REPORT ON THE WORKSHOP FOR AVIATION SOFTWARE SYSTEMS FOR THE SECOND CENTURY OF FLIGHT: DESIGN FOR CERTIFIABLY DEPENDABLE SYSTEMS (HCSS-AS)

October 5-6, 2006

Claire Tomlin, Stanford and UC Berkeley, co-chair
R. John Hansman, MIT, co-chair
Jonathan Sprinkle, UC Berkeley, co-chair

(This document is submitted to NSF the final report for NSF CNS-0647523.)

December 20, 2007

About this Report

The High Confidence Software and Systems (HCSS) Coordinating Group (CG) of the Federal Networking and Information Technology R&D (NITRD) Subcommittee, Committee on Technology of the National Science and Technology Council, jointly with NSF, sponsored a workshop on Aviation Software Systems for the Second Century of Flight: Design for Certifiably Dependable Systems (HCSSAS) on October 5-6, 2006. The workshop was held in the Hilton Alexandria Old Town in Alexandria VA.

The Federal government recognizes that the rapidly increasing software and system complexity of aviation systems makes the development of high integrity, high confidence aviation software and systems a crucial issue for the future of military and civilian aviation systems. The purpose of the HCSSAS workshop was to provide an open, working forum for leaders and visionaries concerned with Aviation Software Systems from industry, government, research laboratories, and academia, with the goal of developing a roadmap to overcome crucial aviation software and systems issues and challenges facing the specification, design, certification, and testing of aviation software systems. In addition, the workshop served the goals of bringing together practitioners (from government and industry) with the academic community interested in high confidence systems technologies and mathematical and computational techniques for systems verification and validation.

Position papers were solicited broadly from the aerospace, control, embedded systems and software, and real time operating systems communities, and 35 position papers were accepted to be presented at the workshop. The workshop also featured four invited keynote talks, and several breakout sessions. Overall, the workshop included 75 participants. This report summarizes the key findings of the workshop. Source materials from the workshop, including position papers, presentation slides, scribe notes, and other documents gathered at the workshop, are available at the workshop web location: `http://chess.eecs.berkeley.edu/hcssas/agenda.html`

Executive Summary

Overview

Software related issues are the "Achilles Heel" of modern aviation system development. Traditional embedded software development is characterized by low level programming, ad hoc approaches, stand-alone and static implementations, custom systems, little code re-use. This results in prolonged design schedules, excessive cost, limits in functionality, and difficulty in maintenance, upgrades, and retrofits. In such systems, verification and validation is labor intensive and expensive. These issues are exacerbated for critical systems where high integrity requirements yield certification challenges and barriers.

It is also important to note that current processes are inefficient and inadequate for future needs. Looking ahead, increased functionality in aerospace systems will lead to added complexity. There is a move towards networked, interconnected systems, and a need for distributed computation models. These new designs will feature reconfigurability, adaptability, and dynamic modifications. Mixed initiative systems will be featured: it is not understood how to incorporate the human element into modeling, verification and validation. Code correctness and safety concerns will be paramount, as will security issues.

Thus, new approaches, understanding and breakthroughs are required. Success would be a significant economic and opportunity stimulant: these issues recognized by many organizations but real progress has been slow.

PART I: Background, Scope, and Framework

Purpose and format of the national workshop

The purpose of the HCSSAS workshop was to provide a working forum for leaders and visionaries from industry, research laboratories, academia, and government concerned with aviation systems and software. The main goal was to develop a roadmap for overcoming crucial issues and challenges facing the design, certification, and use of aviation software and systems. An additional goal was to bring together aviation systems designers and regulators with systems theorists and engineers experienced in verification and validation. Of particular interest was the organization of technology needs and promising research directions that could change the way that aviation systems are designed and validated in the future. The HCSSAS workshop included plenary lectures, short presentations and panel discussions, as well as breakout sessions. The panels and breakout sessions addressed the following five areas in HCSSAS:

1. **Applications**. Applications include aviation systems from both the civil and military domains. From the civil aviation side, commercial aircraft, general aviation aircraft, unmanned air vehicles, air traffic management (communication,

navigation, surveillance, and decision support systems), as well as integrated air-ground systems and secure aircraft data networks were discussed. On the military side, platforms for different missions were discussed, including persistent precision strike, persistent ISR, and cooperative airspace operations.

2. **Certification.** Certified integrated systems have failed in unexpected ways. Also, the costs to develop certified integrated systems are excessive: there are high initial and incremental costs. Finally, the approaches to certification of integrated systems are not well understood. There is agreement that there needs to be more acceptance of new technologies and methods for certification and system development, and that these methods need to be developed with a very good understanding of the integrated aviation systems. This requires strong collaboration between the certification sector, industry, and academia.
3. **Methods.** A method is a systematic process for specification, analysis, verification, development, and validation of a system. Methods can be formal or informal, and generic or domain specific. It is important to know the limits of a method, and to understand these limitations in the context of the underlying model and assumptions. Domain knowledge is also vital, as it informs and enriches the models and methods. Integrating verification and certification processes with development processes could reduce costs significantly: yet the application these methods must be seamless, to make the barrier to entry sufficiently low.
4. **Systems and Cross-cutting Issues**. Systems engineering is generally defined as an interdisciplinary approach for enabling the realization and deployment of successful systems. It integrates different disciplines and specialty groups into a team effort, forming a structured development process that proceeds from concept to production to operation and life cycle updates. For aviation systems and software, systems engineering must be performed "up front" and must include software components that are "annotated" with the plan that will be used for certification, and verification and validation. Systems engineering needs to define a "contract", and an enforcement policy, for interfaces so as to enable systems certification and verification from the specification of components. The user must be considered an integral part of the entire system. Overall, a systematic philosophy for integrating requirements management and "design-to-test" is needed.
5. **Challenges in Education.** Each of the four working groups above was asked to address the issue of educating the next generation of engineers and computer scientists to meet the challenges outlined here.

Each working group that participated in the workshop was asked to summarize the state of the art in practice, development, and research in its area, and to identify R&D needs and challenges, along with a roadmap to address the needs and challenges. This report, the full document of the HCSSAS workshop, includes the Executive Summary and the four working-group summaries. The presentations of the working groups, keynote speakers, and panelists, along with the submitted position statements of participants, are available on the workshop web location:
`http://chess.eecs.berkeley.edu/hcssas/agenda.html`

Findings from the workshop are summarized below.

PART II: Technical Perspectives and Analyses

1. Applications.

Participants: Scott Lintelman (Moderator), Jim Paunicka (Scribe), Alex Bayen, Ray Bortner, David Corman, Eric Feron, Helen Gill, Kevin Harnett, David Homan, Gabor Karsai, Frankie King, Col. Mike Leahy, Xiaogong Lee, Mingyan Li, Vince Rakauskas, Johann Schumann, Jonathan Sprinkle, Banavar Sridhar.

The applications for high confidence systems and software in aviation arise in both civil and military domains. These include:

- Commercial and military aircraft
- General Aviation (GA) aircraft and Very Light Jets (VLJs)
- Unmanned Aerial Vehicles (UAVs)
- Air Traffic Management (ATM)
- Flight control
- Command and Control (C&C)
- Communication, Navigation, and Surveillance (CNS) systems
- Aircraft and infrastructure integration

Summary of key lessons learned.
Aviation software has not kept up with, and has not leveraged, emerging IT technology: there are concerns with certifiability of emerging technology advances, and our intended use of emerging technology is quicker than our ability to regulate it. Emerging algorithms applied to flight vehicle software cannot be fielded because the associated certification methods do not exist, for example, in dynamic scheduling of complex on-board software components, or in adaptive flight control. In addition, life-cycle cost savings that are the goal of emerging software techniques have not been totally realized. As an example, the architecture implementing change containment in embedded mission management software is not always being recognized by the test community in reducing need for regression tests. Also, world-wide policies for airspace activities are needed to include evolutions that include pilotless vehicles, to include evolution from controller-centric mechanisms to more automated approach, and to address contingencies, world-wide, for 9-11 type events. For example, there is a current reluctance to share critical protocol information with potential foreign participants in airspace, due to considerations such as ITAR.

Timely availability of airplane software upgrades is needed: here, a world-wide solution would be preferred, with the goal of transferring information over open networks using very high assurance PKI (Public Key Infrastructure). Finally, ensuring cyber security (availability of link, data integrity, confidentiality, anti-tamper) for unmanned air systems (UASs) and netted platforms is crucial: here, there is a real need for trusted autonomous systems, and for methods to assess the trustworthiness of software.

Research challenges.
Cost-effective, standardized certification techniques for emerging airborne capabilities represent the major challenge in moving forward. Methods for analyzing fault tolerance and hazards of systems and software, such fault tree analysis, or FMEA applied to aviation are needed, as well as analysis methods for new flight control methods (e.g., adaptive flight control), correct functionality, and for the layered approach to autonomy and fault recovery at different levels. Human interaction with aircraft systems (humans in loop, mixed-initiative) needs to be better characterized. This includes **v**ehicle operators (on-board and off-board), as well as ATM personnel. There needs to be a way of understanding the certification of the system-of-systems aspect of Air Traffic Management, which would be key in handling increased traffic in NAS.

A second key challenge evolves around system trustworthiness and security. Measuring trustworthiness of autonomous systems, including systems with weapons needs to be better characterized. Wireless security, perhaps using high assurance PKI is needed. In addition, a breakthrough in affordable and implementable ways to evaluate and ensure safety and security of aviation systems and software is required: these could be mapped to certification techniques (criteria, processes) to get customer (e.g., FAA, military airworthiness authorities) buy-in. We need to explore the relationship between security and safety, taking into account real-world issues, such as new developments (787, A380) as well as heritage upgrades (e.g., Internet on 737). In particular, it is important to be able to detect unintended vulnerabilities in code that could be exploited by hostile adversary.

Third, there needs to be an acceptance of pilotless vehicles, which would require increased trust in these autonomous systems. For example, an increased confidence that UAVs will be able to exercise appropriate contingency reactions when things go wrong is needed. In such an environment, protocols for in-air conflict resolution in an airspace that includes piloted and unpiloted aircraft are needed: these protocols must be simple and easy to understand, as well as easy to implement.

2. Certification.

Participants: Jim Krodel (Moderator), Richard Robinson (Scribe), Oleg Sokolsky, Tucker Taft, Hal Pierson, Steve Jacklin, Jim Alves-Foss, Natasha Neogi, Patrick Graydon.

Software in aviation systems is regulated according to the FAA regulatory standard DO-178B (Eurocae standard ED-12B): "Software Considerations in Airborne Systems and Equipment Certification". This is a "process-based" certification: certification applies to the end product (ie. airframe), including all systems, it applies to a given application of a given product (other applications of the same product require further certification). DO-178B requires that all code MUST be there as a direct result of a requirement, and it specifies that full testing of the system and all component parts (including the software) must take place on the target platform and in the target environment.

Summary of key lessons learned.
The certification of integrated systems is not well understood: the cost to develop certified integrated systems is excessive, requiring high initial and incremental costs including the cost of rework. There is very little understanding of how to cope with the complexity of an integrated system. Even after these high development costs, certified integrated systems have failed in unexpected ways. There needs to be an acceptance of new technologies, new methods, for system development and certification, and for being able to characterize the deployed system as safe.

Research challenges.
The first major research challenge is to develop alternate approaches to acceptance and certification. The approaches should be product evidence based vs. process based, building on, for example, new techniques in component-based certification, methodologies to certify expressive interfaces in a formal manner, and compositional methods for certification that include environmental aspects and legacy components. All of these approaches need acceptance and working together with the academic, certification and industry communities.

The second major research challenge is the design of new tools for certification, that take into account commercial off-the-shelf (COTS) components, open source components (including both software and integrated circuits). These tools must consider integration of safety, as well as other complex system attributes or requirements (human factors, security or other related system attributes that require resolution of conflicting requirements). Finally, the tools must take into account unmanned vehicle autonomy in a certified system or sub-system, where the control authority is shared between the autonomy and human operator.

3. Methods.

Participants: Azer Bestavros (moderator and scribe), Matthew Dwyer, Allen Goldberg, Paul Jones, Martha Matzke, Paul Miner, Cesar Munoz, David von Oheimb, Calton Pu, John Rushby, Lui Sha, Bill Spees, Reinhard Wilhelm.

A “method” is a systematic process for specification, analysis, verification, development, and validation of a system. It may be classified as formal, or informal; generic, or domain-specific.

	Formal	Informal
Generic		
Domain-Specific		

Summary of key lessons learned.
A method is as good as the model of the system on which it is built: thus claims of correctness must always have a disclaimer about the limitations of the underlying model and assumptions. Successful methods are those that are well-integrated with an application domain, since domain knowledge informs and enriches both the models and the methods. It is rather hard to perform cross-domain analysis, for example, how could timing and power analysis leverage each other?

Integrating verification and certification processes with development processes reduces costs considerably and results in better developed systems and practices. The application of verification methods must be seamless; the barrier to entry should be sufficiently low.

Models need to be rich (complex) enough to be able to deal with the "frame" problem: they should be "resource aware", and interfaces should reflect assumptions about behaviors (e.g., dynamics). Yet models also need to be simple enough for scalability of verification.

There needs to be adoption of an "hour-glass" paradigm for verification and validation, in which one is able to apply methods to intermediate abstract models of systems. For this, we need to figure out what are these "right" abstract models, and interactions involving critical components must be simple.

Methods need to distinguish between manageable versus unmanageable residual bugs: the absence of unmanageable bugs needs to be verified, and strategies for dealing with manageable bugs need to be developed.

Research challenges.
The key specific challenges are to develop automated methods fit for deployment, and which leverage byproducts of method application. Methods and frameworks for integrating disparate analyses and verification methodologies need to be developed, as do better methods/architectures for establishing/enabling non-interference. There needs to be an understanding of the balance between the philosophies of "design only what you can analyze" and "resilience against the unknown unknowns".

4. Systems and Cross-Cutting Issues.

Participants: John Baras (Moderator), Eric Cooper (Scribe), Claire Tomlin, Mingyan Li, Lyle Long, Walter Storm, Peter Stanfill, Kristina Lundqvist, Ernie Lucier, Andres Zellweger, Barbara Lingberg, ElRoy Weins, Glenn Roberts.

Systems engineering is generally defined as an interdisciplinary approach for enabling the realization and deployment of successful systems. It integrates different disciplines and specialty groups into a team effort, forming a structured development process that proceeds from concept to production to operation and life cycle updates.

Summary of key lessons learned.
Systems engineering must be done up front and must include software components that are annotated with the methodology that will be used for certification, verification and validation. A "contract" needs to be defined (and enforced) for interfaces so as to enable systems certification and verification from specification of components. The user must be considered an integrated part of the whole system. In general, a formalized approach to systems engineering is lacking with respect to tracking requirements across system components and generating, refining and clarifying requirements.

A systematic philosophy for integrating requirements management and "design-to-test" is required. This integration must have software certification "hooks" in place for all system components from the very beginning, in order to enable systems level verification and validation, and testing assurance. Models of pilot and user behavior are needed, and these must be integrated into the systems engineering process. A unified model representation is needed to facilitate the use of analysis tools, as are quantitative methods that link verification and validation results back to requirements and testing.

Research challenges.
A key challenge is to define and extend what is meant by certifiable and dependable (reliable, available) including such new elements such as monitoring and self-correcting software components. Methods to perform timing analysis are needed. Both of these challenges are realizable in the next 2-3 years.

Development of an interface modeling methodology to fully describe the component interface behavior so as to enable system verification and validation is needed. Then, change management is required to guide the transition in requirements and technologies. These challenges are realizable in 5-10 years.

In addition, tools to monitor software performance, integrate experimental data into monitoring, for updating software, are needed. Software re-use needs to be understood and quantified. Finally the implications of COTS on system certification needs to be understood (this will differ based on level of certitude).

5. Challenges in education.

How should the next generation of engineers and computer scientists be trained to face the challenges outlined here?

Students should receive training in building production-quality code within schedule and budget, thus the software engineering curriculum should be infused with an appropriate systems engineering content, and similarly, software engineering should be infused appropriately into curricula of other disciplines that will be the writing aviation systems code of the future (e.g., Electrical Engineering, Aerospace Engineering). Topics under consideration should be: developing fault tolerance in software designs, wrapping autocode into a bigger system, and identifying good requirements.

There is great benefit in providing challenge problems from industry to the academic research community. Past examples have included DARPA MoBIES, SEC, PCES, CerTA FCS, NSF-ITR on Embedded Systems experience.

Incentives need to be developed to motivate people to enter the domain and stay current. The professional nature of the practice needs to be better recognized. A demand needs to be created for new programs in universities that focus around high confidence systems and software. In addition, there need to be clear benchmarks to test US education capability vs. world standards (both higher education and in industry). We need to shift culture so that certification is an integral part of system development. Sample systems could be developed for use in academia.

In the current courses in academia, we need to teach system-level thinking, including composability and scalability; avionics systems are very different from other software-based application disciplines. The life cycle view of software systems (from requirements, to design, through operational maintenance) needs to be taught, and industry needs to be involved in the education process (since there is not enough domain expertise currently in academia). Industry internships for students and faculty could be developed. Finally, there needs to be more technically rigorous software engineering programs in the US at all levels (undergrad, grad, and continuing education).