A Pervasive Software Validation Approach for Next-Generation Avionics Systems

Matthew B. Dwyer, Sebastian Elbaum, Steve Goddard
Department of Computer Science and Engineering
University of Nebraska - Lincoln
{dwyer,elbaum,goddard}@cse.unl.edu

1. POSITION

Future aviation systems are the vanguard for a new breed of software that must address opposing forces demanding high confidence and high adaptivity in systems. Advances in autonomous systems will enable unmanned flight to become common place and will provide the opportunity to offload the monitoring and planning of an increasingly dense airspace from human pilots and controllers. Autonomous software, by its very nature, is highly adaptive – it must assess the system environment, plan an appropriate response, and execute that plan. It must do this correctly even in the face of unforeseen circumstances in the environment or failures on-board the system it is controlling. Assuring high-confidence in such systems goes well beyond the state of the art in software testing.

Our position is that researchers and aviation software and systems developers must adopt a *pervasive* approach to software validation and, in particular, that validation must continue into the field with deployed systems. Validation must be grounded in meaningful, demonstrable, and formally defined *evidence of software quality*. Evidence may be contributed to the software quality *case* for a system through the entire software life-cycle, from early phase formal requirements specification, pre-deployment static analysis and verification to assess behavioral conformance with requirements, and, finally, online monitoring of fielded systems for those requirements that cannot be assured statically. An integrated body of evidence would provide a basis for avionics software certification and acceptance. Simultaneously, this evidence will allow *quality-oriented autonomic adaptation* of systems to adapt their run-time behavior to to maximize system safety.

2. CHALLENGES

Our position that a pervasive approach to software validation will be needed is predicated on the fact that the complexity of today's systems is already beyond the state of the art in software verification and validation. To compensate for this weakness, aviation systems are deployed with predefined modes of operation, wherein each mode has been verified and validated in isolation. Any deviation from predefined modes of operation places the aircraft in an invalidated state. Clearly a new approach is needed to validate future systems that will need to adapt to non-deterministic environments in a timely and safe manner.

Software for next-generation aviation systems will include advances in software-enabled control and technologies supporting distributed systems-of-systems. Thus, rather than focus on the challenges associated with specific technical advances in those areas, we believe that it is crucial to emphasize the systemic and cross-cutting challenges facing developers of such systems.

Challenge 1: Develop meaningful, formally defined, quantifiable, direct measures of the quality of software components and systems for aviation systems. The current practice for developing and validating aviation software is insufficient to meet the needs of future systems. For example, DO-178B makes demands on the software development *process* with the inference that such a process will produce high-quality software *products*. This approach is too human-oriented to scale to the size and complexity of next-generation systems. Thus, a shift to product-oriented quality assessment is essential.

Challenge 2: Define a total life-cycle software validation process that produces an explicit body of evidence of software product quality. Software products throughout the life-cycle must be validated and associated with quality measures to enable traceability of linkages between assurances of quality in early phase artifacts, such as requirements models, and late phase artifacts, such as deployed code. In this setting, it is clear that monitoring and adapting the development process based on intermediate product qualify will be essential. Due to the dynamic and unpredictable nature of the aviation systems environment, quality monitoring and adaptation activity must further pervade the execution of systems by enabling run-time systems to adapt and steer system behavior to assure correct operations even in unforeseen circumstances.

Challenge 3: Develop curricula to train software developers in the use of advanced software and systems modeling, development and validation methods that emphasize explicit evidence of software quality. These challenges demand a significant shift in the skill set of software developers. Current curricula typically include very limited exposure to software quality assurance techniques, and almost no exposure to the kinds of formal modeling and formal analysis techniques that will be needed in next-generation aviation software development environments. This is especially true in the area of quality-oriented autonomic computing where even the basic continuous mathematics and control-theory to understand these concepts is missing from existing curricula.

3. RESEARCH NEEDS

The challenges identified are formidable. If we are to meet these challenges, a bold new research agenda must be set. The following research needs provide a starting point.

Formalizing notions of correct software behavior. Existing measures of software quality are an incomplete patchwork of approaches that provide only the weakest information about how a program may operate. We must define approaches to to capture a wide-variety of disparate forms of information about software behavior and to integrate that information into a holistic view of software product quality. For example, rich notions of *behavior coverage* are required to support claims of correct operation across a wide range of system execution contexts. Such notions must account for the space of possible program executions rather than sam-

pling specific syntactic criteria of a program, such as MCDC.

Extending and formalizing correctness by construction. Many researchers advocate the use of technologies for assuring *correctness by construction*. These are promising ideas that must be explored, but in the context of aviation systems it is crucial that the *implicit* assurance provided by such trusted tools can be made explicit. This will allow auditing of such quality evidence and leveraging of that information to drive other software validation activities.

New integrated models of software quality. Software quality models must be able to encode information from a wide range of validation activities, including traditional testing, static analysis and verification techniques, and system synthesis techniques. An integrated model of software quality will allow different validation technologies to contribute to an overall definition of system correctness and to work in a synergistic fashion. For instance, if a system synthesis method generates a component along with a guarantee that it operates according to specified requirements within a specific, formalized, range of behavior, then other methods can target the validation of the component outside of that range.

Specifying and validating intent rather than action or state. Today's systems are specified and validated with very specific low-level actions. Next-generation aviation systems will require the specification of intended action that can be observed, measured, and monitored in both offline, e.g., static analysis, and online, e.g., dynamic analysis, modes. Unless new, high-levels of specification abstractions can be formalized and validated, the complexity of future systems will be limited by our ability to specify them. Imagine trying to develop an enterprise-level application using assembly language. The evolution of development languages has dramatically increased the complexity of the systems we are able to build and deploy. A similar evolution is needed in the way we specify and validate systems.

Pervasive validation approaches. If we had a perfect and scalable method for synthesizing an implementation from a specification we would still not solve the problem of developing trustworthy aviation systems. Humans will always drive the development of requirements and specifications and they may well make errors in defining proper system behavior or, more subtlely, in characterizing the nature of the environment in which the system will operate. For these reasons, we believe that it will be essential for researchers to develop methods to efficiently extend the validation process to system run-time. To achieve this, advances in deployed run-time monitoring, process scheduling, run-time systems and program adaptation must all be pursued in concert with the goal of providing an online monitoring *safety net* that operates transparently with respect to system functional and temporal requirements.

Methodology and tool support. As in any complex engineering domain, aviation software system engineering demands tools and techniques. The nature of these systems is such that tools that are cost-effective for more traditional classes of software, such as enterprise-computing or web-based applications, are completely inappropriate. New tools for software developers that simplify the use of advanced software validation methods are needed. These tools must scale to the size and complexities of distributed real-time aviation systems-of-systems and must demonstrate the ability to increase system trustworthiness.

4. ROADMAP

Given the complex interplay of governmental, industrial and academic participants in driving next-generation aviation software systems any realistic roadmap must account for the different stakeholder positions. Here we only provide a focused research roadmap based on the position and vision espoused in this paper:

Early years It is essential to engage with aviation systems developers to understand and build support for the advantages of explicit statements of quality goals and explicit evidence of goal satisfaction. Without that research advances will not transition to practice at the end of the roadmap.

Years 1-2 Development of families of behavioral coverage models and criteria that account for the diversity of system structure in complex, distributed, real-time systems. This must be coupled with thorough empirical evaluation of the relative costs and strength of using those models.

Years 2-4 Adaptation of existing testing, validation, verification and synthesis techniques to produce behavioral coverage information in an integratable form. Development of adaptive transparent run-time monitoring technologies and maturation of those technologies to the point where they can be integrated into production run-time platforms.

Years 3-6 Development of over-arching validation processes that integrate the application of individual techniques by monitoring behavior coverage achieved and quality indicators and targeting new validation activities to maximize cost-benefits.

Years 5-9 As technologies stabilize, the development of robust usable toolsets and curricula that emphasize the underlying mathematics and modeling required to use those toolsets must be developed.

Years 7-10 To enable effective technology transition, a series of increasingly demanding challenge problems should be posed. These challenges should be driven by industry to meet their needs and responded to by researchers. Both competitive and collaborative approaches to challenge problems can be used to maximize engagement with the research community.

5. CONCLUSION

We have reached a critical junction in the development of aviation software systems. The second century of flight offers both significant promise and grand challenges. New research will be needed to meet these challenges. It is our position that a new pervasive approach to software validation will be needed. This approach must consist of an integrated model of software quality that begins with formal requirements specification and continues through deployment via adaptive, transparent online monitoring.

6. BIOGRAPHIES

Matthew Dwyer, the Henson Chair of Software Engineering at UNL, has over a decade of experience developing static software analysis techniques and tools and studying their application in practice. Sebastian Elbaum is an Associate Professor in CSE at UNL with expertise in run-time monitoring of deployed software and empirical studies of software testing techniques. Steve Goddard, an Associate Professor in CSE and College of Engineering Distinguished Scholar at UNL, is a highly-regarded expert in real-time and embedded systems and enterprise-scale decision support software. This team of researchers has led more then 25 research projects for ARO, DARPA, NASA, USDA, and NSF that have produced more than 10 software systems which have been leveraged for industrial use.