

Autonomous Mobility Software Validation Challenges for Planetary Surface Missions

Edward Tunstel
Robotic Vehicles Group
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA, 91109
tunstel@robotics.jpl.nasa.gov

Abstract

This paper discusses characteristics of the surface mobility problem that present challenges for validation of autonomous mobility software. We provide a general description of a commonly used hardware-centric and dynamic approach to mobility software validation for flight systems. We also highlight some effects and impacts of flight schedules and deliveries on this validation approach and related mobility software development. Finally, we offer suggestions for dealing with some of the challenges and improving the process for autonomous mobility software development and validation on future projects.

1. Introduction

NASA employs autonomous rovers as surrogate explorers on remote planetary surfaces. The utility of autonomous rovers is a function of their ability to move about and explore intelligently without frequent contact with Earth-based mission operators. More increasingly, robotic vehicle autonomy is required to achieve aspects of overall success for planetary surface missions such as the 2003 Mars Exploration Rovers (MER) and 2009 Mars Science Laboratory (MSL) flight projects. As such, the software that enables autonomous mobility must be validated against related functionality required for mission success. The process of validation establishes that the software system design is actually capable of executing required mission functions.

Verification of isolated mobility software functions against specific requirements is relatively straightforward. However, validation of autonomous mobility functionality is non-trivial due to the fact that robotic surface mobility systems interact non-deterministically

with the physical environment. This presents a number of challenges when it comes to establishing the capability of autonomy software for executing required mission functions involving surface mobility and navigation in unstructured and/or uncharted terrain.

A variety of effective approaches exist for validating software using formal and informal methods depending on the application. Autonomous mobility software validation methods often include high-fidelity simulation and extensive physical testing as part of functional (“black-box”) testing strategies for which a number of methods apply [1]. While the key words here are “high-fidelity” and “extensive,” flight schedule and resource realities may only afford us one, the other, or neither as the case may be. Indeed, in our rover surface mission experience thus far, we have seen that flight systems do not always have the luxury of development plans and schedules that permit extensive testing of mobility systems in realistic environments [2-4]. Even with these luxuries, software validation in general [1, 5] and autonomy software in particular [6] remains a developing art. Nevertheless, flight system personnel and project management must be convinced that autonomy software will satisfy related mission requirements at all levels.

In the sections that follow, we discuss autonomous mobility and its software validation for surface missions. In section 2, we relate mobility functionality to mission success, followed by characteristics of surface mobility that present challenges for validation in Section 3. Section 4 provides a general description of an approach used to validate mobility software for Mars rover missions. In Section 5, some realities of flight system implementations are highlighted as well as effects and impacts on autonomous mobility software development and validation. Brief recommendations are offered in Section 6 for dealing with some of the challenges and for

improving the validation process for future projects, followed by a summary and conclusions.

2. Autonomous mobility and mission success

Robotic vehicles for planetary surface missions are designed to effectively maneuver in a complex target environment and extend the reach of onboard science instruments beyond that of stationary landers. Whether the target environment is the planet surface or subsurface, mission success is in some way enabled by the basic mobility functionality. Autonomy is required to enhance surface missions by improving the effectiveness of the mobility function as a science-driven tool for achieving mission goals [7]. Accordingly, recent and planned surface missions include requirements that rely on autonomous mobility to achieve mission success.

In 1997, the Mars Pathfinder Mission conducted a Microrover Flight Experiment (MFEX) using the Sojourner rover. The rover software design included a capability for low-level reactive navigation for the purpose of safely reaching targets of scientific interest in the area surrounding the lander. The rover was required to traverse to a target of opportunity to acquire spectroscopic measurements as part of mission success [3]. This could be accomplished by sequencing basic mobility commands or by using the reactive navigation functionality, and so this requirement was only loosely tied to autonomy from a validation standpoint. As the need for autonomous mobility increases it is reflected more explicitly in mission requirements against which software must be validated.

The twin rovers launched by the MER project and the rover being planned for the MSL mission are explicitly required to use autonomy in support of mission success. Software designs for these rovers include autonomy capabilities of varying complexity for navigation, instrument placement, resource management, and science data gathering. General statements of the related autonomous mobility requirements are similar to the following: the rover(s) must be able to safely traverse some substantial minimum distance per day of operations in terrain of some reference complexity while maintaining estimated position knowledge within some small percentage of distance traversed [4, 8]. The reference complexity of the terrain is typically as documented by images taken at Viking Lander (1970s) or Mars Pathfinder (1997) landing sites. Following a successful MER mission, the MSL terrain complexity may be expressed in terms of terrain traversed by MER rovers. Furthermore, the mobility requirements for MSL will likely call for increased autonomy to achieve primary mission capabilities of longer distance traverses and autonomous short distance approach to designated science targets followed by instrument placement (all

with less interaction with mission operators than prior missions) [9].

In this paper, we limit the scope of our discussion to software validation issues related to autonomous mobility only. For the purpose of the discussion, we refer to autonomy software for surface mobility as that onboard software designed for data processing associated with sensing, perception, reasoning and decision-making for the purpose of directing servo-level execution. Therefore, we are addressing software at a higher level than servomotor control and at a lower level (in most cases) than symbolic planner-schedulers. Within this context, high-level autonomy includes perception, reasoning, decision-making, etc., while low-level autonomy includes sensor-based reactive motion, hazard detection and avoidance, and local navigation. High-level autonomous mobility software is often designed to interface with low-level autonomy software and, as such, is at least one architectural level removed from interactions with mobility hardware and its environment interactions. For this reason, high-level autonomy software evaluation can more readily be handled using verification methods (e.g. model checking and model-based reasoning [10]).

Low-level autonomous mobility software responds to perceptions and recommendations derived from high-level software and the sensor-based perception system. It can be verified to a limited extent using similar methods. However, the validation challenges increase considerably as we move deeper into the autonomous mobility system from software-software interactions to software-hardware interactions particularly due to transitions from determinism to non-determinism in software-induced system behavior. The non-deterministic system behavior manifests itself at the level of hardware-environment interactions, where the "rubber hits the road." It is here where it becomes more difficult to establish that the system is actually capable of executing the mobility functions required for mission success.

3. Non-determinism as "rubber hits road"

Modeling, simulating and/or predicting the functional behavior of orbiter and fly-by spacecraft is facilitated by reasonably well behaved dynamics and operating environments. For such robotic spacecraft, conventional estimation and control techniques [11] have similar effects on spacecraft behavior in simulation as they do in reality. This is due to the fact that the physical laws of orbital mechanics and planetary atmospheric aerodynamics are reasonably well understood and well behaved in space. Unfortunately, the complexities of interaction between mobility systems and planet surfaces dominate when the "rubber hits the road," and the problems are sometimes compounded by reduced-gravity effects. The result is non-deterministic behavior as the

system interacts with the world and increased uncertainty in how the autonomous mobility system will respond to operational commands. A few characteristics of the surface mobility problem that serve to illustrate this are described below.

Mobility and navigation problems for outdoor rough terrain vehicles are characterized by high levels of difficulty and increased measurement uncertainty. This is due to the fact that complex motions outside of the ground plane occur quite frequently as the vehicle traverses undulated terrain, encountering multidirectional impulsive and resistive forces throughout. In addition, common mobility and navigation sensors often inadequately handle the tremendous variability of surface features and properties of outdoor terrain. Such sources of uncertainty in input interpretation and output execution reduce the predictability of system behavior.

Wheeled mobility systems are also subject to undesirable wheel-terrain interactions that cause wheels to slip on rocks and soil. Frequent loss of traction due to wheel slip during traverses from one place to another will detract significantly from the ability to maintain good rover position estimates. These factors impact the ability to guarantee required accuracy of localization estimates.

In soft soils, loss of traction due to excessive wheel slippage can lead to wheel sinkage and ultimately vehicle entrapment. It is possible for wheels to sink to soil depths sufficient to prohibit rover progress over terrain, thus trapping the vehicle at one location. This is also possible on soils with insufficient bearing strength to support the rover (incidentally, a property to which a look-ahead perception system may be insensitive). Such factors potentially impact our ability to guarantee compliance with traverse safety and/or distance requirements.

How do we convince ourselves, then, that autonomous mobility software that induces non-deterministic behavior will perform well enough to execute mission functions as required? We respond to this challenge by conducting a validation testing program that aims to bind the relevant uncertainties to limits within which mobility requirements can be met with high probability. This requires extensive functional testing and system characterization. Thus, our approach is based on the notion that given sufficient testing, it is possible to make reasonably comfortable predictions about the software capabilities [1].

4. Autonomous mobility software validation

Flight system validation activities place emphasis on validating the implemented system design and include challenging the system to establish its operational limits through functional testing in nominal and off-nominal scenarios. Here, we briefly describe in general terms an approach used to validate autonomous mobility flight software to a level of acceptable confidence.

We are still gaining valuable experience in rover surface missions building upon the first success in 1997 with Sojourner, the MFEX rover. As such, validation methods are inherited to some extent from the technology development practices that led to Sojourner and continue today. These methods generally rely upon the availability of one of more prototypes of a flight rover referred to as a Software Development Model (SDM), followed by later availability of an Engineering Model (EM) rover that is very similar if not identical to the flight article.

4.1 Reliance on physical rover models

Depending on project resources and/or the maturity of the rover design early in project development, an SDM may also be an EM for all intents and purposes. More often than not, SDMs vary in hardware and software fidelity with respect to the flight rover from project to project. At a minimum, the SDM bears some similarity to aspects of the flight system under development (physical configuration, subsystem functionality, etc) but does not approach the fidelity of the EM. The higher the fidelity with respect to the flight article, the more value-added to the validation process by these prototypes. Availability of an SDM for use by rover flight software developers is preferred, recommended, and most valuable early in the project definition or development phases.

An SDM rover is typically used in a preliminary test arena such as an indoor sandbox collocated with the software development laboratory. This physical testbed is equipped with a variety of reconfigurable terrain artifacts (and ideally, lighting options) that permit arrangements of realistic landscapes for mobility testing. Additional essentials include measurement systems for ground truth, data logging, and other means for test related documentation and post-analysis. Finally, this infrastructure is augmented by more realistic outdoor facilities that resemble the planetary terrain as closely as is practical given project resources. These facilities are used for realistic field trials to validate autonomy algorithms [12] and, eventually, operational readiness tests to validate required functionality. Field tests may include end-to-end operations using appropriate facilities and infrastructure including satellite communications between JPL and remote field sites [13]. The general validation approach for autonomous mobility software is hardware-focused and utilizes validation metrics such as requirements coverage, which ensure that all required functionalities are covered by at least one test [5].

4.2 Validation of Mars rover autonomy

The steps taken to formulate and create the methodologies and experimental/testing facilities used during Sojourner's software development [2] were

important steps toward enabling systematic performance evaluation and validation for later Mars rover prototypes as well. Evaluation of Sojourner rover autonomous mobility software for the MFEX mission activities consisted of many navigation trials in an indoor sandbox and outdoor trials in realistic terrain using SDMs. Simulation runs were also used as an alternative to laborious test setups (manual arrangements of rock distributions) that offered an automated means of achieving more complete coverage of software scenarios in lesser time than physical tests [2]. The simulated rover runs were used to validate simulation predictions via comparison to real runs. Simulation was shown to accurately predict rover SDM behavior in a statistically significant manner for runs without failures [14].

The MER autonomous mobility software validation uses similar methods performed in stages on an SDM and later on flight EMs. Early testing and validation of the autonomous navigation software was done using the Athena SDM rover running navigation trials in the JPL MarsYard [8], an outdoor test facility. The Athena SDM differs in size and kinematic configuration from the MER design but utilizes functionally similar mobility and hazard detection hardware. Later validation was done in an indoor sandbox facility using the flight EM rovers by running a number of test cases under different terrain conditions to validate nominal and off-nominal functionality. Further validation plans include robustness and characterization testing in outdoor environments of increased variability. Such testing permits refinement of the many tunable parameter values that are characteristic of autonomy software and govern its performance.

The validation program being contemplated for the MSL autonomous mobility software includes some of the same elements described thus far. The key autonomy technologies required for MSL mobility are presently under development and will undergo a validation process formulated by the Mars Technology Program [9]. Through this validation process, quantification of software and algorithm performance will be based on field experimentation as well as statistical results from simulations. Newly developed mobility (and other) autonomy algorithms will be validated prior to actual consideration by, and infusion into, the flight project. Thus, it represents a generalized validation process aimed at validating different new technologies on different robotic platforms (SDMs essentially) and in various conditions. Any specific autonomy software validated by this process and selected for use by MSL will still need to be validated on an MSL EM rover. Scenarios would be derived from autonomy requirements for long distance traverse and short distance approach to science targets.

5. Validation challenges and flight projects

Thus far we have discussed the fundamental challenge presented by non-determinism and a general validation approach used to deal with it, which is focused on hardware-based functional testing and computer simulations. Unfortunately, the challenges do not end there. In this section, we highlight some of the challenges that may be encountered when attempting to apply the validation approach within the flight project environment in the face of project implementation realities.

5.1 Integrated schedule issues

Flight systems are comprised of numerous subsystems whose respective schedules are in relative flux throughout the design and development phase. As projects go, subsystems rely on and are committed to other subsystems through a collection of receivables and deliverables of documents, hardware and software. Inevitable misalignments in the project integrated schedule must be continually adjusted within the finite time and resources allocated for project implementation. From the vantage point of software development, some of the possible ramifications include slips in schedule that result in late hardware deliveries (SDM or EM), rushed hardware deliveries that require re-work before becoming useful for software testing, and tight software development and release schedules that serve to reduce time allocated for validation testing on hardware.

As emphasized earlier, surface mission projects are best served when autonomous mobility software developers are provided with hardware early. In our rover surface mission experience thus far, we have seen that flight systems do not always have the luxury of development schedules that permit sufficient time for extensive testing of EM rovers. Late hardware or the lack thereof coupled with insufficient time for thorough testing goes against the aim of our validation approach to bind performance uncertainties to limits within which mobility requirements can be met with high probability. This creates a situation wherein we risk falling short of meeting the challenges posed by non-determinism, and thus, our ability to reach conclusions about software capabilities and validated requirements.

5.2 Facility issues

In addition to integrated schedule issues, flight systems do not always have sufficient facilities or resources to conduct extensive testing of rovers in realistic (physically similar to destination) environments. As the requirements for autonomous mobility become more complex relative to the Mars Pathfinder MFEX, the importance of the physical test environment fidelity increases. The bulk of mobility and navigation software validation cannot continue to be done with high

confidence based solely on SDM/EM exercises in indoor sandbox facilities or computer simulations. Sandboxes are adequate for early incremental development and isolated testing of functionality and performance. However, the richer test environment offered by planetary analogue natural terrain is essential for characterization and exposure of software design problems that may not arise in the sandbox.

An interesting dichotomy occurs in the transition from SDM to EM rovers with respect to testing and later validation. When the time comes at which a flight-like EM is available for use by autonomous mobility software developers, test activities migrate to these higher fidelity platforms. Due to the usual high expense and criticality of EM and flight hardware there is a general (and justified) conservatism associated with the handling of the hardware. Developers and test conductors are reluctant to risk breaking the hardware and, to some extent, degrading its pristine condition through the course of aggressive testing on rough terrain. This is contrary to the type of testing needed to validate autonomous mobility software. It serves to steer testing further away from analogue environments where the validation testing would be more meaningful, to more benign and less risky settings in the sandbox. This is an important issue because the flight rover(s) proper may never be extensively tested on analogue terrain until reaching the planetary destination when the surface mission begins. The subset of critical functionality is certainly validated on the flight rover(s), but this occurs in the confines of clean project laboratories where the terrain consists of protective floor-mats, and where critical metrics such as actuator usage must be carefully monitored relative actuator lifetime. Meanwhile, the aspects of validation discussed herein must be done on EMs in "dirty" testbed environments.

Since MFEX, we have started to outgrow existing rough terrain facilities such as the various JPL sandboxes and the current JPL MarsYard. With MER and MSL requirements on traverse distance and related accuracies larger facilities are needed to support the validation effort. Furthermore, terrain facilities of greater variability are needed since the sandboxes and the MarsYard are essentially flat and devoid of continuous courses of sufficient length and variable terrain types to adequately support long distance traverse trials. This is no substitute for the advantages to be gained from remote field trials, however, when practical.

Computer simulation facilities are considered by many to be an attractive validation option in several situations: (1) in lieu of available rover hardware, (2) to improve test case coverage when there is insufficient time or resources for extensive hardware tests, (3) when there is a desire to avoid aggressive tests with EM rovers, and (4) when logistics of remote outdoor testing in analogue terrain are impractical. At present, however, we cannot

model physical environment interactions associated with vehicle motion (i.e., wheel-terrain effects, soil mechanics, vehicle dynamics, etc.) well enough to rely on simulations alone for validating autonomous mobility software. A balance must be struck between functional hardware-based testing and high-fidelity simulation [15] that achieves the aim of the validation process.

5.3 Impact on autonomy performance and usage

In worst-case scenarios, flight development schedule and facility issues discussed above can combine to significantly impact flight software development and performance, as well as the degree to which validation is achieved and autonomy capabilities are actually exercised during a mission. A potential domino effect can occur that comes full circle to impact autonomy technology development and infusion for future missions.

Delayed hardware deliveries lead to deferment or descopes of characterization testing, which we have declared as an essential activity through which software developers establish fine-tuned sets of parameters that govern autonomy performance. While Monte Carlo computer simulations of reasonable fidelity add definite value, characterization of physical mobility systems in physical environments directly feeds back into autonomy software development and enables more meaningful validation. This test and development feedback loop is a "software-rover system identification" process of sorts, which robot software developers are quite familiar with in common practice. It may be thought of as a robotics "mind-body" exercise that can easily produce different sets of configuration parameter values for seemingly identical hardware platforms (e.g., the twin MER rovers and their EMs). Characterization testing could effectively be performed during the post-launch, pre-landing periods on EM rovers; however, at such a late stage such testing could easily turn into increased regression testing.

Schedule slips and inadequate facilities lead to thorough validation of only the most basic functionality. In this case, the autonomy capabilities are less likely to be used by conservative mission operations personnel if they are deemed as not absolutely necessary for achieving minimum mission success. The fact that the minimally tested autonomy remained part of the mission baseline capability is testament to the belief that it would have enhanced the mission and perhaps the science return. If it is not sufficiently validated, then it is not sufficiently trusted, and the development investment and enhanced science return is lost for the mission. At best, we are doomed to resume the validation process for a future mission should a similar capability be required.

In the event that autonomy software is sufficiently validated in the eyes of flight system engineers, perhaps the remaining challenge is presented by a conservative

mission system that is reluctant to use the capability until late in the mission operations. This potential to incompletely exercise the full autonomy capability of the software can result in misleading post-mission assessments about the level of autonomy included among mission capabilities (or worst, reduced science return). In this case, there is no distinction between autonomy functions that are available versus those that were actually exercised during the mission. This fosters a misperception of the actual state of the art of flight autonomy, which influences the focus of future technology development and investments. The final impact is that technology development funds may be allocated to new starts that reinvent the wheel, when all that may be necessary is evaluation and further validation of existing autonomy flight software.

6. Recommendations

As NASA continues to design, develop, and operate missions requiring autonomy for surface mobility, we will gain valuable experience in determining the validation methods that work best for the domain. Indeed, verification and validation of autonomy software systems is a developing art as witnessed by the frequent organization of recent workshops dedicated to the topic [6, 16, 17]. In the meantime, there are a number of steps we can take to improve upon the current approaches in light of the challenges discussed above.

To deal with some of the challenges of validating autonomous mobility software, the validation process must include substantial testing in more realistic environments. This will require investments in facilities and supporting infrastructure. In particular, new rough terrain facilities should be contemplated of size and features commensurate with distance goals and terrain variations expected for future missions. Discussions among interested managers and program offices have already begun towards this end. Representative examples of large-scale outdoor test courses for ground vehicle testing exist and are being used in DARPA and DOE funded research programs. They include a 100-acre testbed at Southwest Research Institute [18] and the 226-acre Robotic Vehicle Range [19] at Sandia National Laboratories. These facilities are several orders of magnitude larger than the JPL MarsYard.

Flight projects should plan to build enough SDM/EM platforms to ensure that the mobility software development team has a dedicated platform to work with throughout project development and validation activities. This would alleviate contention for testbed hardware usage among subsystems. It would also allow maximal time for software-rover system identification (as described in Section 5.3 above). Investment in high-fidelity simulation capabilities should continue so that

they are available to supplement hardware testing, especially if an SDM/EM is not available in the early software development phase. The Mars Technology Program is making strides in this direction [9]. However, to further enhance validation capabilities, new and emerging computing methods should be adopted that can be augmented with simulation capabilities to compensate for any lack of realistic facilities and/or validation time in the project schedule. A good example of this is a means for automated testing and test case generation with high-fidelity simulation in the loop. Some progress in this regard has been made at the Naval Research Laboratory where an approach based on genetic algorithms has been developed to automate an emulation of the process that test engineers follow to search for interesting fault scenarios in a space of possible fault scenarios [20]. Such a search is otherwise done sparsely and more slowly by the manual process used by test engineers. Experience using this approach suggests that it offers advantages over other automated and manual testing methods used for validating intelligent software controllers [20].

Finally (for now), efforts should be made to define and utilize autonomy performance metrics to facilitate mobility software validation. Good performance metrics for autonomy will allow us to describe autonomy capabilities in quantitative terms. This would be directly useful for convincing project management (and the conservative mission operator) not only that a capability exists, but also how good it is with respect to some well-defined and easily understood metric of performance. Performance metrics for autonomy will also enable us to write better requirement statements that are falsifiable, which is a pre-requisite for requirement validation [1]. Good examples of metrics that are germane to autonomous systems are not common knowledge but have been a topic of considerable thought for the past several years. A notable white paper sums up the state of such thought as evolved from an annual workshop on the topic sponsored initially by NIST and most recently with co-sponsorship including NASA [15].

7. Summary and conclusions

In the preceding sections, we emphasize the increasingly stronger relationship between autonomous mobility and surface mission success criteria. We discuss non-deterministic aspects of the surface mobility problem that present challenges for autonomy software validation. A validation approach being used to meet the challenges is generally described and some effects of flight schedules and deliveries on successful implementation of the approach are highlighted. Finally, recommendations are offered for dealing with the challenges presented and include: expansion of test and validation facilities, early hardware delivery to software developers, automated

simulation-based testing, and adoption of autonomy performance metrics.

We acknowledge that functional testing will not expose all problems with autonomous mobility software. It should be used in addition to formal verification where applicable. We advocate the notion that extensive testing in realistic settings will enable characterization, which will enable software tuning and improve predictability, as well as our ability to diagnose problems that arise during operations. Extensive testing will increase the likelihood that the full autonomy capability being validated will be exploited on the mission and lead to healthier returns on the investments made to develop the software capability. Furthermore, this will permit efficient reuse, and accurate assessments of the state of the art, of autonomy software technology for future missions.

8. Acknowledgments

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

9. References

[1] Beizer, B., *Black-Box Testing: Techniques for functional testing of software and systems*, John Wiley & Sons, New York, 1995.

[2] L. Matthies, E. Gat, R. Harrison, B. Wilcox, R. Volpe, and T. Litwin, "Mars Microrover Navigation: Performance evaluation and enhancement." *Autonomous Robots*, Vol. 2, No. 4, Kluwer Academic Publishers, Dordrecht, Netherlands, 1995, pp. 291-312.

[3] Mishkin, A., et al., "Experiences with Operations and Autonomy of the Mars Pathfinder Microrover," Proc. IEEE Aerospace Conference, Aspen, CO, March 1998.

[4] J. Biesiadecki, M. Maimone, and J. Morrison, "The Athena SDM Rover: A testbed for Mars rover mobility," *International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Montreal, Canada, Paper No. AM026, June 2001.

[5] Rakitin, S.R., *Software Verification and Validation: A practitioner's guide*, Artech House, Boston, MA, 1997.

[6] C. Pecheur, J. Caldwell, R. Simmons, and W. Visser, "Verification and Validation of Autonomous and Adaptive Systems," Report from the RIACS Workshop on the Verification and Validation of Autonomous and Adaptive Systems, Pacific Grove, CA, December 2000, Online Version 2.2, Feb. 23, 2001, <http://ase.arc.nasa.gov/vv2000/asilomar-report.html>.

[7] Space Studies Board & National Research Council, "A Scientific Rationale for Mobility in Planetary Environments," National Academy Press, Washington, DC, 1999.

[8] S.B. Goldberg, M.W. Maimone, and L. Matthies, "Stereo Vision and Rover navigation Software for Planetary Exploration," *Proc. IEEE Aerospace Conference*, Big Sky, MT, March 2002

[9] R. Volpe, "Rover Functional Autonomy Development for the Mars Mobile Science Laboratory," *IEEE Aerospace Conference*, Big Sky, Montana, March 2003, Paper #1289.

[10] C. Pecheur, "Verification and Validation of Autonomy Software at NASA," NASA Technical Report, NASA/TM 2000-209602, August 2000.

[11] Kaplan, M.H., *Modern Spacecraft Dynamics and Control*, John Wiley & Sons, New York, 1976.

[12] T. Huntsberger et al, "Rover Autonomy for Long Range Navigation and Science Data Acquisition on Planetary Surfaces," *IEEE International Conference on Robotics and Automation*, Washington, DC, May 2002, pp. 3161-3168.

[13] E. Tunstel et al, "FIDO Rover Field Trials as Rehearsal for the 2003 Mars Exploration Rover Mission," *Proc. 9th International Symposium on Robotics and Applications, World Automation Congress*, Orlando, FL, June 2002.

[14] E. Gat, "Towards Principled Experimental Study of Autonomous Mobile Robots," *Autonomous Robots*, Vol. 2, 1995, pp. 179-189.

[15] A. Meystel et al, "Measuring Performance of Systems with Autonomy: Metrics for intelligence of constructed systems," White Paper, in *Proc. of Performance Metrics for Intelligent Systems Workshop*, NIST, Gaithersburg, MD, August 2000.

[16] Software Assurance Research Program, NASA Software Independent Verification and Validation Facility, *2nd Annual NASA Office of Safety and Mission Assurance Software Assurance Symposium*, Berkley Springs, West Virginia, September 2002, <http://sas.ivv.nasa.gov/>.

[17] A. Christiansen, R. Harrigan, and K. Kwok (Workshop Orgs.), "Validation of Public Sector Robotic Systems: Moving from demos to experiments," *IEEE International Conference on Robotics and Automation*, Washington, DC, 2002.

[18] B. McBride and G. Peri, "Testing Ground Mobile Robots," in A. Christiansen, R. Harrigan, and K. Kwok (Orgs.), *Workshop on Validation of Public Sector Robotic Systems: Moving from demos to experiments. IEEE Intl. Conference on Robotics and Automation*, Washington, DC, 2002, http://tmr.appliedphysics.swri.edu/icra/mcbride_and_peri.pdf.

[19] Sandia National Laboratories, "Robotic Vehicle Range," <http://www.sandia.gov/isrc/RVR.html>, Last updated: 10/28/02.

[20] A.C. Schultz, J.J. Grefenstette, and K.A. De Jong, "Learning to Break Things: Adaptive testing of intelligent controllers," in *Handbook of Evolutionary Computation*, IOP Publishing Ltd and Oxford University Press, 1995.