AI Magazine Volume 22 Number 3 (2001) (© AAAI)

RIACS Workshop on the Verification and Validation of Autonomous and Adaptive Systems

Charles Pecheur, Willem Visser, and Reid Simmons

The long-term future of space exploration at the National Aeronautics and Space Administration (NASA) is dependent on the full exploitation of autonomous and adaptive systems: Careful monitoring of missions from earth, as is the norm now, will be infeasible because of the sheer number of proposed missions and the communication lag for deep-space missions. Mission managers are, however, worried about the reliability of these more intelligent systems. The main focus of the workshop was to address these worries; hence, we invited NASA engineers working on autonomous and adaptive systems and researchers interested in the verification and validation of software systems. The dual purpose of the meeting was to (1) make NASA engineers aware of the verification and validation techniques they could be using and (2) make the verification and validation community aware of the complexity of the systems NASA is developing.

The workshop was held 5 to 7 December 2000 at the Asilomar Conference Center in Pacific Grove (near Carmel) California. It was coorganized by Charles Pecheur and Willem Visser from the Research Institute for Advanced Computer Science (RIACS) and Reid Simmons from Carnegie Mellon University. RIACS gave financial and administrative support, with Peggy Leising handling the local arrangements. We invited 42 participants to the workshop, with 28 from the verification and validation community and 14 from the autonomous and adaptive system community; half of the participants were from NASA and the other half from universities and research labs.

The workshop was run over two days, with the first used for the presentation of four NASA autonomous and adaptive system development projects as well as one talk on the verification and validation of neural nets used in highway applications. The second day was used for three technology break-out sessions to discuss verification and validation issues of autonomous and adaptive systems. The workshop concluded with an open discussion on the results of the break-out sessions.

The long-term future of space exploration at the National Aeronautics and Space Administration (NASA) is dependent on the full exploitation of autonomous and adaptive systems, but mission managers are worried about the reliability of these more intelligent systems. The main focus of the workshop was to address these worries; hence, we invited NASA engineers working on autonomous and adaptive systems and researchers interested in the verification and validation of software systems. The dual purpose of the meeting was to (1) make NASA engineers aware of the verification and validation techniques they could be using and (2) make the verification and validation community aware of the complexity of the systems NASA is developing. The workshop was held 5 to 7 December 2000 at the Asilomar Conference Center in Pacific Grove, California.

The five talks on the first day were (1) Deploying Robust Autonomous Systems: Lessons Learned from the Remote Agent Experiment by Nicola Muscettola from NASA Ames Research Center; (2) First Steps Toward Neural Net Verification and Validation by Rodger Knaus from Instant Recall, Inc.; (3) Stability Issues with Reconfigurable Flight Control Using Neural Generalized Predictive Flight Control by Don Soloway from NASA Ames Research Center; (4) Verification and Validation of an Autonomous Agent for Mars Duty at KSC by Peter Engrand from NASA Kennedy Space Center; and (5) Distributive Adaptive Control for Advanced Life Support Systems by David Kortenkamp from NASA Johnson Space Center.

The discussion topics for the break-out sessions were on the verification and validation of intelligent, adaptive, and complex systems. In the rest of this article, we first highlight some of the general issues that were raised during these three break-out sessions as well as in the wrap-up session that followed and then give short summaries of each of the sessions. The last section contains a short retrospective on the workshop and the future of the field.

General Issues

Some of the issues that were discussed throughout the workshop range beyond autonomous and adaptive systems into the more general fields of formal verification and software engineering. This section cites the more significant ones.

Scalability

Lack of scalability is seen as a major obstacle to current verification methods such as model checking. There is definitely a need for improving and extending these methods to be able to address real-size systems. Because formal methods do not scale well, it is most productive to apply formal methods to only the critical areas, where developers have the least confi-

… because the model concentrates all the application-specific knowledge in an abstract representation, it is potentially more amenable to verification and validation, even for larger systems.

dence in the correctness.

Software Engineering Practices

Good software verification and validation starts with a good software engineering process, including clearly defined goals and requirements. Such practices are not as well established in the autonomy software community as in the mainstream software industry. Well-documented requirements are essential for driving the verification and validation work.

Metrics

We have to define ways to measure and compare the utility of different verification efforts. Thus, we need quantitative metrics that adequately address the different factors of the costs and benefits of each method. Such metrics are a necessity to clearly indicate "where you win" by using new verification approaches. It is noteworthy that most of the latest NASA project funding programs required the mention of such evaluation metrics.

Using Different Techniques

It is rarely the case that a single verification and validation technique achieves good results on a real-world problem. In most cases, several techniques (model checking, testing, proofs, static analysis, and so on) must be combined to be able to tackle the complexity of the system to be verified. Progress in integrating different verification and validation techniques is therefore crucial.

Certification versus Debugging

Verification and validation techniques can be used to achieve two complementary purposes: (1) proving a system correct (certification) or (2) proving it incorrect, that is, finding errors (debugging). The terms *verification* and *falsification* have also been coined. Both eventually help to increase the confidence in the reliability of the system but in different ways. Debugging is done in earlier stages as part of the development, and certification is performed independently on the finished product. It should be noted that "easy" verification and validation techniques such as model checking are often limited to debugging because the state space of real-size systems cannot be covered completely.

Design for Verification

Verification and validation can be facilitated if all components are designed with verification in mind. For example, verification and validation of a fault diagnosis system is easier if the controlled system has mechanisms to inject or simulate faults.

Run-Time Verification

Automatic verification techniques, such as model checking, can also be useful at run time during normal operation. For example, model checking can be used to check the results of a heuristic AI-based algorithm such as a planner. Such a run-time verification combines the efficiency of heuristic search with the robustness and formality of verification.

Verification and Validation of Intelligent Systems

This break-out group gathered 13 people, 6 of them from NASA. The topic had been set to verification and validation of intelligent systems, which had been defined as systems based on some form of AI technique, such as model-based, rule-based, or knowledge-based systems. In accordance with the theme of the workshop, there was of course an interest in autonomous and adaptive systems, but the focus was specifically on the AI-related issues of such systems.

The moderator (Charles Pecheur) briefly introduced the topic and presented some proposed issues, then a lively, free-form debate ensued. The discussion turned out to be strongly focused on model-based systems, which did not stem from an intentional orientation or a perception that such systems are more relevant to the topic but, rather, from a strong involvement of model-based autonomy specialists in the discussion.

This section reports on discussion topics related to model-based systems. Many of the more general issues presented earlier also arose during the discussion.

Verification and Validation of Model-Based Systems

For the purpose of this discussion, a model-based autonomous system is viewed as a plant (a spacecraft, a robot, and so on) driven by a controller through a command-sensor feedback loop. The controller itself is based on a generic, AI-based inference engine that peruses an abstract model of the plant. The engine infers the appropriate control actions based on the feedback it receives from the plant and its knowledge about the plant extracted from the model.

As an overall issue, there is a need to define and build experience in the software engineering process for model-based systems. What are the types of requirement that a customer would expect; how could these requirements conveniently be expressed and verified? Can we develop or specialize a theory and practice of testing for this kind of system? In this prospect, abstract autonomy models could provide a good basis for automatic generation of test cases.

A natural approach is to decompose the verification and validation problem across the three core components of a model-based system: (1) the plant, (2) the engine, and (3) the model. Verification and validation of the plant is outside the scope of this discussion. Verification and validation of the engine is a complex task that needs to be addressed but concerns the design-

ers of the engine. From the point of view of the application designer, verification and validation focuses on the model and how it affects the operation of the whole system, which can be decomposed into two threads: First how do we build and verify-validate autonomy models? Second, given a "valid" model, how do we verify-validate the resulting autonomous controller?

Note that because the model concentrates all the application-specific knowledge in an abstract representation, it is potentially more amenable to verification and validation, even for larger systems.

There is even a hope that model-based autonomy is "correct by design": If the model directly captures the specification of the plant, then the correctness of the controller derives from the correctness of the logic inference principles implemented in the engine. However, experience shows that authoring autonomy models is still a difficult and error-prone task and that full correctness (and completeness) of the engine is not always achieved or even desired for performance purposes. In practice, both threads are needed and complementary.

Part of the difficulty resides in reliably writing complete and consistent models. Accordingly, tools for checking consistency and completeness would be useful. Some of this difficulty might be inherent in declarative specifications, but a part of it could be alleviated by richer modeling languages as well.

The model itself can be further decomposed into the different aspects that it captures, such as the plant (for example, define the moving range of a camera), the operational constraints (for example, do not point the camera toward the Sun), and the goals (for example, minimize the delay when the camera moves). Although all three can be expressed in the same logic formalism, they entail different verification and validation activities and should thus be distinguished from each other. Another interesting issue is the fusion of partial models, involving conflict-resolution principles.

Finally, a comparison can be made with the field of classical feedback control. For linear systems, one can, on mathematical grounds, extrapolate a limited set of observations to entire regions of the control space. We should investigate whether the high-level, uniform inference laws used in model-based reasoning would allow a similar reasoning. This idea is very speculative, given the complexity and nonlinearity of autonomous controllers, but it could dramatically decrease the cost of verification if it proved successful.

Verification and Validation of Adaptive Systems

This break-out group gathered nine participants, four of them from NASA, around the topic of verification and validation of adaptive systems. In particular, the group focused on control systems that do learning, either offline (pretrained) or online. Although the focus was fairly broad, much of the discussions centered on approaches based on neural networks.

Initial discussion centered on how verification of adaptive systems differs from verification of traditional control systems. One point was made that adaptive systems have more potential fault modes and, thus, can behave more unpredictably. Another point is that most commercial verification and validation products are based on the software engineering process and, thus, are not really appropriate to learning systems, where the development of the learning program is often secondary to the way it is trained. It is also the case that most current coverage criteria are process oriented and not product oriented (this is a problem even for verification and validation of object-oriented programs!).

Adaptive systems are mainly used when there is no good model of the plant; thus, it is hard to determine what to verify against. It was thought that it is often difficult to specify requirements or acceptability criteria for complex adaptive systems. One recurring theme is that adaptive systems often do not have a good way of telling when they are outside their range of expertise. It was suggested that other methods (such as putting "wrappers" around the neural net code) are needed to prevent such systems from trying to operate outside their range.

Problems exist for verification and validation of both offline and online adaptive systems. For offline adaptive systems, the idea is to train a system and then verify it. For online adaptive systems, the question is how to do verification when the system can evolve many times after it is deployed. It was agreed that verification and validation for online adaptive systems is much harder. We discuss both, in turn.

Offline Adaptive Systems

It was generally agreed that the best current methods for verification and validation of offline adaptive systems are black-box testing and statistical techniques. Although useful, these techniques are not very satisfying because they cannot make any guarantees about stability, coverage, and so on. There is also the problem of traceability: When a bug is found, it is often difficult to "point a finger" at the part (or parts) of the adaptive system that is responsible. Although analysis is possible, the nonlinear nature of most adaptive systems makes formal analysis difficult. Current approaches are either intractable or make very strict assumptions about the form of the plant, which are typically not valid.

We discussed several interesting options. One is to try to prove weaker

Adaptive systems are mainly used when there is no good model of the plant; thus, it is hard to determine what to verify against.

mathematical results than "standard" stability (for example, plain stability rather than the stronger convergence results typically proved for linear systems).

A big problem for adaptive systems is the question of collecting representative data—how to sample and how to determine whether there are "holes" in the test data set. One suggestion was to analyze the learned functions to find partitions of the operating regions where the chosen actions are "similar" and then devise tests for these regions. This partitioning could guarantee coverage for statistical testing. In general, the methodology might be iterative: One would train a system, analyze it to determine how to choose test data, retrain if it fails any of these tests, analyze again, and so on.

Another option is to investigate learning techniques that might be more amenable to formal analysis. Neural nets are widely used, but they are just one of a whole family of function approximators that can be learned. Different families of functions have different characteristics in terms of learnability, expressiveness, sensitivity to noise, and so on. It might be worthwhile investigating whether there are classes of function approximators that are more easily analyzed and, hence, could lead to formal guarantees of safety. For example, one technique described at the workshop uses hyperplanes to approximate the functions of interest. A neural net constructed from a hierarchy of such structures can be expressive, yet tractable enough to lend itself to rigorous analysis of its properties. Such analyses can also aid us in determining how to incorporate domain knowledge into building such function approximators.

Online Adaptive Systems

Three options were discussed for verification and validation of online adaptive systems:

First is continually doing verification and validation as the system evolves (online verification and validation).

Second is verifying that the learning technique cannot move from "safe" areas. The idea here is to demonstrate some sort of monotonicity property—if the system starts out being safe (shown by offline verification and validation), then prove that it cannot become unsafe.

Third is certifying classes of systems rather than single instances. The idea here is that if one could show that a particular structure of neural net is "safe," no matter what training data it receives, then one can have it adapt without worry.

In general, we agreed that this problem is hard and that we did not even understand well what the desired requirements are for online adaptive systems. For example, it is not clear how to specify the failure modes of the system in advance; so, it was clear that monitoring plays an important part in maintaining safety (although it was not clear exactly what this role is). It was suggested that we might need to restrict the types of learning to keep the system safe (for example, not changing the weights of the neural net too rapidly). By explicitly modeling the adaptation process and the process of environmental change, we might be able to estimate the parameters needed to ensure that such safety conditions hold at all times.

The problem might even be unsolvable as stated: If things are changing rapidly, although it might be feasible to use online statistical techniques to detect when dramatic changes have occurred, it might not be possible to guarantee that the system remains safe at all times because adaptation cannot be instantaneous. For example, while adapting to hardware failures, the system might, for a short time, become unstable or unsafe. Is this condition acceptable or not? We might want to require that the system reenter a safe state within seconds or that it adapt at a given speed, which harks back to the point that we do not really understand what the requirements are for online adaptive systems.

Finally, we briefly touched on the issue of adapting to slow degradation in the controlled system, as opposed to qualitative, topological changes in the plant (for example, because of hardware failures). It was agreed that the latter is generally a much harder problem to deal with, both for adaptation and for verification and validation. For example, it was suggested that if training occurs even during the performance phase, perhaps using decaying values of the learning parameters or simulated annealing, then the system could slowly adapt to such changes. However, there are well-known problems where neural nets can "forget" old responses, especially when they are not being trained with a statistically valid sampling of the input space. It was suggested that in this area, formal proofs could give us insight into some of the design issues for adaptive systems.

Verification and Validation of Complex Systems

The group contained 16 people, with a heavy bias toward verification and validation; only 2 NASA researchers from the autonomy and adaptive field were present. The original topic put forward for discussion was the verification and validation of systems with many interacting components, either within one location (for example, layered control architectures) or among several locations (homogeneous or heterogeneous multiagent systems).

The discussion took an interesting turn before any meaningful progress could be made on the stated subject. A debate ensued on the merits of verification and validation in general rather than just limits to only complex systems. In the words of one participant, we spent the whole time justifying the use of formal methods to the two nonverification and validation participants.

Issues

In this section, some of the highlights of this discussion are recounted, but the majority of the issues were presented in the section on general issues from the workshop. The discussion was in the form of questions being raised by the NASA researchers followed by intense discussions. The output of the session was a list of issues (17 in all) from which we list a selection here.

System engineering: System engineering problems are often addressed as software engineering. "It is like

addressing architectural problems that arise with the construction of a bridge as issues of how to engineer blocks of concrete," Gerard Le Lann (INRIA) stated. We should try and learn from other engineering disciplines, especially in how, for example, civil engineers seem to learn far better from their mistakes than software engineers do.

Verification and validation is not possible without a formal specification of requirements for the system under analysis. Stating formal requirements for autonomous and adaptive systems is hard and, as such, not something often done during system development at NASA.

Implementation should not be attempted without a provably correct design. Doing mathematical proofs is hard, even using a theorem prover, but is worth the effort for critical parts of a system. It was also interesting that comments made after the workshop seemed to indicate that many participants felt that proofs of correctness are being shunned in favor of automated error-finding techniques such as model checking and that this trend is worrying and should be addressed.

Formal methods must be customized to specific domains to get maximum benefit from exploiting domain-specific information. Domain-specific knowledge is one of the best ways to attack the scalability issue of formal methods.

Compositional techniques: Just as systems are built up from smaller pieces, one should use compositional reasoning to reduce the complexity of applying formal methods to a complete system. Unfortunately, it is often the case that only a global system property is to be verified, and then it is hard to decompose this property into ones to be shown for components. It is often easier to compose properties known to hold for local components that hopefully will imply the desired global property is valid.

Challenge problem: The members of the group felt that one of the most important issues was for NASA to provide examples of autonomous and/or adaptive systems that need to be verified for the verification and validation community to try out its numerous techniques. Such a test bed would allow both communities to benefit: NASA will be in a better position to defend the use of complex systems to mission managers, and verification and validation techniques would be improved and evaluated with respect to new challenging problems from the autonomous-adaptive domain.

Conclusions

The feedback after the workshop was very positive, but most of all it was clear that the problem of verification and validation of autonomous and adaptive systems is a hard one to solve—especially given that it is even unclear whether verification and validation of "normal" systems can be done with any degree of success with current techniques. Given the importance of this field, we believe this area will be a fruitful research field for some time to come. Interestingly, in an unrelated event (High-Dependability Computing Consortium Kick-Off Workshop at NASA Ames, 10–12 January 2000), one of the main observations made by the formal methods working group was that verification and validation of autonomous and adaptive systems is a long-term problem with at least a 20-year horizon. This independent assessment of the field closely mirrors this workshop's view.

Acknowledgments

The organizers of the workshop are very grateful to all the workshop participants whose comments helped to improve this article. Special thanks go to James Caldwell (University of Wyoming) for his general comments and Rodger Knaus (Instant Recall, Inc), who provided detailed feedback on verification and validation of adaptive systems. We would also like to thank Peggy Leising for her hard work to make this workshop a reality.

Charles Pecheur is a research scientist for the Research Institute for Advanced Computer Science (RIACS) at NASA Ames. He has an E.E. and a Ph.D. from the University of Liège, Belgium. His current research concerns the formal verification and validation of NASA's critical software systems, with a particular focus on model-based autonomous controllers. Prior to this, he used model-checking tools to validate two distributed applications at INRIA Rhine-Alpes in France. His doctoral research at the University of Liège aimed at improving data-type definitions in the LOTOS specification language. He has authored or coauthored 15 publications and 2 conference tutorials. His e-mail address is pecheur@ptolemy.arc.nasa.gov.

Willem Visser received his Ph.D. from the University of Manchester in 1998 for his work on efficient model-checking algorithms for CTL*. Since then, he has been a scientist at the Research Institute for Advanced Computer Science (RIACS), conducting his work within the Automated Software Engineering group at NASA Ames. The main focus of his work is on the application of model checking to the analysis of software programs, and to this end, he has been the principal developer of the JAVAPATHFINDER model checker (a model checker for JAVA programs). His e-mail address is wvisser@riacs.edu.

Reid Simmons is a senior research scientist in the School of Computer Science at Carnegie Mellon University. He earned a Ph.D. from the Massachusetts Institute of Technology in 1988 in AI. His thesis work focused on the combination of associational and causal reasoning for planning and interpretation tasks. Since coming to Carnegie Mellon in 1988, Simmons's research has focused on developing self-reliant robots that can autonomously operate over extended periods of time in unknown, unstructured environments. This work involves issues of robot software architectures that combine deliberative and reactive control; probabilistic planning and reasoning; and heterogeneous, multi-robot coordination. Simmons is also interested in the problem of formal verification of autonomous systems and has done work in automating the verification of fault diagnosis and execution systems. His e-mail address is reids@cs.cmu.edu.

Call for Participation

AAAI–2002 Student Programs

July 28–August 1, Shaw Convention Center, Edmonton, Alberta, Canada

Sponsored by the American Association for Artificial Intelligence

AAAI-02 Student Abstract and Poster Program

AAAI-02 invites submissions to the student abstract and poster program. The goal of this program is to provide a forum in which students can present and discuss their work during its early stages, meet some of their peers who have related interests, and introduce themselves to more senior members of the field. The program is open to all pre-Ph.D students. Nonstudent advisors or collaborators should be acknowledged appropriately, as coauthors or otherwise. However, students are requested to honor the spirit of the program by submitting only work for which they are primary investigators.

Summary of Dates

Authors must submit six (6) printed copies of a two-page abstract describing their research, to arrive at the AAAI office no later than January 25, 2002. We also request that authors submit the URL of a location where reviewers can access complementary material about the student's research. The URL is critical to reviewers because of the brevity of the hard-copy submission.

Notification of acceptance or rejection of submitted abstracts will be mailed to the author by March 22, 2002. Camera-ready copy of accepted abstracts will be due by April 9, 2002.

Summary of Format

Submissions must be printed on 8 1/2 x 11 inch or A4 paper using 12 point type (10 characters per inch for typewriters). Each page must have a maximum of 38 lines and an average of 75 characters per line (corresponding to the LaTeX article-style, 12 point). All abstracts must be no more than two pages, not including the bibliography. The first two pages must include the following: title; the primary author's full name, affiliation, postal address, phone number, URL (if available), and e-mail address; all coauthors' full names and affiliations; text; and any figures, tables, or diagrams. Up to one additional page may be used exclusively for the bibliography if necessary. Papers exceeding the specified length and formatting requirements are subject to rejection without review.

Submissions to AAAI-02 or Other Conferences

Students are free to submit abstracts for work reported in a regular paper submitted to the AAAI-02 or another conference, but not for work that has already been published. Abstracts will be accepted or rejected for the student session regardless of the outcomes of related paper submissions.

Publication

Accepted abstracts will be allocated two (2) pages in the conference proceedings. Students will be required to transfer copyright of the abstract to AAAI.

Poster Session

Accepted abstracts will be allocated presentation time and space in the student poster display area at the conference. Student authors of accepted abstracts must agree to prepare a poster representing the work described in their abstracts and to be available to discuss their work with visitors during their allocated time in the student poster display area.

Student Abstract Submissions & Inquiries

Please send abstracts to:

AAAI-02 Student Abstracts
American Association for Artificial Intelligence
445 Burgess Drive
Menlo Park, CA 94025-3442 USA

Registration and call clarification inquiries may be sent to ncai@aaai.org. All other inquiries and suggestions should be directed to the Student Abstract and Poster Program Cochairs:

Mark Craven
Department of Biostatistics
and Medical Informatics
University of Wisconsin, Madison
1300 University Avenue
Madison, WI 53706
Telephone: (608) 265-6181
Fax: (608) 263-0415
E-mail: craven@biostat.wisc.edu

Sven Koenig
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332
Telephone: (404) 894-5095
Fax: (404) 894-2970
E-mail: skoenig@cc.gatech.edu

Student Scholar and Volunteer Program

AAAI is pleased to announce the continuation of its Student Scholar and Volunteer Programs. The Student Scholar Program provides partial travel support and a complimentary technical program registration for students who are full-time undergraduate or graduate students at colleges and universities; are members of AAAI; submit papers to the technical program or letters of recommendation from their faculty advisor; and submit scholarship applications to AAAI by April 15, 2002. In addition, repeat scholarship applicants must have fulfilled the volunteer and reporting requirements for previous awards.

In the event that scholarship applications exceed available funds, preference will be given to students who have an accepted technical paper, and then to students who are actively participating in the conference in some way. However, all eligible students are encouraged to apply.

After the conference, an expense report will be required to account for the funds awarded. For further information about the Scholarship Program, or to obtain an application, please contact AAAI at scholarships@aaai.org, or 445 Burgess Drive, Menlo Park, CA 94025. Telephone: (650) 328-3123.

All student scholarship recipients will be required to participate in the Student Volunteer Program to support AAAI organizers in Edmonton, Alberta, Canada. The Volunteer Program is an essential part of the conference and student participation is a valuable contribution. Students not requiring travel assistance should only apply for the Volunteer Program, which provides complimentary registration to full time students, including conference proceedings, in exchange for assisting AAAI-02 organizers in Edmonton. This program does not provide any scholarship funds, and is designed for local students or students who have other sources for travel funds. AAAI membership is required for eligibility. For further information regarding the Student Volunteer Program, please contact AAAI at volunteer@aaai.org. The deadline for volunteer applications is May 31, 2002.