

Verification, Validation, and Certification Challenges for Adaptive Flight-Critical Control System Software

Stephen A. Jacklin^{*}, Michael R. Lowry[†], Johann M. Schumann[‡], and Pramod P. Gupta[§]
NASA Ames Research Center, Moffett Field, CA, 94035

John T. Bosworth^{**}, Eddie Zavala^{††}, and John W. Kelly^{‡‡}
NASA Dryden Flight Research Center, Edwards, CA, 93523

and

Kelly J. Hayhurst^{§§}, Celeste M. Belcastro^{***}, and Christine M. Belcastro^{†††}
NASA Langley Research Center, Hampton, VA, 23681

This paper presents some of the unique verification, validation, and certification challenges that must be addressed during the development of adaptive system software for use in safety-critical aerospace applications. The paper first discusses the challenges imposed by the current regulatory guidelines for aviation software. Next, a number of individual technologies being researched by NASA and others are discussed that focus on various aspects of the software challenges. These technologies include the formal methods of model checking, compositional verification, static analysis, program synthesis, and runtime analysis. Then the paper presents some validation challenges for adaptive control, including proving convergence over long durations, guaranteeing controller stability, using new tools to compute statistical error bounds, identifying problems in fault-tolerant software, and testing in the presence of adaptation. These specific challenges are presented in the context of a software validation effort in testing the Integrated Flight Control System (IFCS) neural control software at the Dryden Flight Research Center. Lastly, the challenges to develop technologies to help prevent aircraft system failures, detect and identify failures that do occur, and provide enhanced guidance and control capability to prevent and recover from vehicle loss of control are briefly cited in connection with ongoing work at the NASA Langley Research Center.

I. Introduction

Since the inception of the Wright flyer 100 years ago, aircraft (and spacecraft) have required feedback control to be flown in a stable manner. Whereas the controller in the Wright flyer was a human being who physically sensed vehicle motion and moved mechanical control rods, present day aircraft are controlled by complex, computer-based control systems. These control systems allow even inherently unstable (yet highly maneuverable) aircraft to be flown by human pilots. In the future, advanced controllers will likely fly aircraft autonomously,

^{*} Computer Scientist, Computational Sciences Division, M/S 269-2, Senior Member AIAA.

[†] Computer Scientist and Area Lead, Computational Sciences Division, M/S 269-2.

[‡] Computer Scientist, RIACS/NASA Ames, M/S 269-3.

[§] Senior Scientist, QSS/NASA Ames, M/S 269-3.

^{**} Aerospace Engineer, Controls and Dynamics Branch, MS 4840D, Associate Member AIAA.

^{††} Aerospace Engineer, Project Planning Office, MS 2217, AIAA Associate Member.

^{‡‡} Aerospace Engineer, Flight Systems Branch, M/S 4840D, AIAA Associate Member.

^{§§} Senior Research Scientist, Airborne Systems, M/S 130.

^{***} Senior Research Engineer and Technical Lead, Airborne Systems, M/S 130, Senior Member AIAA.

^{†††} Senior Research Engineer and Technical Lead, Airborne Systems, M/S 161, Senior Member AIAA.

without human intervention. Adaptation to unforeseen events, such as aircraft damage or failed control surfaces, is presently beyond the scope of certified flight systems, yet may soon become a reality.

The advent of autonomous aircraft and the proposed expansion of adaptive control capability have increased the desire to develop safe, reliable control systems for next generation aircraft and spacecraft. Adaptive controllers have been proposed to identify aircraft stability and control derivatives in-flight and also to recover aircraft control in the event of sudden control surface failure.^{1,2,3} Advanced control algorithms to support fully autonomous aircraft and spacecraft operation have also been designed to allow vehicle self-health assessment, navigation, and mission planning to occur simultaneously and without human intervention.^{4,5}

A serious difficulty hindering the deployment of advanced, flight-critical software, however, is the requirement to show that it can operate as intended and with very high reliability. Adaptive controllers that can make rapid and automatic adjustments to enable self-healing in the event of vehicle damage, might also act to make a healthy aircraft un-flyable or a safety hazard to other vehicles. How can it be assured that safety-critical malfunctions can never occur? The software implementation must be verified and validated to provide sufficient assurance of its intended functionality, safety, and the absence of unintended functionality.

II. Verification and Validation Process of Adaptive System Software

The process of checking the correctness of software is termed verification and validation. According to Ref. 6, verification is the evaluation of the results of a process to ensure correctness and consistency with respect to the inputs and standards provided to that process. Validation is the process of determining that the requirements are the correct requirements and that they are complete. With regard to software development, verification is the process of testing the software at each stage of its development to make sure it has been programmed as specified in the software requirements document. Validation comprises the testing effort to assure that the verified software is able to accomplish the purpose as stated in the software requirements document. Validation failures are generally the result of the requirements being stated incorrectly or incompletely.

Figure 1 shows a diagram of the software development life cycle, depicting all stages from requirements to deployment. The software development process has been described as a spiral path, with four or more distinct phases for each turn of the spiral: requirements, design, coding, and testing.⁷ On the left side of this diagram, the requirements are gradually transformed into the actual software; the right hand side depicts efforts to validate the software. If this spiral development path is followed, modified verification and validation plans are required for certification with each turn of the spiral. Whenever the software does not pass a validation activity, it must be redesigned, re-coded, and then re-verified, before it can undergo further validation testing. Scenario-based testing is often used to validate flight control software. This approach relies heavily on testing and makes up a significant portion of the software development costs for modern aircraft. The end point of the verification and validation cycle is intended to be a software product that meets regulatory requirements for safe use in safety-critical applications.

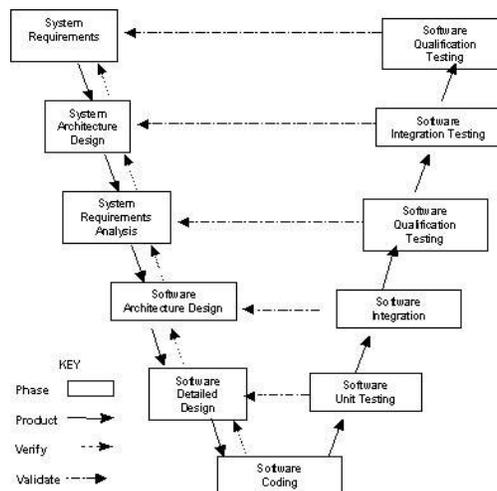


Figure 1: Software development life cycle showing verification and validation steps.⁸

III. Certification Challenges for Adaptive and Autonomous Control Software

Currently, there is no established way to verify and validate adaptive and autonomous flight-critical control software leading to certification. In order for any flight critical software to be certifiable by the Federal Aviation Authority (FAA), it must be developed according to a detailed and well-documented software development process, which is extremely time-consuming and costly. The specific requirements for adaptive systems, however, are not easily discernable.

Requirements for regulatory approval or certification of fully adaptive control systems for next generation aircraft or spacecraft will likely vary according to whether the application is civilian, military, or research, and according to the risk to mission success and public safety. To get a broad idea of certification challenges, it is helpful to consider current regulatory requirements from the FAA for software in commercial transport aviation. Standards used for commercial transport aircraft are arguably the most conservative (compared with US military and NASA standards), yet help outline the ultimate challenge to certifying the routine operation of fully adaptive controllers in the National Airspace System (NAS).

An intricate system of rules and regulations govern the use of software in the NAS. The fundamental tenets, though, are that aircraft systems meet their intended function, do not negatively impact other systems or functions on the aircraft, and are safe for operation. To meet these requirements with software, FAA Advisory Circular # 20-115B specifies the use of RTCA/DO-178B (*Software Considerations in Airborne System and Equipment Certification*) as a means for obtaining certification approval.^{6,9} An adaptive control system would need to comply with the guidance in DO-178B, or provide an equivalent alternate means of compliance, to be used in commercial aircraft operating in the NAS today or in the foreseeable future. This requires that all necessary, relevant requirements and descriptive material is to be provided along with a defined response for the range of valid input data. All of this material must be verifiable, meaning that they can be checked for correctness by a person or a software tool.⁸ The goal is to have a complete requirements document by the end of the development process. Ensuring that the final software product meets its specification is fundamental to compliance with DO-178B. The ability to completely specify requirements and expected behavior is essential under current regulations.

A key problem for adaptive system software (and even conventional) is that few software designs, if any, start with a complete and verifiable software requirements specification. For conventional aircraft systems, the software developed and tested in the lab is intended to be the same software that is fielded. The current approval process outlined in DO-178B is based on this premise. Any changes to a deployed system are handled through a carefully controlled change management process. With a fully adaptive system, the fielded system is expected to learn and change over time. Hence, the behavior of the system during field usage might not duplicate its behavior under test.¹⁰ There is no guidance in DO-178B, or other similar standards, that addresses the evolutionary nature of adaptive systems.

In addition to challenges related to traditional development activities for software, there are certification challenges related to the training of adaptive systems. Standards, such as DO-178B, do not provide guidance for the unique training and learning aspects of neural networks. New review, analysis, and testing methods may be needed to address the following aspects relevant to the training:¹¹⁻¹³

- Quality and completeness of network training data
- Recognition of convergence to local minima versus global minimum
- Network behavior for data outside of the training set
- Network memory and data retention
- Complexity of interaction with other components
- Assessment of performance due to time-variant characteristics

There are ongoing efforts by NASA and others to develop effective criteria, methods, and frameworks that may allow for the certification and safe use of adaptive neural networks in future systems.^{8,11,13,14} Part of the solution may reside in formal verification methods.

IV. Challenges to Extend Formal Verification Methods to Adaptive Systems

The time and cost for certifying new safety-critical systems could potentially be radically reduced through the application of new software technologies. Formal verification methods have been proposed to be used as tools to analyze complex system analysis problems. These methods, however, tend to be costly, time-consuming, and not user-friendly. To be useful, formal verification methods need to be made cost-effective and highly automated. These methods also need to span and link the full lifecycle of software engineering – so verification is a continuous activity.

NASA and other researchers have developed a number of technologies for different aspects of software verification that could meet these criteria. These technologies range from tools that find errors early in the design cycle through tools that find programming language errors and facilitate testing. These tools include the formal methods of model checking, compositional verification, static analysis, program synthesis, and runtime analysis.

Static analysis tools analyze every instruction in the source code to determine if the operations performed in that instruction can create a problem at runtime.¹⁵⁻¹⁷ These tools can efficiently detect a wide range of problems *even before unit testing* including: buffer overruns, un-initialized variables, arithmetic overflows and underflows, and unreachable code. The advantage of static analysis methods is their ease of use. Static analysis programs are used in much the same way an ordinary compiler is used to compile a program. These methods can save many hours of human code review and can therefore provide substantial cost savings. However, a continuing challenge is the development of static analysis methods that can detect more types of problems and that, more importantly, can avoid burying real errors in a large list of possible errors.

The formal methods of model checking,^{18,19} runtime analysis,^{20,21} and program synthesis²² can be used to verify the conformance of adaptive controller code to the software design criteria. Additionally, these tool can also be used to find errors that are almost impossible to find by human code review in autonomous systems for unmanned aircraft and spacecraft. These programs employ multi-threaded programs whose threads execute in parallel and may sometimes interact in unexpected ways or conflict with each other in the use of shared resources. Traditional, scenario-based testing may never discover some bad thread interaction sequences if they occur infrequently or only under circumstances not envisioned by the test team. The use of these formal methods allow complete verification of every possible program execution path and can verify the programs are free of the most severe problems in multi-threaded programs, including thread deadlocks and data races.

A continuing challenge, however, is that the application of formal methods typically cannot be done by general computer programmers or control system engineers. Model checking, for example, requires temporal logic constraints and a model of the control system software to be written in a formal (artificial intelligence-like) language. This step is usually too difficult for control system engineers who program using more standard languages such as C, C++, or Java. Nevertheless, progress in this regard is being made. In one of the largest empirical studies ever undertaken to benchmark the effectiveness of advanced V&V technologies (static analysis, runtime analysis, model checking),²³ it was found that these advanced V&V tools were sufficiently mature to be effectively used by people who are not themselves the tool developers, yet were still computer science professionals. The study showed that different V&V technologies were appropriate for finding different classes of errors, and that all were superior to manual testing. In addition, the study results showed significant potential for synergy among the advanced V&V technologies and with testing. To be effective over the software lifecycle and to reduce certification costs, the formal tools need to be integrated into one program that can be used by engineering professionals in general. This tool would ideally accept control system software code as input directly and produce a list of program errors as output.

V. Challenges for Validation of Safety-Critical Aircraft Technologies

Compliance with DO-178B is non-trivial for both adaptive and non-adaptive systems alike. As mentioned above, the software validation effort must demonstrate that the software meets the design requirements and produces the expected outputs and control actions under all known test conditions. For adaptive and autonomous systems, the proof of this compliance is very difficult due to the non-deterministic nature of the software. Adaptive systems may incorporate advanced system identification techniques (e.g., neural networks) and use state estimation methods (e.g., Kalman filters), thereby making the task of predicting the system output prior to testing a very difficult task.

Moreover, adaptive systems can mask the detection of component failures and can display a host of stability and convergence problems. Several specific validation challenges in this regard are discussed below.

A. Neural Network System Identification Validation Challenges

The validation of neural networks or nonlinear system identification approaches using scenario-based testing methods is difficult because these networks comprise over-parameterized models that typically do not have a unique specification of network connection weights. According to Zakrzewski,²⁴ the on-line adaptation of network connection weights makes it “impossible to predict, in a deterministic sense, the future form of the neural network mapping.” Nevertheless, neural networks are being proposed with increasing frequency as a means of solving adaptive flight control problems.^{1,25}

As a case in point, the Integrated Flight Control System (IFCS) project is being conducted by NASA at the Dryden Flight Research Center in an effort to explore practical adaptive flight control and to develop software tools and techniques for the verification and validation of neural network controlled aircraft. In this research program, a controller with adaptive neural networks is used to compensate for control errors in aircraft dynamics caused by damaged or failed aircraft control surfaces.^{1,2} This research program has demonstrated the need to develop validation methods that can accommodate adaptive control systems whose structure may change in-flight.

The first generation F-15 IFCS used a dynamic cell structure neural network in which both the number of nodes and network connection architecture changed as the system encountered different input data.²⁵ This system was tested using “conventional” tools and methodologies and was deemed acceptable for experimental flight testing at the Dryden Flight Research Center. It was not realized until flight test, however, that the computation time was affected by the number of connections between the nodes. As the number of connections was not predictable prior to flight, an upper bound on computation time could not be established. In addition, it was found that computation time increased as the system was “learning” and actually decreased after the new nodes were added. With all this variability, it proved difficult challenge to validate the worst-case computation time required for this type of neural network.

In an effort to develop a validation tool to handle this type of adaptive system, researchers at the NASA Ames Research Center have created a software tool, called the Confidence Tool, that uses a Bayesian approach to analyze the probability distribution of the neural network output.²⁶ This approach combines mathematical analysis with dynamic monitoring to ensure robust convergence and stability. The Confidence Tool computes the probability density function of the neural network outputs, online, to produce a real-time network variance estimate (Fig. 2). A small variance indicates the network is likely producing a good, reliable estimate, and therefore, good performance of the neural network software can be expected. The confidence tool can be used for pre-deployment verification and as a software harness to monitor quality of the neural network during test flight. Further flight testing may show that the Confidence Tool can be used as a signal to stop and start neural network adaptation and also be used (with modification) to provide a guarantee of the maximum network error for certification purposes.

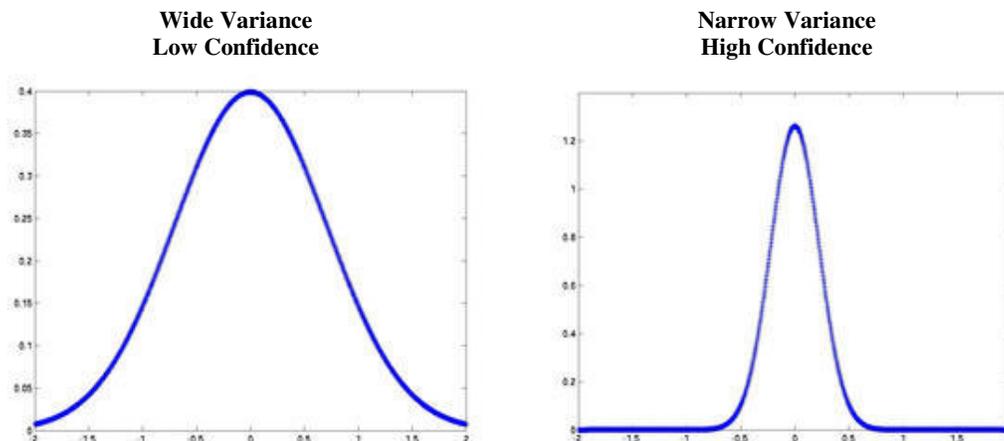


Figure 2. Variance of neural network output as computed by the Confidence Tool.

It should be noted that the validation and certification challenges for off-lined trained, static neural networks are less formidable than for on-line neural networks designed to track changes in aircraft operation during flight. In static or pre-trained neural networks, only the fixed weight values are used in on-line computation of the controls. Zakrzewski and others²⁷ have proposed approaches for assuring static neural networks in compliance with DO-178B. These methods, however, will not serve to address the certification problems of adaptive neural networks.

B. Validation of Adaptive Control System Performance and Stability

Validation of system performance and stability is challenging for adaptive and long-duration autonomous control systems because it is in general not known how the system will adapt over long periods of time. A key problem is ensuring the validation testing has adequate test coverage. Given the wide range of possible test conditions (disturbances, damage), it is difficult to prove complete validation coverage using a test plan that has a finite number of test conditions and network permutations. If there are multiple steady-state conditions, it is desirable to repeatedly test those states for stability in simulation. Unfortunately, some states may take a long time to reach or require a special set of inputs to be realized. A few seconds of time history data to assess frequency and damping of the vehicle response may not provide enough data to prove network convergence over a long period of time. The validation of controller stability and performance is more difficult than conventional, non-adaptive controller software because the design guidelines are not defined nor well understood for adaptive controllers. This obviously makes verification to a pre-defined standard a difficult challenge.

An important challenge, then, is to develop better tools to analyze and guarantee the performance of adaptive control systems over periods of sustained operation. Frequency sweeps or sums of sine waves used to determine stability margins of conventional systems cannot be easily employed on adaptive systems. A verification and validation program for an adaptive control system is needed which can assign quantitative bounds to the control system output errors under all operating conditions, and guarantee that no combination of inputs will result in an undesirable output which may lead to some catastrophic situation. For this reason, it is critically important that the controller's model of the aircraft reflects the actual system with sufficient accuracy. This means that it needs to be shown that for the entire operational envelope, that the modeling error does not exceed a given threshold. This envelope includes not only expected variations in the physical operating conditions, but also situations involving external disturbances and component failure conditions. As mentioned above, advanced tools like the Confidence Tool may be developed to address this need.

In theory, the stability of any adaptive system can be assured using the Liapunov stability method.²⁸ To employ this methodology to an adaptive neural network controller, a Liapunov function must be found to determine how fast the network weights may be updated or changed. It was initially thought that the Liapunov method would yield a suitable criteria for the F-15 IFCS testing that would be analogous to the stability margin criteria of non-adaptive control systems. Unfortunately, the Liapunov method proved to be of little practical use because it was difficult to find an appropriate Liapunov function. In simulation testing, several cases were encountered that apparently met the Liapunov stability criteria, yet resulted in unstable systems. Therefore, the challenge to develop better methods for validating adaptive systems stability remains to be solved for the general case.

C. Failure Detection Challenges in Fault Tolerant Systems

Failure detection in fault-tolerant systems is another extremely difficult validation problem to uncover with robust, adaptive controllers. The main problem is that failure or the slow degradation of a component performance may be accommodated by control law adaptation. Whereas this may well be viewed as a strength for an unmanned vehicle in flight, this property also makes the detection of failures during validation testing much more difficult. A challenge for adaptive systems is how to integrate the design of the controller with failure detection and accommodation systems from the ground up, rather than as separate systems, designed separately. In order to do this, mathematical models need to be developed that consider analytic redundancy, control performance measures, and redundancy management. Some theoretical developments to establish the relationship between reliability and fault tolerant control are provided in Refs. 29 and 30.

D. Effect of Test Inputs on Validation Testing

An interesting challenge for validation testing is that the inputs used for validation testing may cause the system to change in unknown ways. In the validation of non-adaptive systems, it is generally true that the test input does not modify the behavior of the system. With an adaptive system, the test inputs themselves may likely cause adaptation to occur. In some cases, the test inputs might provide the excitation needed for better learning and

adaptation that might not be available to the fielded system. The control system might therefore perform better in response to the test inputs than the real inputs found in the field. In other cases, the test inputs might not provide enough variation to check the full range of possible test conditions. A short period of external disturbance could change the identified system model and controller gains, so that, if followed by some extreme pilot command, catastrophic consequences could result before proper convergence to the correct control values could be reestablished. A challenge, therefore, is how to test a system for this “diaboloic” combination of external disturbance followed by pilot command that could break the system. Methods to estimate the reasonable probability of this occurring must be developed and are currently being studied through several human-in-the-loop (HIL) simulation studies. Some of these methods are highlighted in the next section.

VI. Failure Identification and Recovery from Failure Challenges

The purpose of verification and validation is to ensure the development of software and systems that have a very low probability of failure once successfully deployed. Nevertheless, loss of control is among the highest accident categories across all vehicle classes for both the number of accidents and the number of fatalities. For this reason, the FAA (in Federal Aviation Regulations (FAR’s), such as FAR 25.1309) requires that procedures for all possible modes of failure be considered as part of the requirements for certification. These failures include damage from external sources, the probability of multiple and undetected failures, the effects and severity of failures, failure detection, and what corrective actions are needed for recovery. NASA, the FAA, the aviation industry, and the Department of Defense are developing a number of technologies to help prevent aircraft system failures, detect and identify failures that do occur, and provide enhanced guidance and control capability to prevent and recover from vehicle loss of control.

A validation process under development at the NASA Langley Research Center is an integrated approach involving analytical, simulation-based, and experimental methods.^{31,32} Figure 3 outlines the basic approach. Analytical methods include robustness and worst-case analysis, reliability analysis, and other verification techniques. The analysis results will be used in guided Monte Carlo simulation evaluations and in defining real-time piloted simulation studies. The Systems and Airframe Failure Emulation, Testing, and Integration (SAFETI) Laboratory is being established at NASA Langley Research Center to conduct extensive hardware-in-the-loop testing under a variety of faults, failures, and adverse environmental conditions (including electromagnetic effects), while operating in closed-loop with a vehicle simulation. The SAFETI Lab will also enable the validation of integrated Vehicle Health Management (VHM) and Crash Upset Prevention and Recovery (CUPR) technologies under combinations of adverse and upset conditions, including linked lab experiments. A subscale aircraft flight test capability, the Airborne Subscale Transport Aircraft Research (AirSTAR) test bed, is also being developed at NASA Langley for conducting high-risk flight tests, such as those experienced in upset conditions. The objective is to provide a comprehensive approach to validating and verifying the VHM and CUPR technologies themselves in order to ensure their safety and reliability for commercial application. It is hoped that these methods and other tools can be used to assist in demonstrating compliance with the FAR’s.

A. Challenges for Vehicle Health Management

Vehicle Health Management technologies include sensors and monitors, artificial intelligence and data fusion algorithms, and diagnostics and prognostics to detect and predict failures in the propulsion system, aircraft flight systems, and the airframe structure. A continuing challenge is to develop better failure detection and identification (FDI) algorithms for critical control components, including control surface actuators and sensors. These algorithms need to be implemented in real-time to detect and identify faults and failures during flight. The primary function of the VHM technologies is to prevent system and component failures through condition-based maintenance, rather than the replacement of components at predetermined intervals. In the case of exceptionally long component life, VHM allows components to be used for extended periods of time, thereby enabling a significant cost saving. In higher than expected wear environments, VHM algorithms identify failure trends and call for component replacement or vehicle service before failures occur.

Also challenging for VHM systems is the need to develop diagnostic and prognostic systems for onboard and ground-based use. Onboard algorithms are needed to assess critical components during flight to then either alert the flight crew, or telemeter the diagnosis of any detected problems to ground maintenance facilities. Algorithms for the maintenance crew need to be designed to provide repair guidance and/or maintenance scheduling. It is critical that VHM crew notifications and warnings of failures (and other adverse conditions), be integrated with other existing

aircraft systems, as well as data links and interfaces for improved ground maintenance. A key challenge of implementing these technologies into the aircraft fleet (both for retrofit and forward-fit) is the verification and validation of the VHM algorithms themselves.

B. Challenges to Create Better Control Upset and Recovery Methods

Control Upset Prevention and Recovery (CUPR) technologies provide control under adverse flight conditions in order to accommodate failures, prevent loss of control, and recovery control during loss-of-control events. The primary function of the CUPR technologies is to prevent or recover from vehicle loss of control both with and without the presence of failures. Prevention algorithms need to be developed to accommodate failures and other adverse conditions while maintaining vehicle control, while upset recovery algorithms need to be developed to recover control in the event that control is lost. These adaptive algorithms must be implemented with varying degrees of automation including cueing the crew about how to recover control, semi-automatic control, pilot-engaged automatic control, and fully automatic control.

A significant challenge is to develop enhanced models of vehicle dynamics under upset conditions and adaptive guidance and control laws to provide control accommodation. In order to effectively develop and evaluate algorithms for control upset prevention and recovery, vehicle dynamics characteristics under loss of control (upset) conditions must be investigated through enhanced vehicle simulations. Upset conditions need to include extreme flight conditions within the normal operating envelope of the aircraft, as well as flight conditions beyond normal operation.

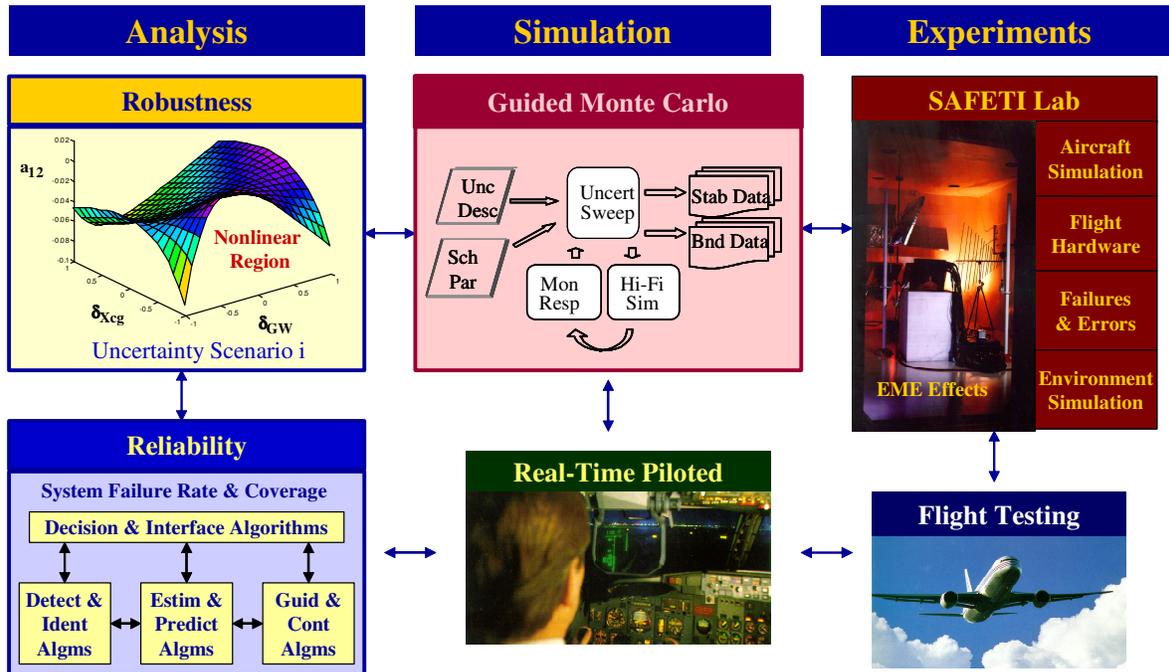


Figure 3. Integration of analytical and simulation methods for validating VHM and CUPR technologies.

VII. Conclusions

In this paper, verification and validation challenges associated with the development of adaptive and autonomous control software were presented. These challenges include:

- The molding of formal verification methods into tools that accept control system software code (e.g. C, C++, Java) as input and produce a list of program errors as output, but do not bury those results in large list of possible errors

- Development of better tools to verify the performance and stability of adaptive control systems over periods of sustained operation
- Development of software verification tools that can assign quantitative bounds to the control system output errors under all operating conditions
- Design of verification (model-checking) methods to ensure total state coverage
- Validation of worst case computation time required by adaptive (neural network) software
- Integration of adaptive controller design with failure detection and accommodation systems from the ground up to allow failure detection in fault tolerant systems
- Development of diagnostic and prognostic systems for onboard and ground-based vehicle health management and maintenance
- Verification and validation of the VHM and CUPR algorithms themselves
- Acquisition of more flight and wind tunnel data to enhance current simulation databases and models

Equally challenging is the task of making verification and validation affordable in the market place. A key aspect of this challenge is to define verification and validation methods that allow the efficient evaluation and confirmation of intended functionality and absence of unintended functionality, for both control algorithms and software implementations. To that end, NASA has developed a number of tools, including the formal methods of model checking, compositional verification, static analysis, program synthesis, and runtime analysis, and also validation techniques that seek to combine analysis and experimental techniques for VHM and CUPR technologies. These tools promise to greatly reduce the time spent in formal code review and testing which comprises a significant part of the cost of software development.

References

- ¹Rysdyk, R. T., and Calise, A. J., "Fault Tolerant Flight Control via Adaptive Neural Network Augmentation," AIAA 98-4483, August 1998.
- ²Kaneshige, J., Bull, J., and Totah, J. J., "Generic Neural Flight Control and Autopilot System," AIAA-2000-4281, August 2000.
- ³Kaneshige, J. and Gundy –Burlet, K., "Integrated Neural Flight and Propulsion Control System," AIAA 2001-4386, August 2001.
- ⁴Gat, E., "ESL: A language for Supporting Robust Plan Execution in Embedded Autonomous Agents," *Proceedings of the AAAI Fall Symposium on Plan Execution*, AAAI Press, 1996.
- ⁵Nayak, P. P., et al, "Validating the DS1 Remote Agent Experiment," *Proceedings of the Space Technology Conference and Exposition*, Albuquerque, NM, September 1999.
- ⁶RTCA/DO-178B, *Software Considerations in Airborne Systems and Equipment Certification*, RTCA, Inc., Washington, D. C., December 1992.
- ⁷Lowry, M., Havelund, K., and Penix, J., "Verification and Validation of AI Systems that Control Deep-Space Spacecraft," *Tenth International Symposium on Methodologies for Intelligent Systems*, Charlotte, North Carolina, October 1997.
- ⁸Mackall, D., Nelson, S., and Schumann, J., "Verification and Validation of Neural Networks for Aerospace Systems," NASA/CR-2002-211409, July 2002.
- ⁹*Advisory Circular #20-115B*, U. S. Department of Transportation, Federal Aviation Administration, issued January 11, 1993.
- ¹⁰Cortellessa, V., Cukic, B., Del Gobbo, D., Mili, A., Napolitano, M., Shereshevsky, M., and Sandhu, H., "Certifying Adaptive Flight Control Software", *Proceedings ISACC 2000*, Reston, VA, September 24-26, 2000.
- ¹¹Taylor, B., Darrah, M., Pullum, L., Smith, J. T., Luxemburg, L., Skias, S. T., and Cukic, B., "Methods and Procedures for the Independent Verification and Validation of Neural Networks," Final Report for NASA NAG5-12069.
- ¹²Stewart, D. B., and Brown, R. A., "Grand Challenges in Mission-Critical Systems: Dynamically Reconfigurable Real-Time Software for Flight Control Systems," *Proceedings of Workshop on Real-Time Mission-Critical Systems in conjunction with the 1999 Real-Time Systems Symposium*, Phoenix, AZ, November 10, 1999.
- ¹³Schumann, J., and Nelson, S., "Toward V&V of Neural Network Based Controllers", *WOSS '02*, Charleston, SC, November 18-19, 2002.
- ¹⁴Bedford, D. F., Morgan, G., and Austin, J., "Requirements for a Standard Certifying the use of Artificial Neural Networks in Safety Critical Applications," ICANN 96.

¹⁵Brat, G., and Klemm, R., “Static Analysis of the Mars Exploration Rover Flight Software”, *Proceedings of the 1st International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, Pasadena, California, July 2003, pp. 321-326.

¹⁶Holtzmann, G. J., “The Model Checker SPIN,” *IEEE Transactions on Software Engineering*, Vol. 25, No. 5, May 1997.

¹⁷Venet, A., and Brat, G., “Precise and Efficient Static Array Bound Checking for Large Embedded C Programs”, *Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'04)*, Washington DC, USA, ACM Press 2004, pp. 231-242. This is the main reference for CGS.

¹⁸Havelund, K., and Pressburger, T., “Model Checking Java Programs Using Java Pathfinder”, *International Journal on Software Tools for Technology Transfer*, Volume 2, Number 4, April 2000.

¹⁹Visser, W., Havelund, K., Brat, G., Park, S., and Lerda, F., “Model Checking Programs”, *Automated Software Engineering*, Volume 10, Number 2, April 2003.

²⁰Havelund, K., and Rosu, G., “Monitoring Programs using Rewriting”, *Proceedings of the 16th IEEE Conference on Automated Software Engineering (ASE'01)*, San Diego, California, November 2001. Selected for publication in the journal *Automated Software Engineering*.

²¹Artho, C., Drusinsky, D., Goldberg, A., Havelund, K., Lowry, M., Pasareanu, C., Rosu, G., and Visser, W., “Experiments with Test Case Generation and Runtime Analysis”, *Proceedings International Conference on Abstract State Machines (ASM'03)*, March 3-7, 2003, Taormina, Italy. *Lecture Notes in Computer Science*, Volume 2589, Springer. Selected to appear in the journal *Theoretical Computer Science*.

²²Denney, E., Fischer, B., and Schumann, J., “Using Automated Theorem Provers to Certify Auto-Generated Aerospace Software”. To appear in *Proceedings of the 2nd International Joint Conference on Automated Reasoning*, Springer, 2004.

²³Brat, G., Giannakopoulou, D., Goldberg, A., Havelund, K., Lowry, M., Pasareanu, Venet, A., Washington, R., and Visser, W., “Experimental Evaluation of Verification and Validation Tools on Martian Rover Software”, *SEI Software Model Checking Workshop*, 2003. To appear in *Formal Methods in System Design*, November, 2004.

²⁴Zakrzewski, R. R., “Verification of Performance of a Neural Network Estimator,” *Proceedings of 2002 IEEE World Congress on Computational Intelligence*, Honolulu, HI, May 12-17, 2002.

²⁵Jorgensen, C., “Direct Adaptive Aircraft Control Using Neural Networks,” NASA TM-47136, 1997.

²⁶Schumann, J., Gupta, P., and Nelson, S., “On Verification and Validation of Neural Network Based Controllers,” *Proceedings of Engineering Applications of Neural Networks (EANN'03)*, Malaga, Spain, September 2003.

²⁷Hull, J., Ward, D., and Zakrzewski, R. R., “Verification and Validation of Neural Networks for Safety-Critical Applications,” *Proceedings of the American Control Conference*, Anchorage, AK, 2002.

²⁸Slotine, J.-J. E., and Li, W., *Applied Nonlinear Control*, Prentice Hall, 1991.

²⁹Wu, N., “Eva: Reliability of Fault Tolerant Control Systems: Part I.,” *Proceedings of the IEEE Conference On Decision & Control*, 2001.

³⁰Wu, N., “Eva: Reliability of Fault Tolerant Control Systems: Part II.,” *Proceedings of the IEEE Conference On Decision & Control*, 2001.

³¹Belcastro, C. M., and Belcastro, C. M., “On the Validation of Safety Critical Aircraft Systems, Part I: An Overview of Analytical & Simulation Methods,” *Proceedings of the Guidance, Navigation, and Control Conference*, Austin TX, August 2003.

³²Belcastro, C. M., and Belcastro, C. M., “On the Validation of Safety Critical Aircraft Systems, Part I: An Overview of Experimental Methods,” *Proceedings of the Guidance, Navigation, and Control Conference*, Austin TX, August 2003.