Characterizing Performance Guarantees for Multiagent, Real-Time Systems Operating in Noisy and Uncertain Environments

Damian Lyons
Computer & Information Science
Fordham University
Bronx, NY 10458
dlyons@cis.fordham.edu

Ronald Arkin
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA 30332
arkin@cc.gatech.edu

Stephen Fox
Computer & Information Science
Fordham University
Bronx, NY 10458
stfox88@gmail.com

Shu Jiang
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA 30332
sjiang@gatech.edu

Prem Nirmal
Computer & Information Science
Fordham University
Bronx, NY 10458
prem.nirmal88@gmail.com

Munzir Zafar
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA 30332
mzafar7@gatech.edu

ABSTRACT

Autonomous robots offer the potential to conduct Counter-Weapons of Mass Destruction (C-WMD) missions in an efficient and robust manner. However, to leverage this potential, a mission designer needs to be able to determine how well a robot system will operate in the noisy and uncertain environments that a C-WMD mission may require. We are developing a software framework for verification of performance guarantees for C-WMD missions based on the *MissionLab* software system and a novel process algebra approach to representing robot programs and operating environments.

In this paper, we report on our initial research for the Defense Threat Reduction Agency (DTRA) in understanding what is required from a performance guarantee to give a mission designer the information necessary to understand how well a robot program will perform in a specific environment. We link this to prior work on metrics for robot performance. Using a simple mission scenario, we explore the implications of uncertainty in the four components of the problem: the robot program, and the sensors, actuators and environment with which the program is executed.

Categories and Subject Descriptors

I.2.9 Robotics; D.2.4 Software/Program Verification; D.2.6 Programming Environments

General Terms

Performance, Languages, Verification, Robotics.

Keywords

Performance guarantees, probabilistic and emergent robotic systems.

1. INTRODUCTION

To effectively deploy an autonomous robot or robot team to search and locate weapons of mass destruction, it is important to have performance specifications and guarantees available for the equipment. Because of the severe potential downside in these mission-critical operations, the robot and its software must have the best chance of succeeding given the environmental conditions and other constraints in which it must operate. However, this environment may be uncertain, and the software that operates the robot or robot team may be probabilistic [20], emergent [1], and/or multiagent [3]. Although tremendous strides have been made in software verification (e.g., [9]), this high-impact problem remains extremely challenging.

An important component of the solution is to understand what performance guarantees are useful and possible for Counter-Weapons of Mass Destruction (C-WMD) missions. In this paper we present an overview of the system, which is based on the *MissionLab*[1] mission specification system [17], being developed for integrating the generation and use of performance guarantees as an iterative step in the design of robot software for C-WMD missions. Using examples in this design framework, we analyze what mission performance guarantees are of value to a mission designer from the perspectives of understanding how well the system will function and of understanding how to improve its performance.

In the next section, we review related work in the area of automatic verification of system performance, and in the development of performance measurements and guarantees. Section 3 reviews a selection of performance measurements. In Section 4 we introduce a simplified example scenario to help understand how uncertainty in sensor, actuator and environment models influences the form of the performance guarantee, making it quite different from the form of liveness and safety guarantees typically seen in software verification. Section 5 then introduces the architecture we have developed to integrate verification into the *MissionLab* software system.

2. RELATED WORK

The field of formal specification and verification of software systems (e.g., Hinchey et al. [7], Clark et al. [4]) has made impressive progress. However, leveraging these results to validate software for mobile robot systems has raised challenges. Probabilistic [20] and behavior-based mobile robotics [1] employ assumptions quite different from those used more generally in the

[1] *MissionLab* is freely available for research and educational purposes at: http://www.cc.gatech.edu/ai/robot-lab/research/*MissionLab*/.

formal analysis of software. One key example is a reliance on emergent behavior: even simple behavior-based systems exhibit complex behavior when acting in a complex environment. This means that formal analysis must include the control program and models of the sensory and motor apparatus as well as environment models.

Discrete-Event Control techniques (e.g., Ramadge [19], Kosecka [10]) have been applied to this problem. Most use Finite State Automata (FSA) as a modeling tool. However, FSA models can suffer from state-space explosion when used to model the kind of realistic search environments that occur in C-WMD. While prior work addresses issues of noisy and uncertain applications, it does so for problems at a relatively low sensorimotor level as compared to for example, algorithms from data mining, artificial intelligence, machine learning and complex adaptive systems theory. Also, work in this area is focused on automatically producing a control strategy or controller, whereas our focus is on verifying software produced by some other means (in our case, generated by a human operator using *MissionLab*). More recently the discrete-event and hybrid approach has been extended to robot path planning and motion control (e.g., Kress-Gazit [11]) with the idea that a human provides a high-level, rich constraint description in linear or interval temporal logic, and a controller is automatically synthesized for these constraints. However, the input constraint or constraints in these systems are quite complex and themselves may now need verification.

The metrics for the performance measurement and guarantees of behavior-based and probabilistic software systems have not been standardized so far, although considerable work is proceeding in the characterization of performance metrics for robot performance [8]. This is the case not only with behavior-based systems but with a broader category of systems that are required to carry out specific tasks intelligently by interacting with real world environments. Serious effort is underway towards standardization of these metrics [16] but the challenges are many. Behavior-based system requirements need to cover a wide spectrum of behaviors ranging from simple tasks such as point-to-point locomotion to relatively complex tasks such as human-robot interaction. The expectations are growing regarding reliable and predictable performance as new possibilities in design are being explored and milestones are being achieved.

Urban search and rescue (USAR) is a domain that is being heavily studied in this context. There are two groups of performance metrics for the characterization of USAR systems that can be broadly classified as system characterization and behavior characterization. System characterization seeks accurate specification of specific robot capabilities to facilitate direct comparisons of different robotic platforms, and particular configurations of similar robot models. The National Institute of Standards and Technology (NIST) has taken a leadership role for defining performance standards for USAR robots [8]. These standards are categorized as human-robot interaction, system, safety, mobility, etc., along with documentation for standard reproducible test procedures. For our purposes, these system metrics will primarily serve as specifications of particular capabilities of the robot with the view of providing a guarantee to the user regarding the ranges of behaviors the system provides, before it is deployed in the real world in the context of a C-WMD mission. Behavior characterization deals with the problem of predicting performance guarantees for high-level tasks to be carried out in uncertain, unstructured, and potentially hostile environments such as navigation, localization and mapping, room search, etc. Some related research exists in performance characterization of higher-level algorithms, i.e., [18] [5], that is intended for the comparison of different algorithmic performance. This comparison would traditionally be done by demonstration (empirical evaluation) instead of formal analysis. Such metrics, however, may prove to be useful as they may improve the expressiveness with which the operator can specify required performance.

3. PERFORMANCE CRITERIA

An important requirement for any evaluation is the establishment of the performance criteria which will serve as the basis for specification and evaluation of the system in question. A method is needed for defining performance goals which not only accommodates various ranges of capabilities but in our case also comfortably fits into the process algebra framework we use for verification; this framework is based on that described in [12]. The absence of any published standards in this regard as well as the growing needs for the capabilities of C-WMD/USAR systems makes this an important area of investigation.

Due to the complexity associated with many formal methods, the performance of control algorithms designed for robots has traditionally been guaranteed only through empirical evaluation and demonstration on real systems. Many performance criteria have been devised to compare the performance of such algorithms in this context [8]. Those criteria serve as a reference for defining the mission performance criteria for our verification procedures.

Since we are targeting the USAR/C-WMD applications, a good starting point is to identify the most common requirements in this application area. These include navigation, exploration, localization, mapping, search, and victim identification (among other things). We can then refer to the large body of literature available for the performance evaluation of the algorithms designed for these high-level system goals. In navigation, for example, [18] has proposed a set of useful performance metrics along with their formulae and algorithms that could directly be applicable to our framework. These include safety metrics (e.g. mean obstacle distance), dimensional metrics (e.g. trajectory length, time of completion) and smoothness metrics (e.g. bending energy, smoothness of curvature). Similar propositions are made in [5]. Related work is available for other areas of application as well. Currently there is no universal agreement with regards to these metrics, but it is hoped that the availability of common tools and techniques to verify, validate, and formally prove performance guarantees for high-level mission controllers will lead to standardization of such performance characterization.

The metrics discussed above can be accommodated as part of our framework, allowing the user to specify the mission goals and expectations, i.e., specific mission criteria. In the case of multiple metrics/criteria, the user may then choose to investigate whether a mission is likely to experience a catastrophic failure or whether a graceful degradation is more likely. This is a powerful feature of our approach; we are not just interested in binary yes/no answers regarding performance guarantees as might be typical for more traditional software verification. The information that a mission designer or operator needs to decide whether to deploy a robot mechanism for a C-WMD mission includes not only the standard concepts of mission completion ('liveness') and safety, but also information about how likely overall success might be, given the noisy and uncertain environment for the mission.

4. ROBOT SCENARIOS

Performance criteria need to reflect the missions with which robots will be tasked. In this section we look at several example missions and consider how they impact what must go into a performance criterion. In the first example the robot control strategy is deterministic, where the sensor and actuators operate with no noise and where there is no uncertainty in the environment model.

4.1 Deterministic Scenario

A robot searching an area for a target executes actuator commands to move through the search area, deploying its sensors to search for the target.

- The robot program is deterministic.
- If the actuators always carry out the motion commands exactly, then the robot program can always rely on knowing where it is and hence where it has been.
- If the sensors always report the situation in the environment with certainty, then obstacles, other agents and the target can always be reliably detected.
- Finally, if the environment in which the robot operates has no associated uncertainty, then the robot program will always fulfill its mission requirements or it will always fail.

This deterministic scenario does not reflect many actual operating situations; however, it is necessary to include it as a base case. We introduce a very straightforward example of a search task to drive this and the succeeding scenarios. Consider a robot moving from one location A to a second location B repeatedly as shown in Figure 1.

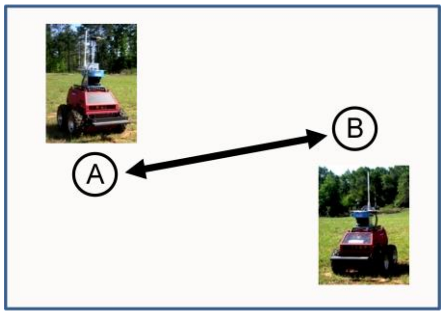


Figure 1: Repeated Traverse Mission

The mission designer is interested in two kinds of guarantees which we can broadly categorize using the traditional Liveness and Safety terms:

1. **Liveness**: Will the robot achieve a mission objective? Examples might include:
    - Will the robot arrive at B? (Note that the complexity of the control strategy or environment model, or the accuracy of the sensors or actuators, may still render this a difficult verification problem.)
    - Will the robot complete *n* traversals from A to B?
    - Will the robot complete *n* traversals from A to B by time *t?*
2. **Safety**: Will the robot be free of error situations while carrying out its mission object? Examples could include:
    - Will the robot avoid any and all obstacles between A and B?
    - Will the robot keep its power consumption within safe levels at all times?
    - Will the robot always read its radiation sensor at a rate of 10Hz or higher.

Because there is no uncertainty in this example scenario, the performance guarantees exhibit a binary nature; the robot program will conform to the performance guarantee or it won't. This is typical of the kind of verification constraints seen in general-purpose software verification.

4.2 Nondeterministic Environment

Consider a modification of the previous example in which the terrain between locations A and B has an element of uncertainty with respect to its traversability. The actuators and sensors remain deterministic in their performance and the search program itself is deterministic.

The environment in which the robot now has to operate is one that can contain patches of terrain that are more difficult to traverse and the robot will make less progress on these patches. Any particular execution of the robot mission will encounter some number of patches and be slowed as a result. Different executions might encounter different numbers of patches, and hence exhibit a range of performance.

This possible range of performance complicates the performance guarantee beyond the binary case we have discussed before. Now consider the liveness condition: Will the robot complete *n* traversals from A to B in time *t?* In the deterministic scenario, the robot would either always or never achieve this. However, in this scenario, there will be some executions in which the robot does achieve this performance and some in which it does not.

4.2.1 Expected performance

If we leverage the probabilistic concept of expected value, then one approach is to ask:

- Is the number of *expected traversals* from location A to location B in time *t* equal to *n?*
- Alternatively we can ask, is the *expected time* for the robot to complete *n* traversals from location A to location B equal to *t*?

Even though the environment is not deterministic, this form of the performance guarantee maintains the easy binary structure of the deterministic case. This increases the realism of the scenario without complicating the way in which the mission designer has to understand performance.

Nonetheless, this approach does hide the variation in performance behind the concept of expected value. That variation may itself be a useful and sometimes necessary tool for the mission designer.

4.2.2 Performance Confidence

In scenarios where the options are limited and the risks are high, a mission designer may consider it reasonable to deploy a robot for a mission even though the reasons to believe the robot will succeed are somewhat slim. Therefore it is also important to make the information about the variability in performance available to the designer in a performance guarantee.

Returning to the traverse example, a designer can reasonably want to know:

- *how likely it is* that the robot will complete *n* traversals from location A to location B in time *t* given the environment in which it has to carry out the mission.

This additional information is purchased at the cost of complicating the performance guarantee to include a probability that needs to be interpreted by the mission designer. A reasonable interpretation might be: For a very large number of executions in this environment, in what percentage of executions does the robot complete *n* traversals from location A to location B in time *t* or less?

4.3 Noisy Sensors and Actuators

Moving another step towards making our initial, deterministic scenario more realistic, let us now consider a situation where the robot sensors and actuators operate with noise. That is, the motion command communicated to the robot by the robot program may not always produce the same effect on the robot, and a sensor reading taken during the identical environmental conditions may yield different measurements. The robot program remains deterministic.

4.3.1 Expected Performance

The consequence of this uncertainty for the repeated traversal mission is that the robot may not always reach the locations A and B, irrespective of terrain traversability. After some number of traversals, the robot may conceivably have drifted far from A and B. A mission designer might ask:

- After *n* traversals from A to B, will the expected location of the robot be within a distance *r* of location B?

This is an application of the expected value concept again, but in this scenario to a spatial objective rather than a temporal one.

4.3.2 Performance Confidence

In the scenarios in which knowledge of the variation in performance is important, a designer may want to ask:

- After *n* traversals from A to B, how likely is the robot to be within a distance *r* of location B, given the environment in which the program is carried out.

This more complex performance criterion can be interpreted as follows: after a large number of different executions of the program in this environment, in what percentage of them was the robot within a distance *r* of the location B.

Even this more complex form of the performance criterion hides information. If the likelihood of being within *r* of location B is a value *p,* then for the remaining *1-p* cases we can ask, how badly do they each fail to meet this criterion?

4.3.3 Performance Distribution

A description of the performance of the system in the cases in which the robot program does not meet its performance criteria contains valuable information. Let us consider that the sensor and actuator models are now extended to include the case of sensor and actuator failure. For the repeated traversal mission, not only may the robot position drift from the goal locations, it may go catastrophically wrong as the robot becomes stuck at a location.

Consider the graphs shown in Figure 2. The horizontal axis is position and the vertical is the likelihood of attaining that position given the environment in which the program is executed. The location of the point B is indicated as a vertical line intersecting the horizontal axis.

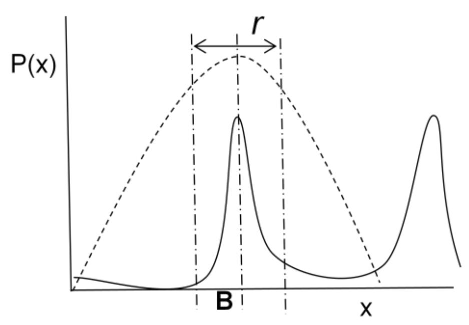


Figure 2: Two examples of spatial distributions

The figure shows examples of two different models for the distribution of the spatial likelihood. The first, shown as a dotted line, is one in which the likelihood falls off smoothly on either side of the location B. If a threshold range *r* around location B is selected, and the performance criterion asks the likelihood of the robot being within *r* of location B, then in both of the example distributions shown here, the likelihood is fairly large. However, in the case of the distribution shown as a dotted line, the failure cases are also locations close to location B. This is a model of a favorable kind of failure.

This is in contrast to the distribution indicated as a solid line in Figure 2. In that case, few of the failure cases, those cases outside of the spatial interval *r* around B, are close to B. The failures in this case are mostly severe failures.

4.4 Probabilistic Robot Program

The final level of complexity that we add to the simple scenario introduced in this section is the inclusion of probabilistic algorithms for control of the robot mechanism. Probabilistic algorithms have been developed for many applications including mapping and for robot localization. Let us consider that we add a probabilistic localization algorithm, such as Monte-Carlo Localization, to the robot program that controls the robot to carry out the repeated traverse mission and explore what this implies for the performance criterion.

The effect of a good probabilistic algorithm should be to improve the performance of the robot in a noisy and uncertain environment, and that of a poor algorithm, to reduce the performance. The mission designer is only interested in whether the robot can achieve location B, with constraints perhaps on the time, the number of traversals and so forth. We note therefore that although the addition of this probabilistic algorithm complicates the mechanics of verification, it does not change the form of the performance guarantee for the program.

5. INTEGRATING VERIFICATION AND DESIGN

This performance guarantee component is being embedded into the *Missionlab* software package, a comprehensive robot mission development, simulation and execution environment. The robot software designer builds her program within *MissionLab* using the visual software authoring tools provided. *MissionLab* allows the high-level mission that is generated to be tested in simulation first, for verification of the user's intent, and then deployed to one or more robot platforms for execution.

The newest components of *MissionLab*, which are based on the formal modeling described in Lyons and Arkin [12], allow the designer to carry out an additional software verification step to establish performance guarantees for the user-defined mission software. This can be very useful in mission-critical or emergency response situations (including C-WMD missions such as finding, containing, and neutralizing Chemical-Biological-Nuclear (CBN) weapons), where it is not uncommon for robot operators to customize the robot software, and even hardware, for the specific mission; and failure of the mission is not an option in these emergency situations.

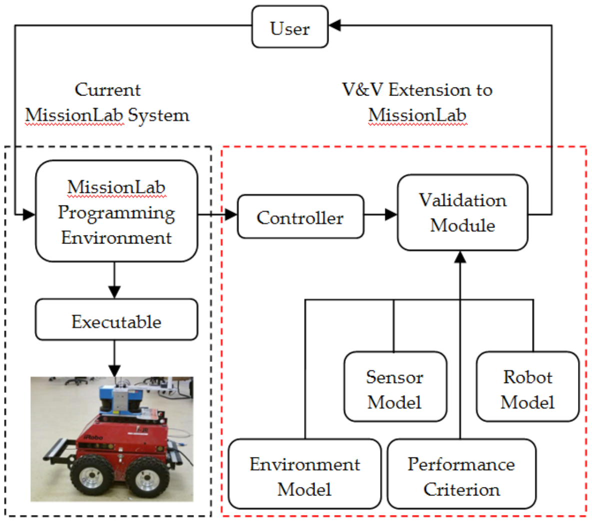
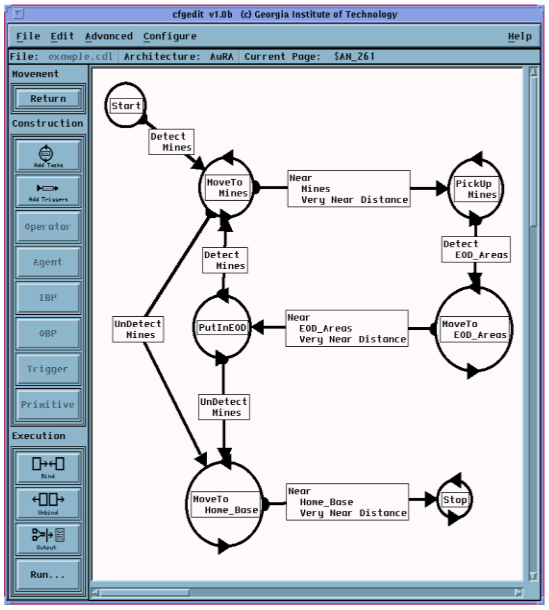


Figure 3: *MissionLab* System with integrated verification module.

Figure 3 depicts the verification extension to the existing *MissionLab* system. The extension provides an operator feedback loop in the robot software design process. The process starts with the designer creating a robot program in the usability-tested *MissionLab* programming environment for a specific mission [6] [14]. Once the high-level mission is specified, the designer may simulate the robot behavior within *MissionLab* to verify correct behavior according to the operator's intent. However, this simulation cannot ever fully capture the interaction between the robotic hardware and the real environment. To further guarantee mission success in the real environment, the robot controller can be validated using the verification module. The verification module provides an output to the user indicating whether the controller will meet the performance criteria specified by the operator. If the controller cannot meet the specified criteria, the designer may modify the robot program and the design loop continues. Once it does satisfy the requisite criteria, the designer may proceed to generate an executable for the robot and then deploy it to undertake the mission.

5.1 Verification Module Inputs

The inputs to the verification module are the robot software controller (specified in an intermediate language referred to as CNL [17]), sensor, robot, and environment models, and the user-specified performance criteria. In *MissionLab*, the robot controller is specified visually by the designer at a very high level of abstraction. An example of using *cfgedit* in *MissionLab* to design a mission is shown in Figure 4. The models of sensors, robots and the environment in which the robot program will execute can simply be selected from existing libraries. These libraries are part of the verification system and are constructed using the modeling approach described in this paper. Figures 5-7 show examples of the model libraries. Due to the limited space here, only a subset of exemplar components of the libraries are shown.



File: example.cdl | Architecture: AuRA | Current Page: $AN_261

Figure 4: Example of Mission Design in *MissionLab*

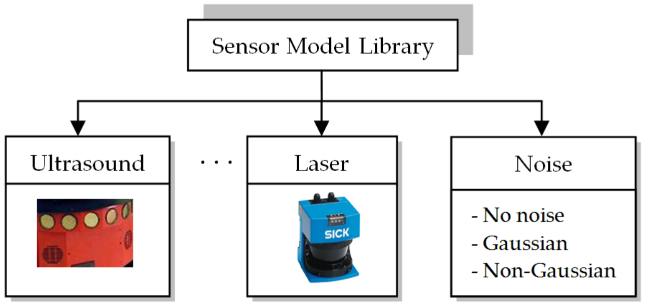


Figure 5: Example of sensor model library

Once the mission has been built, the designer selects from the libraries of sensor and robot models that include a range of noise and uncertainty characteristics (Figures 5 and 6). In a similar fashion the designer composes an environment model by selecting from a library of environments (Figure 7).

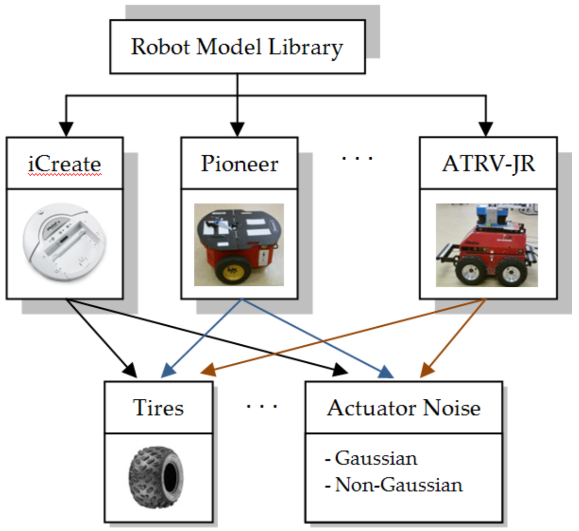


Figure 6: Example of robot model library

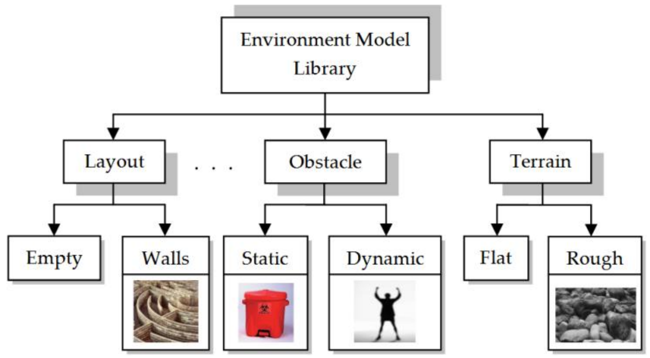


Figure 7: Example of environment model library

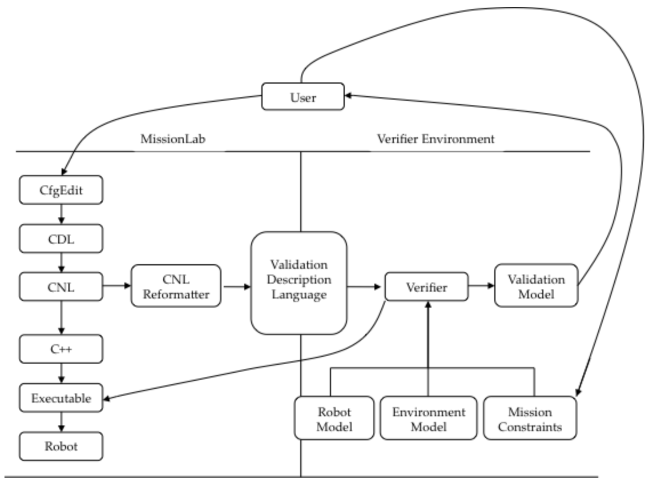
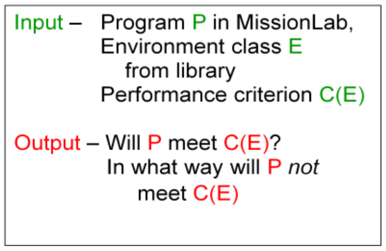


Figure 8: Overall architectural design showing user interaction

Based on the sensor, motor and environment choices made, the designer is offered a selection of customizable verification conditions and constraints. Verification includes the testing of the combination of robot program with the environment model for specific properties of safeness, liveness, and/or efficiency. The result of this testing is the establishment of performance guarantees for the software in the environment represented by that environment model. If the result is unsatisfactory, in terms of design objectives, the designer can use the feedback from the verification to iteratively refine the robot program. In other words, besides telling the designer "yes/no" that the robot program is satisfactory vis-à-vis the mission, the verification module also identifies potential causes of failure in the program and provides the designer with this useful information. This process is illustrated in Figure 8.


Input – Program P in MissionLab,
Environment class E
from library
Performance criterion C(E)

Output – Will P meet C(E)?
In what way will P *not*
meet C(E)

Figure 9: Verification Module Input and Output

5.2 Verification Module

The verification module is based on an approach introduced by Lyons and Arkin [12] to present robot programs and the environment in which they operate as networks of processes. The programs and environments are specified and analyzed using process algebra [13], which is a mathematical framework that takes a compositional approach to describing process networks. The semantics of a process in this framework is a port automaton: an automaton augmented with the ability to send and receive communication messages.

This approach has a number of important advantages:

- The robot program, sensor and actuator models, and environment model can all be specified in one notation.
- The concurrent and communicating composition of program, sensor and actuator models and environment is the object of verification
- Noisy and incomplete information is represented as the interaction of stochastic processes.
- The algebraic foundation supports verification by automated algebraic reasoning rather than by 'simulated execution' or enumerative model checking, both of which have significant computational complexity.

The verification module does not need to carry out a general software verification step, e.g., [9]. In general purpose software verification, the verification criterion can include a constraint on any of the variables within the program and their value.

The performance guarantee in our application concerns the robot and its operating environment, not the robot program directly. Variables from the environment, such as the position of the robot, time, and so forth, can be included in the performance guarantee. However, variable values within the robot program are only of interest in so far as they may affect these variables from the environment.

Furthermore, the models for the robot and its environment, selected by the mission designer to validate the program, come from the robot, sensor and environment libraries mentioned earlier. This means significant preprocessing can be carried out on these models to simplify their composition with other models, and their verification with a robot program.

6. CONCLUSION

In this paper, we described a software framework for validating performance guarantees for C-WMD missions based on extensions to the *MissionLab* mission specification system and on a novel process algebra approach to represent robot programs and operating environments. The key focus in the paper is on the problem of what the performance guarantee should look like from an operator's perspective. We reviewed the state of the art in performance measurements for robots and presented candidate measurements for the performance guarantee. Using a simple example scenario, we looked at the implications of uncertainty in sensor and actuators, as well as uncertainty in the environment, on the form of the performance guarantee.

To be useful to a mission designer, the performance guarantee must allow intuitive expression of the variance in performance of the program due to uncertainty, including the use of the expected value of environment variables, the likelihood of an environmental variable being within a specified range, and, to understand the severity of failure, the distribution of values for an environment variable.

The study described in this paper serves as the basis for our on-going work for the Defense Threat Reduction Agency in process algebra verification of robot missions and in the construction of the verification module for *MissionLab*.

7. ACKNOWLEDGMENTS

This work was supported by the Defense Threat Reduction Agency, Basic Research Award # HDTRA1-11-1-0038.

8. REFERENCES

[1] Arkin, R.C., *Behavior-based Robotics*, MIT Press, 1998.

[2] Arkin, R.C., Diaz, J. *Line of Sight Constrained Exploration for Reactive Multiagent Robotic teams*, AMC02, July 2002, pp. 455-461.

[3] Balch, T. and Parker, L., *Robot Teams: From Diversity to Polymorphism*, AK Peters, 2002.

[4] Clark, E., Grumberg, O., Peled, D., *Model Checking.* MIT Press 1999.

[5] Daniele Calisi, Daniele Nardi *Performance evaluation of pure-motion tasks for mobile robots with respect to world models*, *Autonomous Robots* 27(4):465-481,2009.

[6] Endo, Y., MacKenzie, D., and Arkin, R.C., Usability Evaluation of High-level User Assistance for Robot Mission Specification, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 34, No. 2, pp. 168-180, May 2004.

[7] Hinchey M.G., and J.P. Bowen, *High-Integrity System Specification and Design*, FACIT series, Springer-Verlag, London, 1999.

[8] Jacoff, A., Messina, E., Standard Test Methods For Response Robots, ASTM E54.08.01 Intelligent Systems Division, NIST 2011

[9] Jhala, R., Majumdar, R., Software Model Checking, *ACM Computing Surveys* V41 N4 Oct 2009.

[10] Kosecka, J. (1996). *A Framework for Modeling and Verifying Visually Guided Agents, Analysis and Experiments*, Ph. D. dissertation, Dept of Computer and Information Science, University of Pennsylvania.

[11] Kress-Gazit, H., and G. J. Pappas, Automatic Synthesis of Robot Controllers for Tasks with Locative Prepositions, *IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, May 2010.

[12] Lyons, D., and Arkin, R., Towards Performance Guarantees for Emergent Behavior, *Proc. 2004 IEEE International Conference on Robotics and Automation*, New Orleans, LA, May. 2004.

[13] Lyons, D.M., *Representing and analyzing action plans as networks of concurrent processes*. IEEE Transactions on Robotics and Automation, V9 N3 June 1993 pp.241-256.

[14] MacKenzie, D., and Arkin, R., Evaluating the Usability of Robot Programming Toolsets, *International Journal of Robotics Research*, Vol. 4, No. 7, April 1998, pp. 381-401.

[15] MacKenzie, D., Arkin, R.C., and Cameron, R., *Multiagent Mission Specification and Execution*, Autonomous Robots, Vol. 4, No. 1, Jan. 1997, pp. 29-52.

[16] Madhavan, Raj; Tunstel, Edward; Messina, Elena (Eds.), *Performance Evaluation and Benchmarking of Intelligent Systems*, ISBN 978-1-4419-0491-1, 2009.

[17] MissionLab v7.0 User Manual, available at http://www.cc.gatech.edu/aimosaic/robot-lab/research/MissionLab/mlab_manual-7.0.pdf

[18] Muñoz,N.D., and J. A. Valencia, N. Londoño, *Evaluation of Navigation of an Autonomous Mobile Robot*, 2007.

[19] Ramadge R.J., and W. M. Wonham, 1987. *Supervisory control of a class of discrete event processes*. SIAM J. Control and Optimization, 25(1), pp. 206-230.

[20] Thrun, S., Burgard, W., and Fox, D., *Probabilistic Robotics*, MIT Press 2005.