Simulation and Testbeds of Autonomous Robots in Harsh Environments

Richard S. Stansbury[1], Eric L. Akers[1], Hans P. Harmon[2], and Arvin Agah[1]

[1] University of Kansas richss@ku.edu, eakers@ku.edu, and agah@ku.edu
[2] Pinnacle Technology, Inc. hharmon@pinnaclet.com

1 Introduction

Software development for autonomous field robots can be quite challenging due to the difficulties associated with testing and evalution of the robot and its components. Software for the mobile robot must undergo extensive testing throughout its lifecycle. However, testing a robot for harsh environments requires access to the field, logistic support, maintenance of the robot, and adherence to safety standards. Time, budgetary, and logistics constraints often limit the amount of testing, which in turn adversely affects the quality of the software developed for the field mobile robot.

A harsh environment is one that is very difficult and possibly dangerous to traverse because of obstacles or difficult terrain, dangerous climate, or a location that is difficult to reach by human or robot. For the purposes of this chapter, the harsh environments referred to are the polar regions. These locations have extreme cold temperatures, are very windy, sastrugi and crevasses make traversal difficult and dangerous, and they are difficult to reach.

Through simulation and robot testbeds, software for field robots can be designed, developed, and tested, while effectively addressing many of these challenges. Virtual prototypes can be used to realistically simulate the field robot for hardware and/or software design. Testbed platforms can be used to test and refine the robot software within a real environment. The resulting software can then be implemented on the field robot. It should be noted that although this approach is of great benefit to software development, it does not eliminate the need for eventual testing in the field because the virtual prototype and testbed are only an approximation of the real system. This methodology should, however, reduce the time required to physically test the robot in the field and improve the chances of success. The transition from simulation to physical robot includes iteration of testing and evaluation, modifying the simulation, and retesting and reevaluating, and it is this step that is the most difficult when dealing with harsh environments.

D. Brugali (Ed.): Software Eng. for Experimental Robotics. STAR 30, pp. 71–92, 2007.

For the evolution of software from a virtual prototype to the real robotic system, a software architecture for the mobile robots must exist. The architecture must allow the reusability and portability of software between virtual and physical platforms. The architecture must also support the interaction between the robotic components both locally and in a distributed fashion.

Virtual prototypes can be used to produce a fully simulated and realistic representation of the robot and its environment. The simulated robot can be controlled within the simulator in order to test its software, as well as to determine various constraints and design parameters that could affect the software or hardware of the robot.

Smaller mobile robots may also act as testbeds by providing analogous sensors and actuators. Higher-level control software can be developed and refined on a testbed within a laboratory environment, reducing the need for field experiments. Software development on an analogous robot provides further opportunity to develop software functionality and to refine robot behaviors.

In this chapter, a polar robot developed as part of the Polar Radar for Ice Sheet Measurement (PRISM) project at the University of Kansas [AHSA04, SAHA04] will be presented as a successful case study for the application of the proposed methodology. The PRISM project will be briefly introduced as well as the autonomous field robotic systems used as part of this research. A software architecture for mobile robots will be presented. Next, the virtual prototype of the PRISM mobile robot is described. The transitions between the prototype, the testbed, and PRISM mobile robot is discussed. Finally, the performance of the robot in Greenland is presented.

1.1 Polar Radar for Ice Sheet Measurement Project

The PRISM project is currently underway at The University of Kansas [KU 04], with the goal to develop radar systems to measure polar ice sheet properties in order to accurately determine their mass balance and other characteristics. Such data will help scientists to determine and to better model the contributions of polar ice sheet melting to global climate change and its effects on the rising sea levels.

In order to accommodate a pair of bistatic synthetic aperture radars (SAR), two vehicles have been employed, carrying the transmitter and the reciever components of the SAR, respectively. The first is a manned tracked vehicle that will tow an antenna array for the bistatic/monostatic SAR and will also carry an on-board dual-mode radar. The second is an autonomous robot that will be utilized to carry the second SAR. When operating in bistatic mode, the robot's SAR will act as the receiver. The two vehicles will move along side one another in a coordinated pattern. The autonomous robot will traverse a wide area along side the tracked vehicle collecting data to build the radar image.

Challenges of Polar Traversal

In June of 2001, a workshop analyzed the feasibility and resourcefulness of using mobile robots similar to planetary rovers for collection of scientific data on the ice sheets [CSB01]. This group defined several tasks with which an autonomous rover could aide. These tasks include: traverses with detailed and tedious paths, extremely remote and/or inhospitable environments, data collection parallel to a manned traversal, and data collection at slow speeds. In 2002, researchers at NASA's Jet Propulsion Laboratory proposed that utilizing mobile robotics for polar exploration can provide future benefits toward planetary exploration [BCW02].

Both groups discussed the numerous challenges that exist with polar traversals. These challenges include deep and blowing snow, crevasse detection and avoidance, sastrugi detection and avoidance, and limited supervision. Operating in the polar regions can be more challenging than in more hospitable environments. Equipment may be damaged as the result of sub-zero temperatures, high wind speeds, and blowing snow.

Several unique terrain-based obstacles present themselves in the polar environments. For instance, sastrugies which are snow drifts that form as the result of wind erosion, have heights of up to one meter. In addition, crevasses in the ice sheet are encountered that remain invisible because they are covered by snow. Vision can be limited due to the lack of contrast on the icy surface. With this lack of depth perception, natural obstacles can be overlooked.

PRISM Robotics Testbed

A Nomadic Scout mobile robot [Nom99a] acts as a testbed for development of higher-level software applications. This platform, also known as Bob, acts as a scaled down version of the PRISM mobile robot. Similar to its larger counterpart, it possesses a position sensor, a heading sensor, and obstacle detection sensors. It utilizes differential drive which is similar to the skid-steering of the PRISM robot. Therefore, control software such as waypoint navigation and the precise motion required by the PRISM mobile robot can be tested within the laboratory first, under controlled conditions.

The Nomadic Scout mobile robot base [Nom99a] has a drive system that is comprised of two motors driving two wheels. Each wheel motor is equipped with shaft encoders. The shaft encoders are used to approximate both the heading and the position of the mobile robot. Obstacles are detected using an array of sonar transducers. Communicating via a simple ASCII language allows outside programs to query sensors for data, receive sensor data, and control the actuators. A Sony Vaio Picturebook laptop [Son04] executes Bob's control software and drivers. A variety of sensors utilized by the field robot were also utilized when performing software tests with Bob. Bob operating in the laboratory is presented in Figure 1.

Fig. 1. Bob: a test platform for the PRISM Mobile Robot software.

Mobile Autonomous Robot with Intelligent Navigation (Marvin)

In order to meet the requirements of the PRISM project, the Mobile Autonomous Robot with Intelligent Navigation (Marvin), shown in Figure 2, was designed and built. This mobile robot is the field robot for which the software techniques discussed within this chapter are utilized (referred to as the PRISM Mobile robot, or Marvin, interchangeably).

The MaxATV Buffalo [Rec04], an amphibious six-wheeled ATV, was selected for the mobile base platform of Marvin. With tracks installed, the surface pressure of the Buffalo is greatly reduced, allowing it to better travel across snow and ice. In order to automate the platform, three components had to be placed under actuator control. The left and right brakes levers had to be controlled as they allow the turning of the vehicle. The throttle had to be controlled using an actuator. Magnetically driven linear actuators were used to automate the platform.

The sensor suite selected for the PRISM mobile robot included a Topcon Legacy-E RTK GPS system that was selected to support the robot's scientific objectives, as well as its navigation and localization. The system was composed of a fixed base station and a mobile rover system, and provided centimeter-level position accuracy [Mar02]. The LMS221 laser range finder by Sick AG was used to detect obstacles, and it was also equipped with an internal heater for operating in temperatures below 0°C [SIC98]. An inertial measurement unit provided heading information while the robot turned, as well as monitored its acceleration. It acted as three sensors in one, namely, a three-axis gyroscope, a 3-axis accelerometer, and an internal temperature sensor.

Fig. 2. Marvin operating in Greenland at the Summit Camp

2 A Software Architecture for Autonomous Mobile Robots

A software application programming interface (API) was developed for PRISM mobile robotics research. The API was designed with an emphasis on portability and robustness. As the underlying hardware of the mobile robot changes, the overall structure of the software system should remain intact and only require minor changes to adapt to the new hardware. The software libraries should also be general enough so that they could be utilized for a variety of mobile robots. In order to accomplish these goals, careful design of the software API was required to emphasize abstraction of the higher-level logic from the underlying hardware. The Java [Sun04b] programming environment was utilized to develop the system. It facilitates object-oriented design, which is necessary to provide the abstraction of higher-level control software from the lower-level hardware drivers.

In this section, the design of the software API is discussed. In general, a mobile robot is composed of actuators, sensors, and a higher-level control system that produces actuator output based on sensor input and outside control parameters. Figure 3 illustrates the general design of the software API and how it interfaces with more complex software systems.

2.1 Hardware Abstraction

Software portability was a major goal for designing the API for controlling the PRISM mobile robot. This would allow much of the software developed for one mobile platform to be usable with others. Such portability can greatly aide the development of a mobile robot for field applications. Using this approach, various aspects of the field mobile robot can be tested indoors on smaller mobile robots.

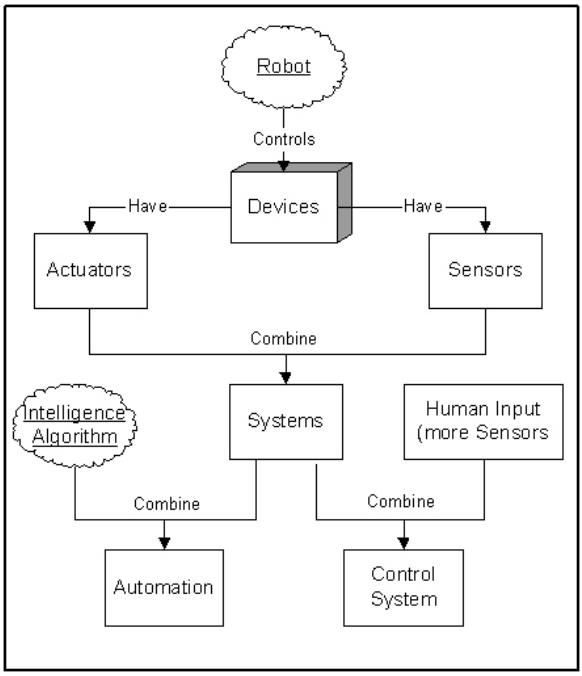

Fig. 3. The PRISM Robotics Software API.

Accomplishing hardware abstraction makes it necessary to write device drivers, or wrapper code, that implement the interfaces necessary for other software components to interact with the hardware. The robot API defines these interfaces such that the hardware can be viewed both generically as its overall purpose (sensor or actuator) and more specifically based on its actual function (e.g., bump sensor, linear actuator, etc.).

Sensors Interfaces

The *Sensor Interface* defines the most elementary representation of a sensor. This class is implemented subsequently by all other sensor interfaces. Its purpose is to define all sensors under common terms. These interfaces are implemented as device drivers or wrappers for a uniform interface for accessing data that are produced by the actual sensor hardware.

A sensor may either poll for new data, or wait for an Input/Output (I/O) event to specify that new data are available. Data from an instantiated sensor class may be accessed in one of two ways. The requesting application may query the sensor for the current data. Alternatively, the application may register itself with the sensor such that whenever the sensor data are updated,

the application will receive a data update event. The Sensor interface of this API allows data access through either method.

Interfaces that extend this basic sensor interface are also defined. These new interfaces provide additional abstraction of the sensor data from the underlying sensor. The *Position Sensor Interface* defines sensors that help the mobile robot navigate over a two-dimensional plane such as a dead reckoning system, or the longitude and latitude of a GPS receiver. The *GPS Receiver Interface* extends the *Position Sensor Interface* by adding elevation and time. The *Heading Sensor Interface* defines a variety of sensors that measure the current heading or yaw orientation. The *Tilt Sensor Interface* facilitates orientation sensors that measure the vehicle's orientation in the pitch and roll (tilt) directions. The *Distance Sensor Interface* defines sensors that measure the distance in meters from nearby objects, i.e., the obstacle detector.

Several other sensors are specified by the robotics API. For each sensor type, a unique interface is created to define the means of accessing the available data. Other sensor types include: temperature, bump, weather, force, current, level, etc.

Actuator Interfaces

Actuators allow the mobile robot to act upon the world. Similar to sensors, some actuators are capable of providing feedback regarding their current state. However, often the actuators are unaware of their current state and simply perform the requested action.

For the robotics API, two basic actuator interfaces are defined, namely, the *Motor Interface* and the *Switch Interface*. Each of these extends a more general *Actuator Interface*. Other categories of actuators do exist. However, for PRISM, these two categories encompassed all of the actuator classes required by polar mobile robot. For each category of actuator, abstract classes provide a general definition of more specific actuators. Figure 4 shows the actuator hierarchy for the API.

A motor causes an object to move in the real world. The *Motor Interface* defines the most basic requirements of a motor such as the ability to tell it to stop all motion. However, to define more specific motor type, the motor type actuators were divided into three sub-types of wheel motors, servo motors, and linear motors.

Switches are capable of changing the current state of the mobile robot without the use of motion. Switches are useful for controlling circuits and the transfer of current through robotic subsystems. A binary switch has only two states, on and off, such as momentary switches, toggle switches, and relays. Linear switches control devices that have a range of input values. For instance a dimmer switch controls the rate of current flow to a light bulb over a given range.

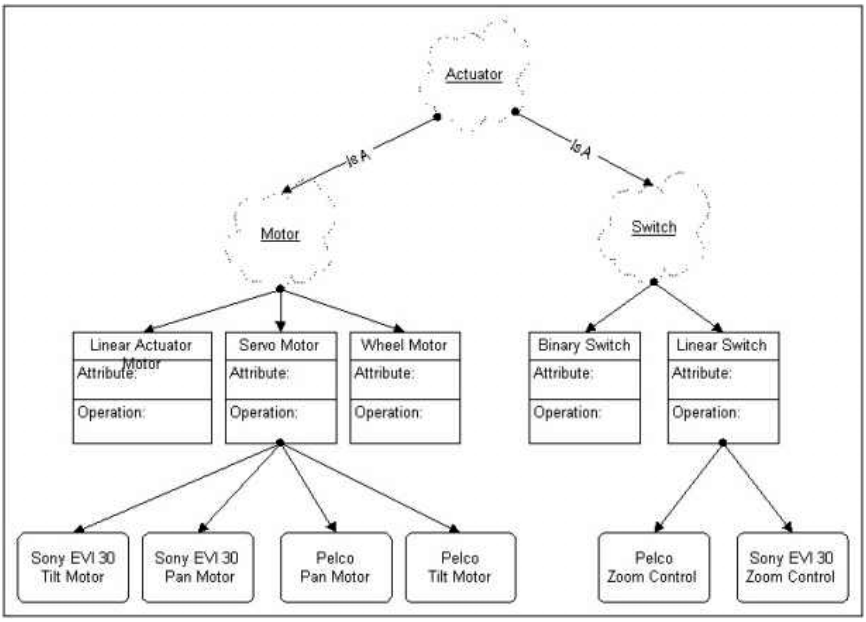

Fig. 4. PRISM Mobile Robot Actuator API.

2.2 Movement2D Interface

The *Movement2D Interface* defines an extension of the Robotics API for more intelligent movement and abstracts away from the underlying mechanism for motion. The extension to the API is shown in Figure 5. This gives a common interface to control the mobile robot regardless of the underlying hardware. The controller sets the left and right velocities for the vehicle's drive system. This defines three potential drive systems which can be utilized for mobile robots.

The *Skid Steer* defines control of vehicles that utilize skid-steering such as the PRISM field robot. In hardware, this system is made up of three linear actuators to control the left brake, the right brake, and the throttle. Given an input of left or right velocities, the linear actuators are sent control signals to define their appropriate position. The *Differential Drive* controls vehicles whose drive system is comprised of two independently driven wheel motors. For this system, the left and right velocities directly translate to wheel motor speeds. The Nomadic Scout utilizes this system. The final drive system, which was not implemented, is the *Steering Drive*. This represents drive systems in which the front wheels are steered in order to direct the heading of the vehicle such as in cars. Such a system can be implemented using a servo for steering and linear motors for throttle and brake.

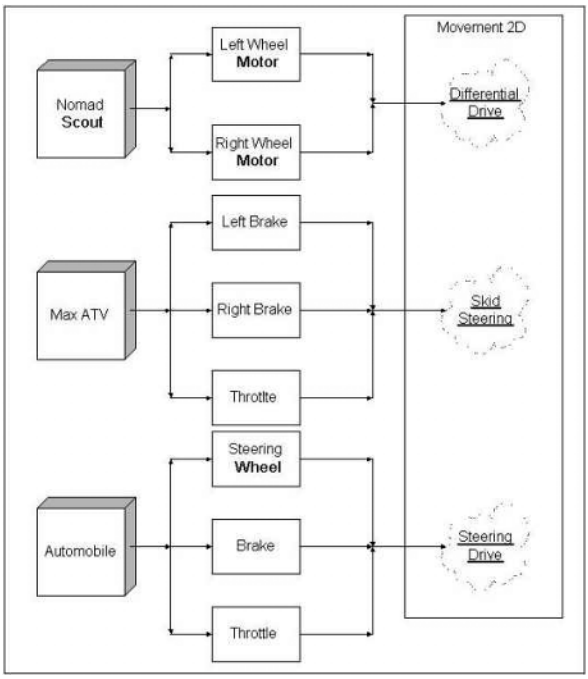

Fig. 5. Extension to the API for two-dimensional movements.

2.3 Challenges and Limitations

Using the object-oriented paradigm, the PRISM Robot API supports distributed computation, control, and sensing. The software architecture is portable and reusable, and through careful abstraction, software can be transitioned between systems without the need to re-implement the control system. However, there are some limitations to the proposed architecture. While the system is portable across platforms, it does limit the developers primarily to Java and Java-based libraries. Java Native Interfaces allows developers to access libraries written with other programming languages. However, depending upon the complexity of the external software, the development of such interfaces may be more difficult.

2.4 Other Software Architectures for Autonomous Mobile Robots

A number of research efforts have focused on software design for mobile robot systems. A common goal of software systems for autonomous robots is to create a system that can be utilized on multiple different platforms with little or no modification. The University of Florida created such a system that can be moved between platforms without modification [ACNW00]. That system

divided up the various robotics tasks into more abstract modules. These included the Mobility Control Unit (MCU), Path Planner (PLN), Position System (POS), Detection and Mapping System (DMS) and a Primitive Driver (PD). This division of the robotic components helped inspire the Join Architecture for Unmanned Systems (JAUS).

JAUS, lead by the Department of Defense, seeks to define robotic control systems for a variety of physical platforms. Such platforms include Uninhabited Air Vehicles (UAVs), Uninhabited Ground Vehicles (UGVs), Uninhabited Underwater Vehicles (UUVs), etc. Software for JAUS is abstracted into a four-level hierarchy. For each layer, the software abstracts further away from the actual hardware. These are the component, node, subsystem, and system levels. The component level provides a direct interface with the hardware. The node level abstracts from the component level by hiding the details of the underlying hardware and accounting for redundancies in component level systems. The subsystems level represents software components that control nodes (e.g., a single uninhabited system). Finally, the systems level acts as a planner that controls the subsystems to achieve higher-level goals [JAU02].

The proposed software architecture shares similar goals with the Player architecture developed by the University of Southern California [GVH03, VGH03]. Unlike the PRISM robotics architecture, Player's primary focus was the control of multi-robot systems. Therefore, more concern was placed upon lowering overhead in order to increase scalability for a large groups of collaborative robots. Similar to the PRISM software architecture, Player relies upon the interface/driver model for interaction with the robot's hardware while abstracting away the underlying details of the robot's actual hardware. Similar to Unix-based systems, for each hardware driver, Player treats the device as a file that can be assigned read or write access for communication. Player also utilizes TCP sockets for communication between software modules within a system.

3 Virtual Prototyping

The PRISM mobile robot's operating conditions in Kansas are quite different than that of the polar regions in Greenland and Antarctica. Properties such as the robot's maximum turn speed, how the robot would react when it impacts an obstacle, or how the robot would handle climbing various slopes are difficult to obtain outside of the field. These properties can strongly influence the design and construction of the mobile robot. Factors such as weight distribution and center of gravity of the robot can also affect these properties. In order to design the PRISM mobile robot and test its abilities, it was first modeled and constructed using the MSC.visualNastran 4D [MSC04b] simulation package. Numerous experiments were performed using the simulated robot.

3.1 Virtual Prototyping of Mobile Platforms

Virtual prototyping involves the development of a realistic model (kinematics and/or dynamics) to represent the mechanical system that is to be developed. An environment can also be simulated within which the mechanical model functions. Certain simulation software packages allow the model to be controlled within the environment and utilize a physics engine to simulate the dynamics of the environment. Within the robotics domain, a few simulation packages also provide interfaces so that the robot can be controlled externally, and the data from within the environment can be sensed through virtual sensors. Virtual prototyping is not unique to robotics as it has been utilized in many other research and commercial domains such as mechanical engineering.

Researchers at the University of Perugia in Italy [BCO04] have focused on the analysis and the design of snowmobiles and methods to improve their design. The researchers used the MSC.ADAMS [MSC04a] software to build the model of the snow track vehicle and to test it. They presented a working model and the analysis of all the different parts of the snow track vehicle. The components that were modeled and tested included the track, suspension, frame, upper structural components (cabin and motor), auxiliary rope traction system (winch), front snow shovel, and the rear snow-crushing device.

The model and its components were tested on plane ground, rough ground, a 15-degree slope, and a 30-degree slope. The results included the histories of data such as the track force, gear angular velocity, the velocity of the vehicle, and much more. This work illustrates how different components can be modeled and tested, and the data that can become available by using such methodology.

A research project at Helsinki University of Technology [AKS00], describes the simulations used to study the load balancing and stability of arobot. Two models were used, a kinematics model and a dynamic model. The dynamic model was the same as the kinematics model, except that dynamic properties such as mass, inertia, and ground contact forces were added to the model. The kinematics model was used only for locomotion visualization and monitoring purposes. The dynamic model was used for torque and stability analysis.

Several software applications exist that explicitly support the simulation of robotic systems. As discussed in [KH04], virtual prototypes for mobile robots may be utilized for the co-design of the robot's hardware and software systems. Webots [Mic04] is a commercial application for simulating and controlling mobile robots within a virtual environment. It provides a flexible interface so that it can be controlled using programs written in C++ or Java. Two open source tools have features similar to Webots. Gazebo [KH04] generates 3Ds environment and models to represent mobile robots and their environment. It simulates a variety of sensor types such as position, ray proximity (obstacle detection), and a virtual camera. It is designed such that software can be written to control the model using the Player [GVH03] robot software archi-

tecture. A controllable simulation package was not available at the start of the PRISM project.

3.2 Modeling the PRISM Mobile Robot

Prior to the selection of a specific mobile platform, the Nastran simulation package provided the means for virtual prototyping a variety of mobile platforms to determine their performance in polar-like conditions.

The MaxATV Buffalo which was selected as the mobile platform for the PRISM mobile robot had an optional track kit. It was of interest to compare the performance of the vehicle when driven by six-wheels versus with tracks. A model of the new vehicle was constructed, with extra care given to distribute the vehicle's weight accurately among each component. The tracks were realistically modeled such that they covered the appropriate surface area on the snow and ice. The models for these configurations are shown in Fig. 6.

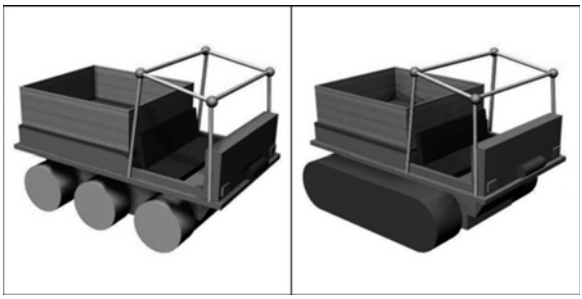

Fig. 6. Modeling the base platform with and without tracks.

In order to simulate Marvin, various robotic components had to be modeled. The rackmounts and generator were modeled as rectangular boxes. The roll cage and truck bed were removed from the existing MaxATV model as these components were replaced by a winterized enclosure. The winterized enclosure was modeled with the use of lexan for the windshield and an aluminum/plastic composite material for the sides and the roof. The simulator allowed the modification of the enclosure so that it could contain the various components onboard the robot. The configuration was modified until the best load balance and lowest center of gravity was developed given all the onboard components. The resulting model is shown in Figure 7, along with a picture of the actual robot.

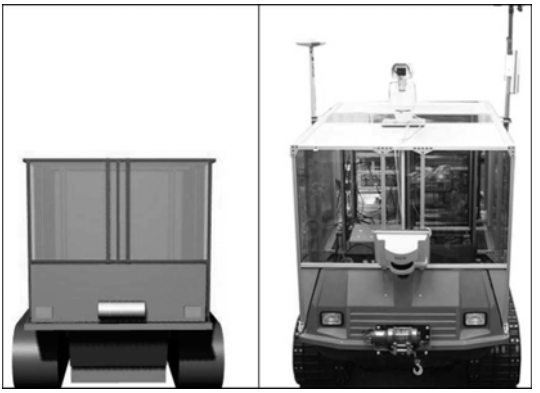

Fig. 7. Modeling the robot and its structure.

4 From Prototype to Testbed

Using the virtual prototype and software architecture, the software for the PRISM mobile robot evolved as transition between a prototype simulation, software development and testing using a testbed platform, and the final implementation of the software for the field robot. In this section, the simulation of the PRISM robot is discussed in further detail with a focus on how these results influenced the software development. Next, several robotic behaviors implemented on the testbed platform are presented. Software reuse and portability is demonstrated as the software is seamlessly evolved to control the field robot.

4.1 Simulation of PRISM Mobile Robot

As discussed previously, the simulation package used did not provide an adequate interface to create a virtual prototype that could be controlled using software for the PRISM mobile robot. Instead, initial software development was performed using the testbed platform. However, the simulation experiments produced results that influenced not only the hardware design of the robot, but also helped define several requirements of the controller.

The simulation experiments were intended to determine how well the robot performs basic tasks such as towing an antenna while turning, and what slopes the robot could climb. This information was necessary to design and build the robot, as it have to be able to traverse potentially harsh terrain. The tests estimated a number safe running parameters such as at what point the robot would tip over when driving over uneven surface. Without these tests, much more experimentation would have to be performed at the actual field site which is difficult due to the logistical limitations.

The experiments performed were designed to answer some specific questions about the performance of the robot. One objective was to decide the starting point be for the safety parameters. The term starting point is used because the model is only an approximation of the real world, therefore, the answers from the experiments should only give an approximation. The safety parameters include the maximum slope the robot could climb (the pitch angle) and the point at which it might roll over (the roll angle). Another objective was to determine how the robot would handle while performing basic movements and with different load configurations. The experiments were grouped into three series of experiments. The first series of experiments generated a baseline of how the model performed. These experiments were performed without any load on the vehicle. The second series of experiments were performed with the antennas and towing mechanisms added into the model. The antenna experiments were performed with different antennas, different towing mechanisms, and at different speeds. The third series of experiments were performed with a single antenna and weights added to the inside of the model to simulate a fully loaded vehicle.

4.2 Software Implementation for Testbed Mobile Robot

The Nomadic Scout platform is a single system treated as six individual components. It is composed of a left motor, right motor, bump sensor, position sensor, sonar sensor, and heading sensor. Input and output for all of these sensors occur through a single communication interface. Drivers were written for each of the robot's sensors and actuators. Each of the wheel motors was passed to the differential drive movement class in order to simplify the control of the mobile robot's drive system.

Some typical robot behaviors were developed using the testbed such as waypoint navigation and obstacle avoidance using a wall-following algorithm. Using the testbed, each of these behaviors could be tested within the controlled environment of the robotics laboratory without many of the risks involved with testing a much larger field robot. These algorithms were developed using class definitions for abstract robot components to facilitate the transition between robot platforms. This approach produced reusable control software for the mobile robots.

Once these behaviors were refined for the testbed, more complex behaviors were developed to meet the requirements of the PRISM project. In particular, a movement pattern was defined for the collection of radar data using the synthetic aperture radar. This movement pattern, shown in Figure 8 will be referred to as the SAR path. The SAR path directs the robot to make parallel swaths of a specified length at a fixed width apart.

The code section presented in Figure 9 configures the SAR Navigator to operate on the testbed using its suite of sensors and differential drive movement. This code will later be compared with an equivalent section of code from the field robot's controller in order to demonstrate the transition requirements

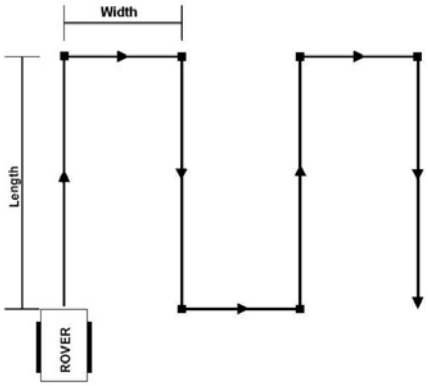

Fig. 8. The SAR path required for radar data collection.

From scout.java

```
1  //Connect to Nomadic Scout
2  NomadScoutAsDevice nc = new NomadScoutAsDevice(SCOUT_PORT_KEY,
3                                                 UPDATE_RATE);
4
5  //Sensors Setup
6  Sensor [] sensors = nc.getSensors();
7  PositionSensor position = (PositionSensor) sensors[nc.POSITION];
8  HeadingSensor heading   = (HeadingSensor)  sensors[nc.HEADING];
9
10 //Drive System Setup
11 Actuator [] motors  = nc.getActuators();
12 WheelMotor left     = (WheelMotor) motors[nc.LEFT];
13 WheelMotor right    = (WheelMotor) motors[nc.RIGHT];
14 Movement2D movement = new DifferentialDriveSystem(left, right);
15
16
17 //Build SAR Navigator given sensors and drive.
18 Navigator navigator = new SARNavigator(movement,
19                                        position,
20                                        heading);
```

Fig. 9. Code: Configuration of the testbed robot for SAR path navigation.

between platforms. Figure 10 shows the testbed following the SAR path within the laboratory.

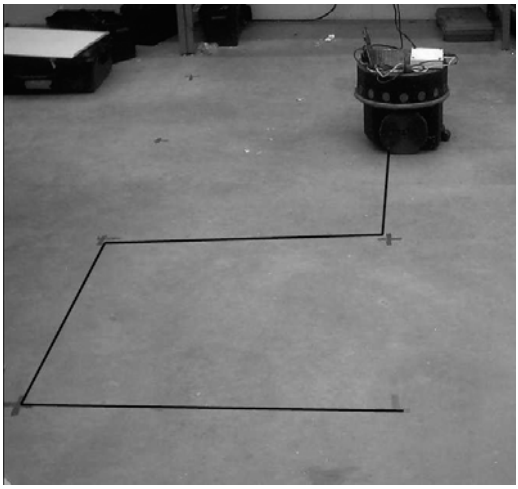

Fig. 10. Approximate path followed by the testbed robot navigating the SAR path.

4.3 Software Implementation for PRISM Mobile Robot

Once software development was completed for the testbed robot, the software was ported to the real field robot. Testing and refinement of the control software then began for the new platform. However, use of the testbed did not cease; any major changes to the control software were first validated using the testbed before they were transitioned to field robot. This approach preserves the requirement of backwards compatibility between the testbed's and the field robot's control behaviors.

In order for the transition between platforms to be performed, drivers were written for each of the robot components utilized by the field robot (and not by the testbed robot). Each driver was developed independently using the software API's available driver interfaces and then tested prior to integration. With the goal of ensuring portability, drivers for the field robot's hardware were implemented such that they were backward compatible with more generic components. For instance, the Topcon GPS receiver's drivers implemented both the *GPSReceiver* interface as well as the *PositionSensor* interface such that it would operate seamlessly with the SAR path navigator.

Figure 11 illustrates the simplicity of the software transition from the testbed to the field robot, i.e., its reusability and portability. Similar to the previous code section, the SAR Navigator receives as input a movement controller, a position sensor, and a heading sensor. Analogous to the *DifferentialDriveSystem* used for the testbed robot, the *SkidSteeringDriverSystem* implements the *Movement2D* interface. Therefore, the navigator can command the robot to perform the same set of actions. Similarly, the drivers for the Top-

From Marvin.java,

```
1  //Sensor Configuration
2  PositionSensor position  = new TopconGPSReceiver(GPS_PORT);
3  HeadingSensor heading    = new Motionpak2(GYRO_PORT);
4
5  //Actuator Configuration
6  LinmotControlBox lcb4000 = new LinmotControlBox(STEERING_PORT,
7                                                  CONTROLLER_ID,
8                                                  NUM_ACTUATORS);
9  Motor [] motors     = lcb4000.getMotors();
10 Movement2D movement = new SkidSteeringDriveSystem(
11                         (LinearActuatorMotor) motors[LEFT],
12                         (LinearActuatorMotor) motors[RIGHT],
13                         (LinearActuatorMotor) motors[THROTTLE]);
14
15 //Build SAR Navigator given sensors and drive
16 Navigator navigator = new SARNavigator(movement,
17                                        heading,
18                                        position);
```

Fig. 11. Code: Configuration of the field robot for SAR path navigation.

con GPS Receiver and the MotionPak II Gyroscope implement the *Position* and *HeadingSensor* interfaces, respectively.

Figure 12 plots the movement of the field robot along the SAR path in a field at the University of Kansas. Through slight modifications to the controller, the robot makes smoother turns toward the next waypoint in order to protect the radar array that is being towed.

By utilizing the PRISM robotics API, software reuse and platform independence emerges. This is demonstrated by porting the software from a testbed robot to a field robot. Not all of the software could be ported, but through abstraction, all higher-level control is reusable. Only drivers for the platform dependent hardware must be re-implemented with every platform change.

4.4 Testbeds for Mobile Robot Design and Testing

A common practice for mobile robot design and development is the utilization of a testbed system so that the designers can focus on the development of software and the control system on a stable mobile platform. Performing experiments to test the software becomes easier and more efficient because the tests can be performed in a controlled environment, thus eliminating limitations such as resource availability, space, and logistics. Many theoretical concepts may be discovered and revised prior to being incorporated into the production system.

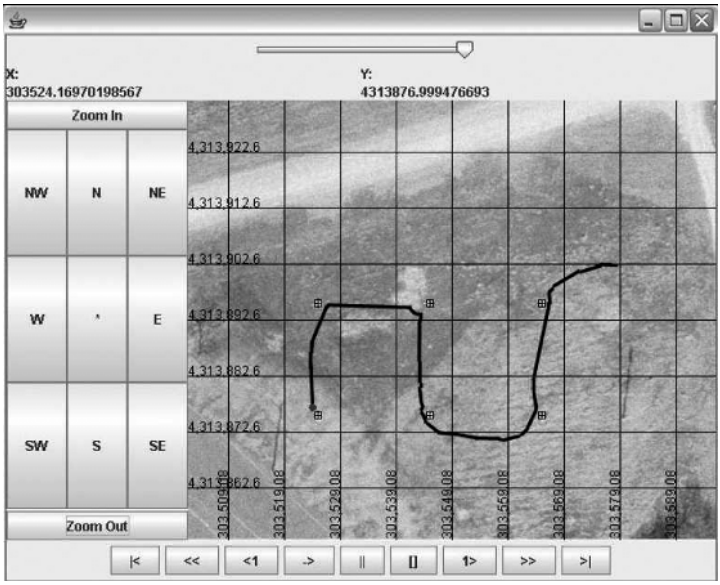

Fig. 12. The precise path followed by the field robot while performing SAR path navigation.

Carnegie Mellon University's (CMU) Robotics Institute has conducted research in the area of autonomous land vehicles. The goal of this research was to develop a mobile robot that was capable of navigating along sidewalks, or along curved roads. To accomplish these tasks, two mobile platforms were employed. The Terregetor mobile robot acted as a testbed [GS87]. It was a custom-built mobile robot platform small enough to safely travel along sidewalks. Its larger counterpart was NAVLAB, an automated delivery truck. This is an example of an application for which testing near the university campus was much easier and safer whenever a testbed mobile robot was utilized, instead of the full sized system.

As part of DARPA's Tactical Mobile Robotics Program, researchers at the Georgia Institute of Technology have worked to develop a mobile robot capable of urban warfare and reconnaissance applications. Their software system is developed so that it may execute on multiple platforms [ACE99]. One such platform is the Pioneer AT developed by ActivMedia Robotics [Act04]. This testbed is designed for both indoor and outdoor robotic applications and is capable of transporting several mobile robot sensors.

The utilization of testbed systems is also common on much larger projects such as the Mars rovers. These testbed systems much more closely resemble the rovers that travel to Mars. However, they provide stable platforms for which new hardware and software components can be tested and integrated. For instance, the Athena Software Development Model (SDM) rover

at NASA's Jet Propulsion Laboratory represents prototypes for the planetary rovers, Spirit and Opportunity, that successfully traversed the Martian surface [SHP03, BMM01].

5 Conclusion

Development of autonomous mobile robots for field applications presents many unique challenges. These challenges are compounded for robots operating in harsh environments such as polar regions. Testing the mobile robot within these environments can be both difficult and costly.

In order to meet these challenges, a software development methodology for mobile robots has been presented that results in portable and reusable software for such robots. A virtual prototype is used to simulate the mobile robot to develop various design parameters for the software system and the hardware system. Some simulators provide virtual prototypes with interfaces such that they can also be used for initial software development. Testbed platforms such as small indoor (or outdoor) mobile robots are then used to develop the robot's higher-level behaviors such as obstacle avoidance and waypoint navigation. Finally, these behaviors are transitioned to the field robot and tested both locally and in the field.

The robotics API was developed as a platform independent architecture for mobile robots. The API can be expanded for other robotic endeavors and othersensor/actuator modalities for robots. Robot control software was developed using the API to produce autonomous mobile robots for the PRISM project. It was first developed and tested using a testbed mobile robot, and the software was then transitioned to control the field robot.

During the summer field seasons of 2004 and 2005, field experiments were conducted at Greenland's North GRIP and Summit camps, respectively. The first field experiments provided the proof of concept that the virtual prototype had successfully produced a mobile robot capable of operating within the polar environment. During the second field experiments, the mobile robot was automated using the robotics API. The robot was integrated with the SAR radar equipment and autonomous data collection tasks were performed. Using the proposed methodology, a viable mobile robot for field applications was produced.

This research is not without some limitations. Improvements could be made to both the simulation and the implementation of the software architecture. The simulator was capable of modeling the robot and general properties of the environment, but is not yet sufficient to simulate more specific hazards of the environment. A simulation environment with additional features would also include hazards such as rough terrain. Virtual prototypes may also be used for initial development of software for robot control and autonomy.

Similar to other software development endeavors, the implementation of the software architecture can always be expanded. The robotics API can be

augmented to include additional sensor and actuator modalities. The architecture includes components (sensors and actuators) that were encountered and utilized within the PRISM robotics laboratory. As the software architecture grows and matures, it is envisioned that more formal techniques may be applied to the design of software for mobile robots.

Acknowledgements
The work described in this chapter was supported by the National Science Foundation (grant #OPP-0122520), the National Aeronautics and Space Administration (grants #NAG5-12659 and NAG5-12980), the Kansas Technology Enterprise Corporation, and the University of Kansas.

References

[ACE99] Ronald C. Arkin, Thomas R. Collins, and Yoichiro Endo, *Tactical mobile robot mission specification and execution*, Mobile Robots XIV **3838** (1999), 150–163.

[ACNW00] David G. Armstrong II, Carl D. Crane III, David Novick, and Jeffrey Wit, *A modular, scalable, architecture for unmanned vehicles*, Association for Unmanned Vehicle Systems International (AUVSI) Unmanned Systems 2000 Conference (Orlando, Florida), July 2000.

[Act04] Active Media Robotics, *Robots and Robot Accesories*, URL: http://www.activrobots.com, 2004.

[AHSA04] Eric L. Akers, Hans P. Harmon, Richard S. Stansbury, and Arvin Agah, *Design, fabrication, and evaluation of a mobile robot for polar environments*, IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2004) (Anchorage, Alaska), vol. I, September 2004, pp. 109–112.

[AKS00] P. Aarnio, K. Koskinen, and S. Salmi, *Simulation of the hybtor robot*, 3rd International Conference on Climbing and Walking Robots (Madrid, Spain), October 2000, pp. 267–274.

[BBD00] Greg Bollella, Ben Brosgol, Petter Dibble, Steve Furr, James Gosling, David Hardin, Mark Turnbull, and Rudy Belliardi, *The real-time specification for java*, Addison-Wesley, 2000.

[BCO04] C. Braccesi, F. Cianetti, and F. Ortaggi, *Modeling of a snow track vehicle*, Institute of Energetics, Faculty of Engineering, University of Perugia, Italy, 2004.

[BEI01] BEI Systron Donner, *Bei motionpak ii: Multi-axis inertial sensing system*, BEI Systron Donner Inertial Division, Concord, 2001.

[BMM01] Jeffrey Biesiadecki, Mark Maimone, and Jack Morrison, *The athena sdm rover: A testbed for mars rover mobility*, 6th Intl. Symp. on AI, Robotics and Automation in Space (Montreal, Canada), no. AM026, June 2001.

[Bot04] Per Bothner, *Kawa, the java-based scheme system*, URL: http://www.gnu.org/software/kawa/, 2004.

[CSB01] F. Carsey, P. Schenker, J. Blamont, S. Gogineni, Kenneth Jezek, Chris Rapley, and William Whittaker, *Autonomous trans-antarctic expeditions: an initiative for advancing planetary mobility system technology while addressing earth science objectives in antartica*, Sixth International Symposium on Artificial Intelligence, Robotics and Autonomy in Space (Montreal, Canada), no. AM017, June 2001.

[FH04] Ernest Friedman-Hill, *Jess, the rule engine for the java platform*, URL: http://herzberg.ca.sandia.gov/jess/, 2004.

[GS87] Y. Goto and Anthony (Tony) Stentz, *Mobile robot navigation: The cmu system*, IEEE Expert **2** (1987), no. 4, 44 – 55.

[GVH03] Brian P. Gerkey, Richard T. Vaughan, and Andrew Howard, *The player/stage project: Tools for multi-robot and distributed sensor systems*, Proceedings of the International Conference on Advanced Robotics (ICAR 2003) (Coimbra, Portugal), July 2003, pp. 317–23.

[Han04] Chris Hanson, *The scheme programming language*, http://www.swiss.ai.mit.edu/projects/scheme/, 2004.

[HSAA04] Hans P. Harmon, Richard S. Stansbury, Eric L. Akers, and Arvin Agah, *Sensing and actuation for a polar mobile robot*, International Conference on Computing, Communications and Control Technologies (Austin, Texas), vol. IV, August 2004, pp. 371–376.

[JAU02] JAUS Work Group, *Joint architecture for unmanned systems*, United States Department of Defense, September 2002.

[KH04] Nathan Koenig and Andrew Howard, *On device abstractions for portable, reusable robot code*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004) (Sendai, Japan), September 28 - October2 2004, pp. 2149–2154.

[KU 04] KU PRISM Team, *Prism home: Polar radar for ice sheet measurements*, URL: http://www.ku-prism.org, 2004.

[Lai99] Sheng Laing, *The java native interface: Programmer's guide and specification*, Addison-Wesley, Reading, MA, 1999.

[Lin04] Linmot Inc., *Linmot data book 2003-2004*, URL: http://www.linmot.com/datasheets/LinMot Data Book 2003-2004.pdf, 2004.

[Mah01] Qusay H. Mahmoud, Sun Microsystems: http://http://java.sun.com/developer/technicalArticles/ALT/sockets/, 2001.

[Mar02] Jim Martin, *Personal communications. griner and schmitz, kansas city, mo*, June 2002.

[Mic04] Olivier Michel, *Cyberbotics ltd - webotstm: Professional mobile robot simulation*, International Journal of Advanced Robotic Systems **1** (2004), no. 9, 39–42.

[MSC04a] MSC Software, *Msc.adams*, URL: http://www.mscsoftware.com/ products/products detail.cfm?PI=413, 2004.

[MSC04b] MSC Software, *Msc.visualnastran desktop tutorial guide*, URL: http://www.mscsoftware.com, 2004.

[Nie04] Douglas Niehaus, *Kurt-linux: Kansas university real-time linux*, URL: http://www.ittc.ku.edu/kurt, 2004.

[Nom99a] Nomadic Technologies Inc., *Nomadic scout user's manual*, URL: http://nomadic.sourceforge.net/production/scout/scout_user-1.3.pdf, 1999.

[Nom99b] Nomadic Technologies Inc., *Nomadic scout user's manual*, URL: http://nomadic.sourceforge.net/production/scout/scout_langman-1.3.pdf, 1999.

[Pre04] Precise Navigation Inc, *Tcm2-50 feature sheet*, URL: ftp://www.pnicorp.com/technical-information/pdf/TCM2-50productsheet.pdf, 2004.

[Rec04] Recreative Industries, *Buffalo all terrain truck*, URL: http://www.maxatvs.com/buffalo-interior.htm, 2004.

[Ril04] Gary Riley, *Clips: A tool for building expert systems*, URL: http://www.ghg.net/clips/CLIPS.html, 2004.

[SAHA04] Richard S. Stansbury, Eric L. Akers, Hans P. Harmon, and Arvin Agah, *Survivability, mobility, and functionality*, International Journal of Control, Automation, and Systems **2** (2004), no. 3, 334–353.

[Sel03] Denish Selvarajan, *Implementation of real-time java using kurt*, Master's thesis, University of Kansas, Lawrence, KS, 2003.

[SHP03] P.S. Schenker, T. L. Huntsberger, P. Pirjanian, E. T. Baumgartner, and E. Tunstel, *Planetary rover developments supporting mars exploration, sample return and future human-robotic colonization*, Autonomous Robots **14** (2003), 103–126.

[SIC98] SICK, *Laser management systems: Technical description*, SICK AG, Germany, 1998.

[SIC04] SICK, *Collision prevention: Active area monitoring with lms laser scanner*, URL: http://www.sick.dk/lasscan/picolli.pdf, 2004.

[Son04] Sony, *Using your sony viao picturebook computer*, URL: http://www.docs.sony.com/release/PCGC1MV.PDF, 2004.

[SS96] Leon Sterling and Ehud Shapiro, *The art of prolog*, 2nd ed., MIT Press, Cambridge, MA, 1996.

[Sun04a] Sun Microsystems, Inc., *Java remote method invocation*, URL: http://java.sun.com/products/jdk/rmi/, 2004.

[Sun04b] Sun Microsystems, Inc., *Java technology*, URL: http://java.sun.com, 2004.

[TOHC01] A. Trebi-Ollennu, T. Huntsberger, Yang Cheng, E.T. Baumgartner, B. Kennedy, and P. Schenker, *Design and analysis of a sun sensor for planetary rover absolute heading detection*, IEEE Transactions on Robotics and Automation **17** (2001), no. 6, 939–947.

[VBSH00] R. Volpe, E. Baumgartner, P. Schenker, and S. Hayati, *Technology development and testing for enhanced mars rover sample return operations*, Proceedings of the 2000 IEEE Aerospace Conference (Big Sky, MT), March 2000, pp. 247 – 257.

[VGH03] Richard T. Vaughan, Brian P. Gerkey, and Andrew Howard, *On device abstractions for portable, reusable robot code*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Las Vegas, Nevada), October 2003, pp. 2421–2427.

[VGH06] Richard T. Vaughan, Brian P. Gerkey, and Andrew Howard, *Reusable Robot Software and the Player/Stage Project*, In Brugali D. (Ed.) Software Engineering for Experimental Robotics, Springer STAR series, 2006.

[BCW02] A. Behar, F. Carsey, and B. Wilcox, *Polar Traverse Rover Development for Mars, Europa, and Earth*, Proceedings of the IEEE Aerospace Conference (Big Sky, MT), MArch 2002, pp. 389–396.