Use of Formal Methods to Satisfy DO-178C Certification Requirements

AFRL Safe & Secure Systems & Software Symposium (S5)
June 15, 2010

Darren Cofer

Rockwell Collins

Rockwell Collins

Outline

- Certification and avionics SW
- DO-178B → DO-178C
- Formal methods in DO-178C process

Definition

- Certification: Legal recognition by the certification authority that a product, service, organization or person complies with the requirements.
  - Type certification: design complies with standards to demonstrate adequate safety
  - Product conforms to certified type design
  - Certificate issued to document conformance
- Example
  - We used verification tool X to accomplish these objectives.
  - These are the reasons why we think the tool is acceptable.
  - We ran 1000 tests using the tool, and this is why we think these 1000 tests are sufficient.
  - And, incidentally, here are the test results.

Convincing the relevant Certification Authority that all required steps have been taken to ensure the safety/reliability/integrity of the system

DO-178B

- "Software Considerations in Airborne Systems"
  - Certification authorities agree that an applicant can use guidance as a means of compliance (but not the only means) with regulations governing aircraft certification
- History
  - DO-178 (1982) – conceptual, "best practices"
  - DO-178A (1985) – 3 levels, development & verification processes described
  - DO-178B (1992) – 5 levels, objectives/activities/evidence

DO-178C?

- RTCA & EUROCAE issue Terms of Reference governing update to DO-178B (2005)
  - minimize changes to core document, yet…
  - update to accommodate 15+ years of SW experience
- Strategy: Address new technologies in “supplements”
  - OO, FM, MBD
  - Also tool qualification
- Air/ground synergy?
  - DO-278
- Rationale, consolidation, issues, errata
  - DO-248

SC-205: The Game

- Players
  - Anyone can play!
  - But you must be interested enough show up
  - All players are equal (but some are more equal than others)
- Rules
  - Form teams (subgroups)
  - Propose text (Information Papers)
  - Write and resolve comments
  - Vote in plenary sessions
  - Achieve consensus: "I can live with it."
  - Watch out for illegal moves
    - "I'm just here to learn."
    - "Let me tell you about our product."
- Strategy
  - Don't raise the bar
  - Don't lower the bar
- Winning the Game
  - FAA issues advisory circular

DO-178C: Overview

- Objectives: New guidance for emerging SW trends and technologies, document rationale, address gaps/inconsistencies
- Structure: Minimize changes to core document
  - Maintain current objective-based, technology-neutral approach
  - *Supplements* to address specific technologies
  - Coordination with DO-278, but no merger of air/ground guidance
- Tool Qualification supplement:
  - More detailed, stand-alone document, complete with objectives
  - 3 tool criteria (1 = development, 3 = verification, 2 = something in between)
  - [criteria] + [SW level] => tool qual. level (1-5) => required objectives
- Model-Based Development supplement:
  - Model = HLR/LLR/SW architecture
  - New guidance for model execution/simulation
  - Model coverage requirements
- Object-Oriented Design supplement:
  - Additional requirements for unique aspects of OO software
- Formal Methods supplement:
  - Facilitate applicant/certifier communication (definitions, expected evidence)
  - Define new objectives/activities/documentation (abstractions, assumptions)
  - Avoid common errors (false hypotheses)

Survey

- Which Technology Supplement will be the most difficult to produce?
  - Tool Qualification
  - Model-Based Development
  - Object Oriented Software
  - Formal Methods

Survey

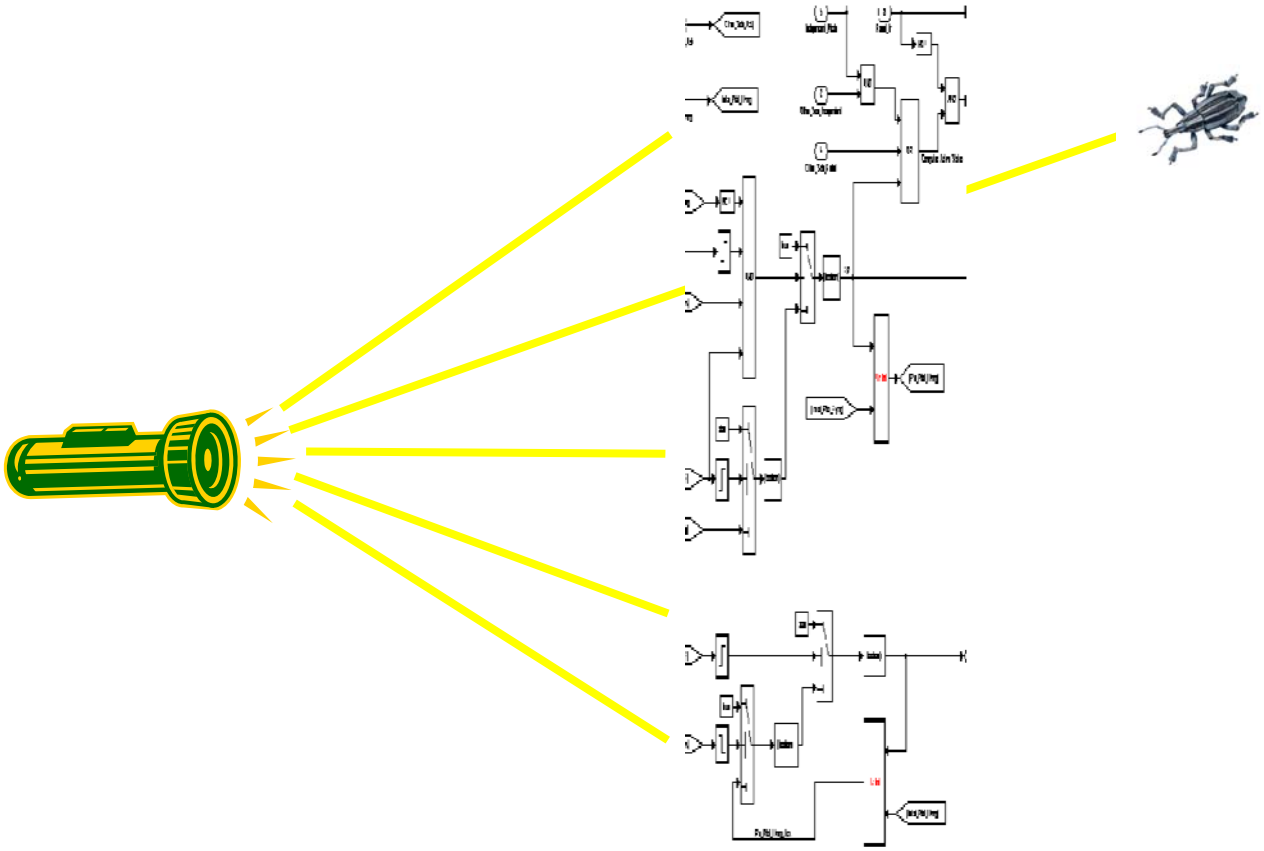
- Which Technology Supplement will be the most difficult to produce?
  - Tool Qualification
  - Model-Based Development
  - Object Oriented Software
  - Formal Methods

Verification in DO-178B

- Verification = review + analysis + test
- Testing of airborne software has two complementary objectives.
  - One objective is to demonstrate that the software satisfies its requirements.
  - The second objective is to demonstrate with a high degree of confidence that errors which could lead to unacceptable failure conditions, as determined by the system safety assessment process, have been removed.
- Formal methods can be used to meet these goals
  - Sometimes better

Advantage of formal methods

Testing checks only the input sequences we select

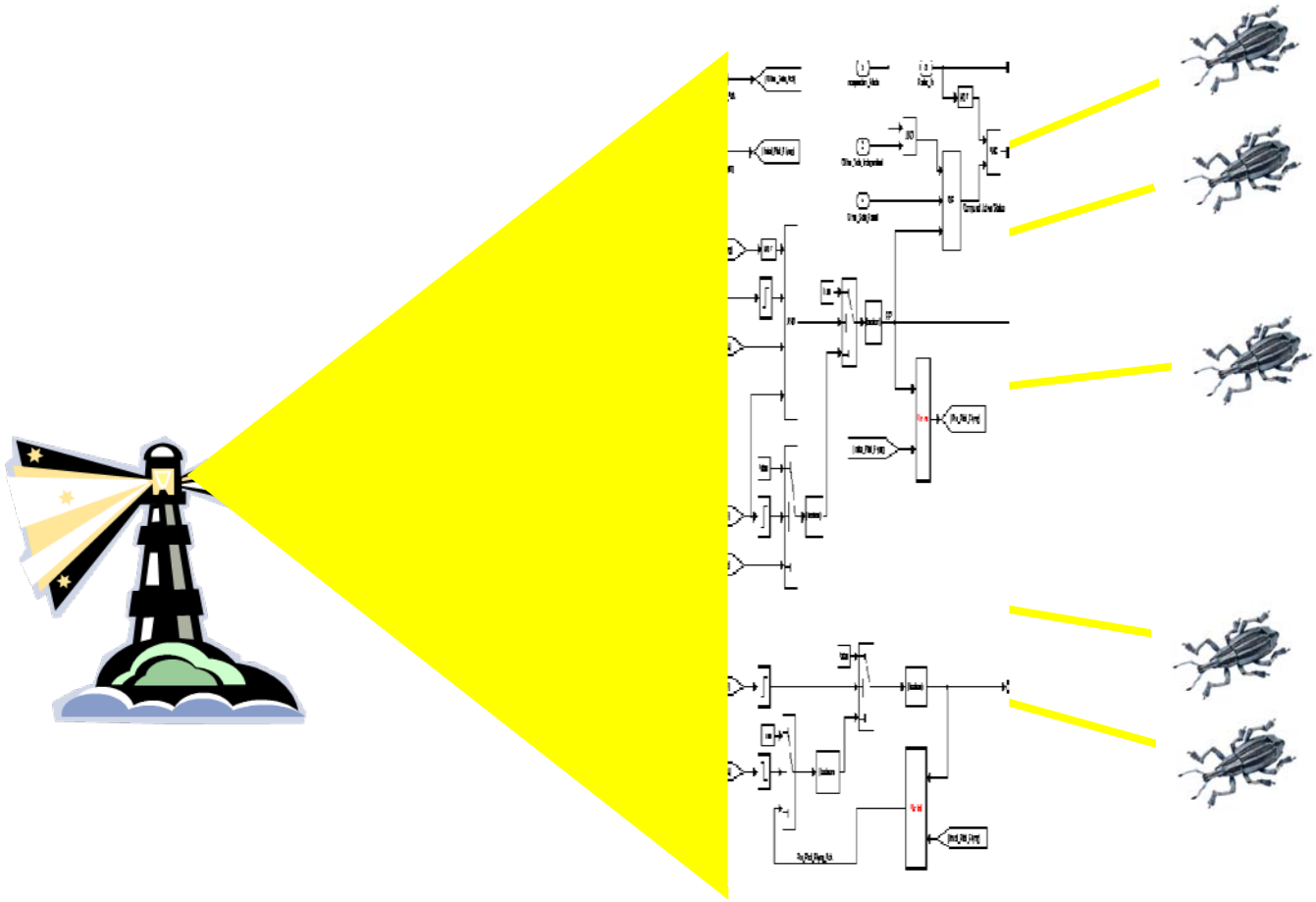


Even Small Systems Have Trillions (of Trillions) of Possible Tests!

Advantage of formal methods

Model checker explores every possible input & state

High degree of automation



Why use Formal Methods?

- Reduce cost
  - Early detection/elimination of defects
  - Evidence…
- Increase confidence
  - Proof of requirements, requirements satisfied
  - Absence of run-time errors
- Satisfy certification objectives
  - DO-178C

Objectives of Formal Methods subgroup (SG-6)

- Identify scope of FM
  - No longer an “alternate method”
  - What objectives can be satisfied and how
  - Focus is on verification (DO-178 section 6)
  - Partial use is OK
- Facilitate communication between applicants and certification authorities
  - What evidence should be expected for satisfying objectives
  - What new process documentation is needed
  - What additional/different activities are needed
- Identify areas deserving of particular scrutiny when FM are used
  - Help to avoid common errors
  - What questions should be asked
- Facilitate use of FM in aerospace community
  - Don’t impose higher burdens than traditional processes
  - But still keep the “bar” high enough

Section 1 – INTRODUCTION

- Characteristics of FM
  - Formal notation/model + formal analysis
  - use may be limited to particular requirements, systems, activities
- Formal modeling
  - unambiguous, mathematically defined syntax and semantics
- Formal analysis
  - guarantee *properties* of software systems (requirements)
  - emphasis on *soundness* of method
- Emphasis on FM as verification technique
  - Provide guidance for using formal methods to meet verification objectives of DO-178, either fully, or in conjunction with additional verification activities, including suitable complementary testing on the target hardware

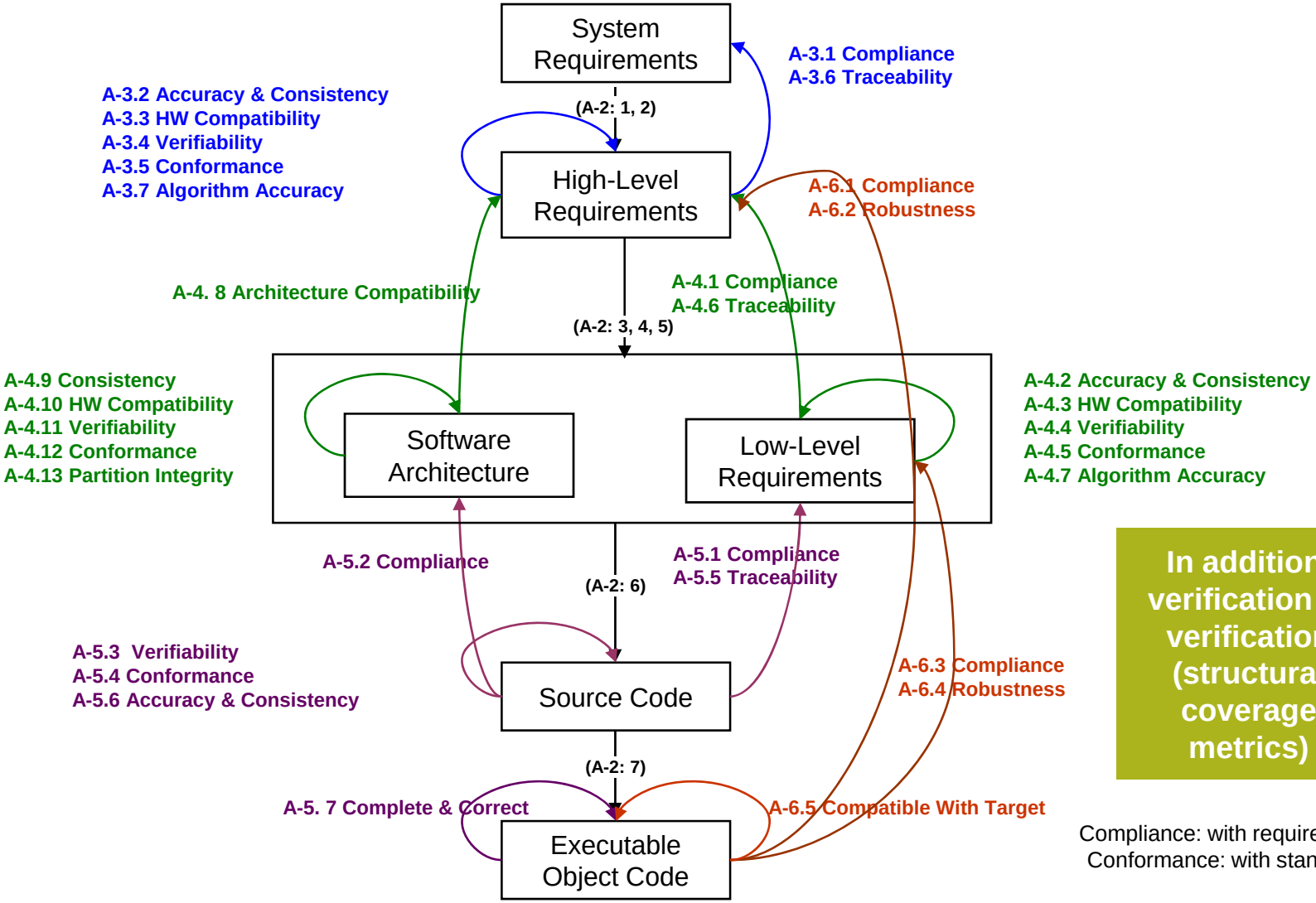
Section 4 – SOFTWARE PLANNING PROCESS

- PSAC will identify how FM will be used and what evidence will be provided
- SDP will identify development steps using FM
- SVP will identify:
  - analysis methods
  - assumptions and impact
  - how verification techniques will be combined to meet objectives

Section 6 – SOFTWARE VERIFICATION PROCESS

- Focus of FM guidance is on Verification Process
- General guidance:
  - All *formal notations* used must have unambiguous, mathematically defined syntax and semantics.
  - The soundness of each *formal analysis* method should be documented. A sound method never asserts that a *property* is true when it may not be true.
  - All assumptions related to the *formal analysis* should be described and justified (e.g. those associated with the target computer, or those about the data range limits).

DO-178B/C verification process (at a glance)



In addition, verification of verification (structural coverage metrics)

Compliance: with requirements
Conformance: with standards

Section 6 – SOFTWARE VERIFICATION PROCESS

Compliance: If the life cycle data items that comprise the inputs and outputs of a software development process are formally modeled, then *formal analysis* can be used to verify compliance. Compliance can be demonstrated by showing that the output satisfies the input. *Formal methods* cannot show that a derived requirement is correctly defined and has a reason for existing; this must be achieved by review.

Accuracy: *Formal notations* are by definition precise and unambiguous, and can be used to demonstrate accuracy of the representation of a life cycle data item.

Consistency: Life cycle data items that are formally expressed can be checked for consistency (the absence of conflicts). If a set of formal statements is found to be logically inconsistent then the inconsistencies must be addressed before any subsequent analysis is conducted.

Compatibility with the target computer: *Formal analysis* can be used to detect potential conflicts between a formal description of the target computer and the life cycle data item.

Verifiability: Being able to express a requirement in the *formal notation* defined in the software verification plan is evidence of verifiability in the same way as being able to define a test case.

Note: *Requirements involving "always/never" cannot in general be verified by a finite set of test cases, but may be verified by formal analysis.*

Conformance to standards: Life cycle data items that are formally expressed must be compliant with any standards defining the formalism.

Note: *Invalid results will be obtained if ill-formed requirements are allowed. Since formal notations have by definition a well-defined syntax, automated syntax checkers are appropriate for verifying that the formally stated requirements are well-formed with respect to syntax.*

Traceability: Traceability ensures that all input requirements have been developed into lower level requirements or source code. Traceability from the inputs of a process to its outputs can be demonstrated by verifying with a *formal analysis* that the outputs of the process fully satisfy its inputs. Traceability from the outputs of a process to its inputs can be demonstrated by verifying with a *formal analysis* that each output data item is necessary to satisfy some input data item.

Note: *Where output data items have been logically derived from input data items by refinement, traceability becomes implicit.*

Algorithm aspects: If life cycle data items are formally modeled, then algorithmic aspects can be checked using *formal analysis*.

Requirement formalization correctness: If a requirement has been translated to a formal notation as the basis for using a formal analysis, then review or analysis should be used to demonstrate that the formal statement is a conservative representation of the informal requirement.

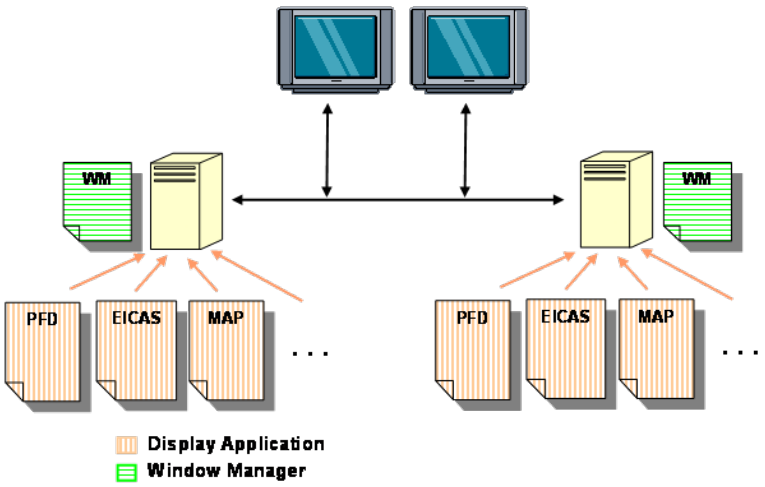
Note: *If the gap between an informal statement of the requirement and its embodiment in a formal notation is too large then this may be difficult to review. The preciseness of formal notations is only an advantage when they maintain fidelity to the intent of the informal requirement.*

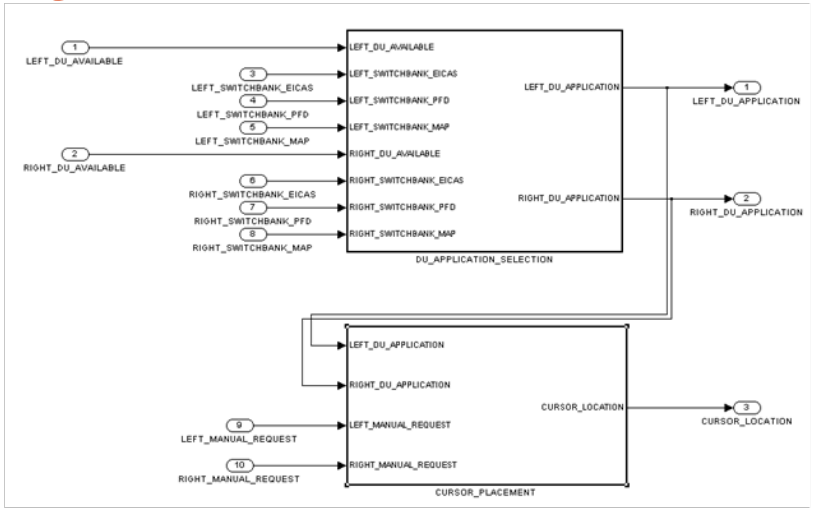
new objective

Section 6 – SOFTWARE VERIFICATION PROCESS

- Formal Analysis of Executable Object Code
  - In DO-178B this is accomplished by testing, and is one of the main objectives of the verification process
  - Requirements-based test + structural coverage measurements
  - DO-178C allows some of this to be replaced by analysis, but some testing must still be performed (HW/SW integration)
  - Based on Airbus A380 experience
- Principles
  - Complete coverage of each requirement
  - Completeness of the set of requirements
  - Detection of unexpected dataflow relationships (dependencies)
  - Detection of dead/deactivated code

Example: Window Manager SW





Subsystem	Simulink Diagrams	Simulink Blocks	State Space	Properties	Errors found
GG	2,831	10,669	9.8×10^{9}	43	56
PS	144	398	4.6×10^{23}	152	10
CM	139	1,009	1.2×10^{17}	169	10
DUF	879	2941	1.5×10^{37}	115	8
MFD	302	1,100	6.8×10^{31}	84	14
Totals	4295	16,117	n/a	563	98

Certification credit?

- Process
  - HLRs are initially expressed as English “shall” statements that are subsequently formalized for analysis.
  - LLRs are software models developed using model-based design tools (Simulink and Stateflow).
  - The LLR models are analyzed using a model checker to verify whether or not they satisfy the HLRs.
  - Source code is automatically generated from the LLRs and tested in conformance with a conventional test-based process.
- What DO-178C objectives could be satisfied using the Formal Methods Supplement?

FM for Certification

FM6.2 Software Verification Process Activities

a. **Formal notations**: Properties to be verified were specified in CTL. Formal definition of CTL may be found in [Em90] and [Hu04]. The models analyzed were specified in Simulink and Stateflow. These models were given formal definition through the translation process, which includes a formal syntax and translation rules for each model element.

b. **Soundness**: The BDD and SAT algorithms are known to be sound. Details of the BDD algorithm used for model checking and its soundness can be found in [Mc93] and [Hu04]. Application of satisfiability solving to the model checking problem and its soundness is described in [CBRZ01].

c. **Assumptions**: Any assumptions necessary for the analysis are justified, as described in Section 4.2.6.

FM for Certification

FM6.3 Software Reviews and Analysis

i. **Requirement formalization correctness**: In this project, all requirements were captured and managed using the DOORS tool. For each requirement, the corresponding formalization was captured in DOORS with one or more CTL statements. Independent reviews were conducted to ensure that the CTL statements accurately described the original English-language requirement.

FM6.3.1 Reviews and Analyses of the High-Level Requirements

d. **Verifiability of HLR**: The ability to express the high-level requirements for the system in CTL is a sufficient demonstration of verifiability in this example.

e. **Conformance to standards**: Requirements that do not conform to the standard for CTL syntax will be identified and rejected by the analysis tools. This feature of the tool would need to be qualified. Alternatively, conformance to CTL syntax can be easily checked by a manual review.

FM6.3.2 Reviews and Analyses of the Low-Level Requirements

a. **Compliance with HLR**: Analysis by model checking demonstrated that low-level requirements (the system model) complied with high-level requirements. This feature of the tool would need to be qualified.

Conclusion

- SC-205 is updating DO-178B → DO-178C
  - Next meeting: Marseilles (next week)
  - Final meeting for approval vote scheduled for Nov 2010
  - Original completion date: 12/2008
- DO-178C will include guidance for using formal methods to satisfy certification objectives
  - No longer an “alternate method”
- Formal methods can
  - Reduce cost through early detection/elimination of defects
  - Improve safety of complex software
  - Provide certification credit