains the same elements. Based on control theory principles, 4D/RCS partitions the problem of control into four basic elements that together comprise a computational node: behavior generation (BG), sensory processing (SP), world modeling (WM), and value judgment (VJ). Figure 1 shows the 4D/RCS control node and the connections between its constituent components. Figure 2 shows a sample 4D/RCS hierarchy for military scout vehicles.

3 Requirements for a Simulation, Modeling, and Development Framework

An architecture is a first step towards guiding and facilitating the construction and evaluation of complex single or multi-vehicle autonomous systems. Tools that help automate the software development and component integration are another important element. NIST has been working with industry, other government agencies, and academia to investigate tools to facilitate construction of the types of large and complex systems that will be represented in this simulation framework. We are developing a large-scale simulation environment that will enable us, along with others, to design the control hierarchy, populate the control nodes, run the system in simulation, debug it, and generate code for the target host. The development and simulation environments are closely tied to the eventual deployment platforms and are intended to be able to operate with a combination of real and simulated entities. The ability to enable human-in-the-loop testing and execution is also crucial, given the novel aspects of human-robot interactions.

A high-level list of the requirements for such a development and simulation environment has been developed to help guide its creation. The requirements are as follows:

- Full support of 4D/RCS architecture
- Graphical user interface for developing, integrating, testing, debugging the system under development

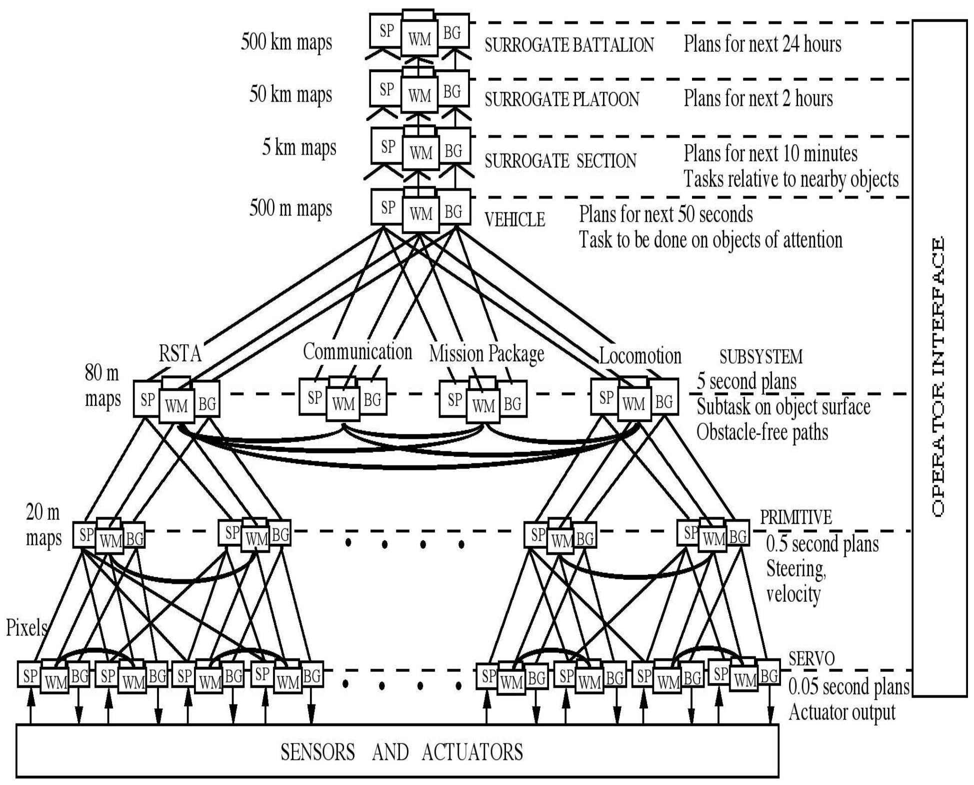


Figure 2: 4D/RCS Reference Model Architecture for an individual vehicle.

- Reuse support:
  - Architecture elements
  - Component templates
  - Algorithms
  - Code
  - Subsystems
- Intuitive visualizations of the control system to support design and development by providing an understanding of what the system is doing and why, and what it plans to do next. Examples of visualizations include:
  - Display of control hierarchy as it executes, including commands flowing down and status flowing up
  - Ability to "zoom in" on a particular node and view states as the system executes
  - Ability to view world models within the system
- Execution controls, including
  - Single step through execution
  - Breakpoints and watch windows
  - Logging
- Simulation infrastructure supporting realistic execution scenarios, visualization, and debugging/evaluation experiments. This includes
  - Population of the environment external to the system with relevant features (such as roads, other pieces of equipment, humans, etc.)
  - Controlled dynamic environment (with prescribed repeatable events)
- Modification capabilities so that the designer and user can perform "what if" experiments. The tools should allow interactive and intuitive modification of situations in the environment or within the system. The modification capabilities should work seamlessly with the visualization, simulation, and execution features. Examples of types of modifications that should be allowed include:
  - Changing world model data
  - Importing datasets that represent what the system's sensors would receive
  - Changing environmental conditions
- Support for real-time computing. All levels of the 4D/RCS control hierarchy must execute within

certain time constraints (e.g., the servo level may have to respond at 60 Hz; whereas higher levels may have several seconds or even minutes per cycle).

4 Proposed System of Systems

We are seeking to create an *integrated* environment that provides capabilities typically associated with software development tools and those associated with simulation environments. All of the pieces necessary for the construction of this environment may exist to some degree as separate commercially available packages. However, whereas several commercial tools exist to help design and construct software, these tool-sets are typically disconnected from the overall system architecture and execution environments (real or simulated). Likewise, many sophisticated simulation systems exist. However, these systems tend to work at either a very broad scope at low resolution, or a very limited scope at high resolution. What we are building is a coherent environment for designing, developing, and validating software components ranging in size from a single module to a team of systems.

As shown in Figure 3 one will be able to take software modules from a repository and interface them directly into either a real or virtual system. This ability will be supported through the decomposition of systems and algorithms into the 4D/RCS architecture and the use of standard interfaces between modules.

This decomposition will be supported through the software design and development support tools that provide the ability to work in a graphical environment to sketch the control hierarchy, bring up partially instantiated 4D/RCS control nodes, easily create connections between nodes (or components within them), and automatically generate executable code. The software support tools will also encompass capabilities typically found under run-time debug tools, including single stepping and setting break points. Furthermore, sophisticated displays of variables and execution states are being created. For instance, a "strip chart" to view one or more variables can be displayed on screen and techniques for helping developers visualize complex world models are being designed. This includes display of the graphs that several path planning algorithms utilize to search for the best (least cost) path. The graphs typically have thousands of nodes, which are connected in a neighborhood to other nodes by edges that have a cost associated with them. The costs vary, depending on operation mode, or as environmental conditions change and are computed based on the various layers in the world model (such as roads, obstacles, and elevation). Therefore, the visualization of relevant and salient aspects of the world model in order to validate the model itself and the planning is a challenging undertaking.

The software development tools will segue smoothly into the simulation environment. Under this concept, a virtual world is being created that brings together existing multi-platform and single platform simulation systems into a system of systems. Through the use of well-defined interfaces that are supported on a wide variety of computer platforms, the simulator's internal command and data flows will be able to be interrupted and modified. This will allow researchers to "plug-in" individual technology components that meet the interface requirements and override the default methods that the simulators normally employ. As shown in Figure 3, interfaces will be provided that range through the entire spectrum of the 4D/RCS hierarchy; from a low-fidelity multi-platform configuration to a high-fidelity single platform configuration, to the ability to add real platforms into the virtual world.

Global variable resolution database resources will also be available. These include a terrain database that contains elevation data, a feature database that contains annotated vector data for roads, signs, buildings, rivers, etc., and an annotated entity database that contains information on all of the platforms participating in the simulation. The annotations include items such as lane markings and names for roads, text contained on a sign, and health status of other entities. Filters will be available to tune the database outputs to the specific needs of each algorithm. For example, specific sensor processing capabilities may be simulated by querying these databases with a specified sensor range and resolution.

In addition to serving *a priori* data, these databases will be able to be modified in real-time. Any modifications made to the databases will be viewable by all participants (both real and virtual) in the exercise. These modifications may be related to sensed information from a real vehicle that is participating in the simulation or may be injected by the user to alter the environment that the simulation framework is operating in.

The final component of the system is a set of data capture, analysis, and evaluation tools. The data capture tool will allow for any or all of the messages being transmitted over the standard message channels to be time-tagged and logged into a trace file. In addition, raw simulation results may be logged, for example the location and activities of each participating entity.

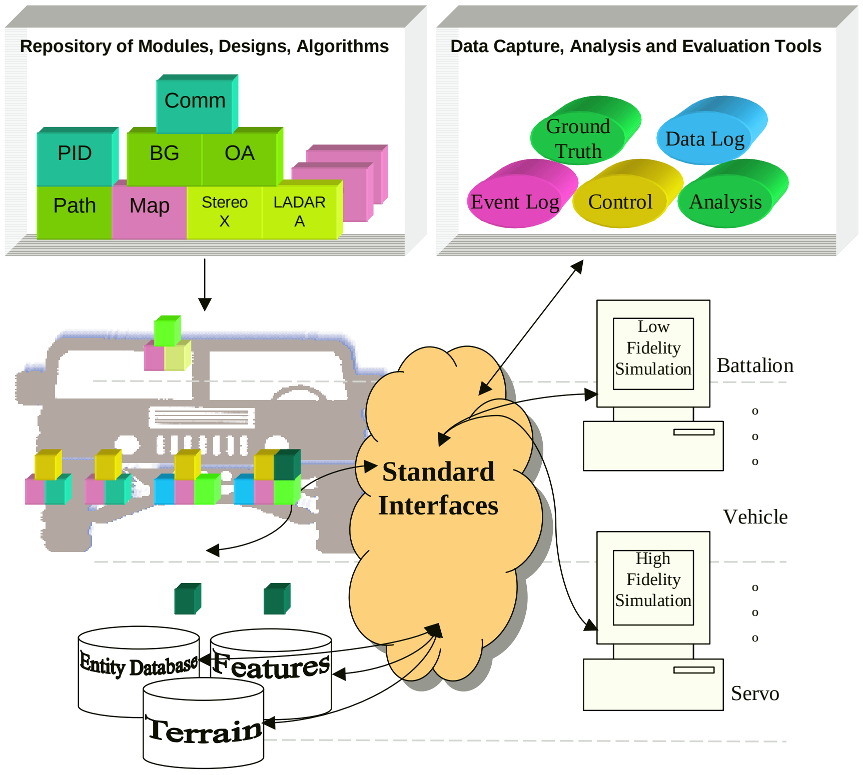


Figure 3: Hierarchy of simulators.

Data analysis tools are also being developed that will allow for a variety of data presentation and analysis options. This will include both the tracking of timings as well as the values of specific variables or combinations of variables. For example, the distance between two entities may be constantly tracked. These data analysis tools will also be tied into ground truth from the simulation systems. This will allow algorithm and system evaluation against known data and the determination of such items as the time from a sensor event to a system reaction or the accuracy of a real systems road following algorithms. Through the use of these standard interfaces and tools, researchers will be provided with a low cost technique for evaluating performance changes due to their system's or algorithm's integration into the overall framework.

5 Current Implementation

While the entire software development and simulation system has not yet been implemented, progress has been made on developing prototypes and designs for the overall system. No single simulation tool has been found that meets all of the criteria discussed in the requirements section. Therefore, a hierarchy of simulators has been explored.

At the top of the hierarchy, a low-fidelity, long temporal and spatial duration, multi-platform simulator is required. As designed, this class of simulator is capable of simulating the interaction, coordination, and movement of large groups of platforms. While these simulators do simulate down to the level of the individual platforms moving across the terrain, the terrain and mobility models are typically low resolution. Therefore, this class of simulator is best utilized in developing algorithms for group behaviors where precise platform mobility and sensing modeling is beyond the scope of the experiment. A second class of simulator has been identified for situations where precise modeling is required. These simulators will need to share interfaces with the low-fidelity simulator, and in fact may take commands from the low-fidelity simulators in order to precisely model one or more of the platforms involved in a particular exercise. The high-fidelity simulators will also be able to read the shared databases and construct simulated sensor output (or pass real sensor output) that may be used by external sensor processing algorithms. Complex, dynamically correct platform motion models and high resolution terrain information will also available at this level.

Interfaces will be inserted into each simulator that will enable the export and import of both world model and behavior generation information at each level of the 4D/RCS hierarchy. This will enable researchers to implement a particular group behavior or algorithm at a particular level of 4D/RCS. For example, a cooperative search algorithm could be implemented at the "section" level of 4D/RCS. The algorithm would receive its command input from the platoon level of the low-fidelity simulator and construct a plan based on information read from the terrain, entity, and feature databases. The planned course of action (individual plans for several platforms) would then be passed back into the simulator for execution. In this particular case, the plans could be passed either back to the low-fidelity simulator or to the high-fidelity simulator. In addition, one or more of the platform's plans could be passed to real systems. As designed, the source and destination of these plans and the data utilized to construct them is transparent to the planning system and totally controlled by the user. This will facilitate an environment where a researcher can simulate as many or as few subsystems and levels as desired.

Both a low-fidelity and high-fidelity commercially available simulation package have been selected and implemented into the prototype framework.

5.1 Low-fidelity Simulator

For the prototype system, we have chosen the US Army STRICOM's OneSAF Testbed Baseline (OTB)[1] for both the low-fidelity simulation and shared database server. We have worked closely with the Army Research Laboratory and Science and Engineering Services Inc. to install the standard interfaces. All of the interfaces communicate over NIST's NML communication channels [6] which provide a multi-platform solution to inter-process communication.

Distributed, shared databases are implemented as part of the standard implementation of OTB. We have added interfaces into the simulation system that allow for simple outside access of this information. These interfaces include hooks into the terrain elevation database, feature database, and entity database. Additional channels that tie the basic information contained in these databases to a full relational database for the storage of attribute information is currently being investigated.

For the terrain elevation database, both all-knowing (what is the elevation in this area, to this resolution) and modeled (what is the terrain map as modeled by vehicle x with its sensors) are available. Feature vector data is available on an all-knowing basis that may be filtered by distance from the vehicle so as to simulate what sensors perceive. In addition to the standard features that are modeled by the simulation system, simulated traffic signals and signs are being implemented. For entity data, filtered information (all friendly, enemy, detected, etc.) reports are available as well as event detections. Events currently supported include line crossings and anticipated line crossings with more to be added shortly.

In addition to the database access interfaces, we are able to interrupt the standard OTB command flow to inject our own plans. This has been demonstrated by having OTB section level plans sent out over an NML channel to a stand-alone vehicle level planner. The results of the vehicle level planner can then be executed on real robotic hardware, sent to a high-fidelity simulator, or sent back into the OTB simulator for execution.

Input from real robot platforms into the simulation environment is also supported. This interface allows a real robotic platform to influence OTB databases by continuously updating their own location as well as adding detected features and entities.

Work is continuing on developing further interfaces. These will provide further breaks in the OTB command flows that will allow for planning systems that compute group plans to be implemented and evaluated.

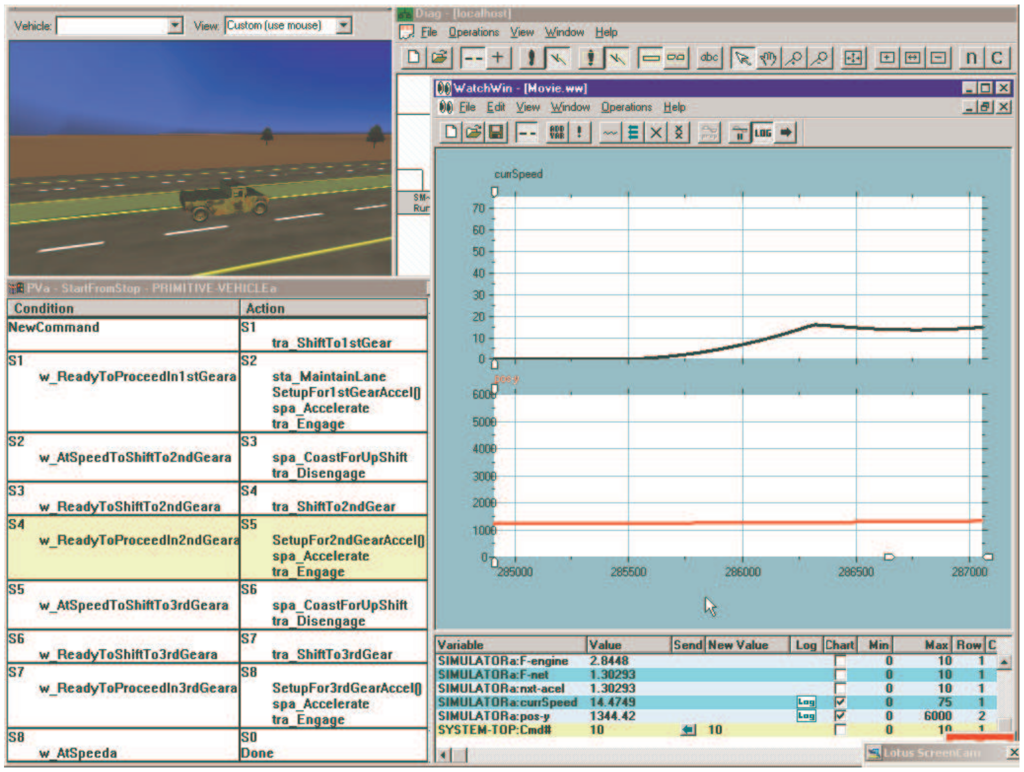
Another feature that is standard with the OTB distribution is a data logger. This logging facility logs all entity movements and events that occur in the simulation. Logging facilities to log message channel traffic are currently under development.

5.2 High-fidelity Simulator

For the high-fidelity simulation, SimRobot from the University of Bremen[2] has been selected for the prototype system. NML channels for low-level command input, and position output have been implemented. Currently, this simulator is only capable of simulations on a flat earth. Therefore, work is being performed to improve the simulators ability to operate in complex 3-dimensional terrain. Once this work is completed, the high-fidelity simulator will operate from the same terrain database as the low-fidelity simulator.

[1]http://www.onesaf.org/publicotb1.html

[2]http://www.informatik.uni-bremen.de/~roefer/simrobot/index_e.htm


Vehicle:
View: Custom (use mouse)
Diag - [localhost]
File Operations View Window Help
WatchWin - [Movie.ww]
File Edit View Window Operations Help
currSpeed
PVa - StartFromStop - PRIMITIVE-VEHICLEa
Lotus ScreenCam

Condition	Action
NewCommand	S1 tra_ShiftTo1stGear
S1 w_ReadyToProceedIn1stGeara	S2 sta_MaintainLane SetupFor1stGearAccel() spa_Accelerate tra_Engage
S2 w_AtSpeedToShiftTo2ndGeara	S3 spa_CoastForUpShift tra_Disengage
S3 w_ReadyToShiftTo2ndGeara	S4 tra_ShiftTo2ndGear
S4 w_ReadyToProceedIn2ndGeara	S5 SetupFor2ndGearAccel() spa_Accelerate tra_Engage
S5 w_AtSpeedToShiftTo3rdGeara	S6 spa_CoastForUpShift tra_Disengage
S6 w_ReadyToShiftTo3rdGeara	S7 tra_ShiftTo3rdGear
S7 w_ReadyToProceedIn3rdGeara	S8 SetupFor3rdGearAccel() spa_Accelerate tra_Engage
S8 w_AtSpeeda	S0 Done

Variable	Value	Send	New Value	Log	Chart	Min	Max	Row
SIMULATORa:F-engine	2.8448				☐	0	10	1
SIMULATORa:F-net	1.30293				☐	0	10	1
SIMULATORa:nxt-acel	1.30293				☐	0	10	1
SIMULATORa:currSpeed	14.4749			Log	✓	0	75	1
SIMULATORa:pos-y	1344.42			Log	✓	0	6000	2
SYSTEM-TOP:Cmd#	10	⬅	10		☐	0	10	1

Figure 4: Example of features in controller software development tool.

5.3 *Software Development Support*

We have been experimenting with various representation techniques and development tools. These range from commercial packages, such as Real-Time Innovation Incorporated's ControlShell to novel formal languages, such as Stanford's Rapide [4]. Recent work has focused on the use of the Unified Modeling Language to support 4D/RCS control system development [2].

A commercial development and execution tool for building simpler versions of RCS-style controllers has been developed by a small company (Advanced Technology and Research), but it is targeted at manufacturing systems that have minimal sensing requirements. This tool is being modified to support the types of visualizations, modifications, and execution controls desired for on-road and off-road vehicles. Figure 4 is a screen shot taken of the tool while running and illustrates some of its features. The top left window shows an animation of the vehicle being controlled as it moves through its environment, which can include other vehicles. The top right window shows 2 variables that have been selected by the user to be graphed (current speed and y position). All other variables in the world model are accessible through the lower right window, where logging, charting, minimum, maximum, and other values are displayable. The lower left window displays the state table for one of the controller nodes (in this case, the Prim) for the vehicle, with the current state highlighted.

6 Summary

This paper has presented an integrated simulation framework that is capable of aiding researchers in developing, debugging, and evaluating algorithms, subsystems, systems, and groups of systems. While the entire framework has not yet been implemented, a prototype system does exist. This system allows the seamless operation of both real and simulated systems in an environment that contains both real and virtual features. In addition, standard interfaces, data logging facilities and ground truth exist to aid in the evaluation of the performance of systems and the comparison of multiple systems under repeatable conditions.

While interfaces exist for the "section" and "vehicle" level of the 4D/RCS architecture, it is desired to have interfaces that allow the testing and evaluation of components that reside at any architecual level. It is also desirable to be able to simulate additional environmental models for both the high and low fidelity simulators. These include traffic signals and signs for the low-fidelity simulator and a more realistic mobility platform simulator for the high-fidelity simulator.

Therefore, future work on this framework includes the development of additional interfaces, further development of the simulation environment, and the incorporation of the design and debug toolsets.

References

[1] J. Albus. 4-D/RCS reference model architecture for unmanned ground vehicles. In G Gerhart, R Gunderson, and C Shoemaker, editors, *Proceedings of the SPIE AeroSense Session on Unmanned Ground Vehicle Technology*, volume 3693, pages 11–20, Orlando, FL, April 1999.

[2] H. Huang, E. Messina, H. Scott, J. Albus, F. Proctor, and W. Shackleford. Open system architecture for real-time control using an uml-based approach. In *Proceedings of the 1st ICSE Workshop on Describing Software Architecture with UML*, 2001.

[3] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara. RoboCup: A challenge problem for AI. *AI Magazine*, 18(1):87–101, 1997.

[4] E. Messina, C. Dabrowski, H. Huang, and J. Horst. Representation of the rcs reference model architecture using an architectural description language. In *Lecture Notes in Computer Science EUROCAST 99*, volume 1798 of *Lecture Notes in Computer Science*. Springer Verlag, 1999.

[5] K. Murphy, M. Abrams, S. Balakirsky, T. Chang, A. Lacaze, and S. Legowik. Intelligent control for off-road driving. In *First International NAISO Congress on Autonomous Intelligent Systems*, 2002.

[6] W. P. Shackleford, F. M. Proctor, and J. L. Michaloski. The neutral message language: A model and method for message passing in heterogeneous environments. In *Proceedings of the 2000 World Automation Conference*, June 2000.

[7] C. M. Shoemaker and J. A. Borenstein. Overview of the demo III UGV program. In *Proc. Of the SPIE Robotic and Semi-Robotic Ground Vehicle Technology*, volume 3366, pages 202–211, 1998.

[8] S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. MINERVA: A second-generation museum tour-guide robot. In *Proceedings 1999 IEEE International Conference on Robotics and Automation*, volume 3, pages 1999–2005, May 1999.