Effective Verification of Flight Critical Software Systems: Issues and Approaches

Devesh Bhatt and Kirk Schloegel
Honeywell Aerospace Advanced Technology
1985 Douglas Dr. N, Golden Valley, MN 55422
{devesh.bhatt, kirk.schloegel}@honeywell.com

a position paper

submitted to the NSF/Microsoft Research Workshop on Usable Verification

The verification and validation of flight-critical software has traditionally required a significant amount of cost and time; often more than half of the entire development budget. Several verification objectives related to various stages of software development, as delineated in DO-178B are satisfied during the verification activity.

For the past several years, Honeywell has been engaged in model-based development of control algorithms and subsequent automation of code generation and verification activities. This effort has encompassed several commercial avionics systems including flight controls, engine controls, auxiliary power unit controls, environment control systems, and cockpit displays. This verification approach was implemented in the Honeywell Integrated Lifecycle Tools & Environment (HiLiTE), and is being applied to control systems of several commercial avionics production programs. Using HiLiTE, Honeywell has reduced the cost and time of certain component-level verification tasks, required by DO-178B certification objectives, by a factor of 20-50 compared to traditional methods.

In this position paper, we summarize the verification capabilities currently being used, the issues encountered and shortcomings of the current efforts, and future capabilities/approaches needed in the areas of component-level and system level verification.

Current Honeywell Automation of DO-178B Component-Level Verification Objectives

HiLiTE is a tool used within Honeywell for the requirements-based verification of aerospace system components that are designed using MATLAB Simulink/Stateflow models. HiLiTE has been qualified for use in DO-178B processes, and has been applied to the verification of thousands of practical MATLAB models from a wide variety of domains. The scalability of HiLiTE usage has been well established for extremely large and complex MATLAB models, over a set of thousands of models across avionics domains. The following verification objectives are currently automated:

1. Verify that the software design (MATLAB model) is accurate, consistent, robust, and verifiable/testable. HiLiTE performs a symbolic static analysis on the model to derive

range bounds on intermediate signals, propagating the ranges through the mathematical functional behavior of the control function blocks in the diagram. The range bounds allow the detection of several types of models defects such as overflow conditions (divide-by-zero), frozen signals, mathematical/logic conflicts, and un-testable conditions. Detecting such defects in early stages of the design cycle is useful from the perspective of cost and schedule reduction.

2. Verify that the object code that is deployed on the airplane implements the design functional requirements correctly and robustly. This is accomplished through static and dynamic analyses by enumerating equivalence classes of behaviors of function blocks – including time-dependent behaviors as well behavioral pivot points that span a range of real numbers and time steps. These analyses result in test cases that are run against the model and/or the object code. A by-product of this verification step is the structural coverage achievement on the object code including.

Lessons Learned during the Current Verification Efforts

Honeywell has applied model-based development and verification automation in the certification of several flight-critical software domains. The following is a summary of certain relevant observations and lessons learned:

- The scalability of the verification methods and associated tools is of utmost importance, since real avionics models can be very large and complex. If a verification tool does not scale to these models, then large parts of the automation benefits are lost. In addition to scalability to the size of models, the verification tools must work for different types of modeling styles, patterns, and types of algorithms. For example, verifying control dynamics aspects of complex hybrid-control algorithms is as important as verifying safety properties.

- The cost benefits of verification automation can be effectively realized only when all specific steps required for a certification process objective are 100% automated, without any need for human translation of models and interpretation/review of verification artifacts. Since DO-178B certification is very costly, industries have highly optimized processes in place to accomplish this. Adding a verification tool that does not precisely map to the process and clearly eliminate verification tasks may increase rather than decrease costs.

- It is important to design for verifiability – which must begin at the higher levels of requirements and design and follow to the lower levels in a model-based design process. It is important that this is recognized as a mainstream activity with appropriate formalisms, metrics, and abstractions to support it so that there is feedback to the designer in early stages to allow changes to improve verifiability.

- There are currently no good methods and tools to formally describe high-level requirements that are commonly captured in textual form. Although tools like model checkers have been used to verify certain “safety properties” derived from the requirements, which is a small part of the verification space. The reason is that requirements describe multiple, diverse aspects of behavior and performance parameters that can span multiple types of modeling constructs and associated engineering domains. This poses significant technical and process-related challenges that will need to be addressed to enable significant automated verification based upon high-level requirements that includes specification and full functional dynamics.

Future Directions for Effective Verification of Avionics Systems

We make the following suggestion for improvements in the verification methods, tools, and processes used for certifying flight critical avionics systems:

- Domain-Specific Verification Techniques: Verification semantics should mimic the levels, and corresponding domain-specific abstractions, of the design process. For example, at the higher levels of the design, higher-level control abstractions should be the basis of doing the verification; i.e., at the control algorithm design level, verification tools should support the control abstractions such as z-domain transfer functions, feedback loops, hybrid control constructs, etc. This is in contrast to the current methods where lower-level states and variables are programmed into model-checker type tools for verifying properties. This results in lack of scalability of verification due to state-space explosion. Recently, we have had success in implementing static analysis using abstractions of certain control constructs such as transfer functions and feedback loops. This has helped derive verification of range bounds and temporal bounds with a high degree of scalability on large, complex models. It is not necessary to a have a unified hybrid control theory to enable this, as long as abstractions of specific aspects of the control design can be excerpted from the design models and subjected to the analysis. Also, there are many application sub-domains such as maintenance, built-in tests, I/O processing, fault-tolerance (selection, voting), discrete mode control, that are do not fit into control theory. We expect that current verification tools will need to be extended and new types of tools will need to be created to provide

- Multiple Aspects of Verification: The verification space consists of multiple aspects and can be quite heterogeneous. Thus, a diverse set of verification methods and tools will be needed, including domain-specific static analyses to verify algorithmic/control properties, extended model-checkers to verify state-related safety properties, constraint solvers and proof systems to prove bounded behavior and parameter ranges within desired envelopes. It is also noted that the verification space can also be quite heterogeneous; i.e., even within the same aspect model of component, not all parts need to be verified to the same degree or using the same method, depending upon what type of property that needs to be verified.

- Composing the Certification Argument: Just like the system design is composed of architectural layers and multiple components interacting over them, so does the certification argument needs to be composed with constituent verification artifacts. These verification artifacts will be produced by the domain-specific verification tools for the various aspects of the components that flow into the certification argument. There will need to be tool support and a “proof” capability to compose the argument – which will be an extension/generalization of the “assume-guarantee” types of compositions that are currently in practice.

Honeywell, in collaboration with SRI International, has started working on some of these verification challenges in NASA’s Verification and Validation of Flight Critical Systems (VVFCS) program. We hope to share the results of these efforts in the future.