Challenges Integrating Formal Methods into Certification of Critical Software & Systems

Michael Durling, Technical Manager – High Assurance Systems
GE Research
May 1, 2019
My experience developing a tool intended for use on aerospace applications that relies on formal methods

- Introduction to GE Research
- Original goals of our research program in 2013
- Experience and progress to date
- Lessons learned
- Questions
“I find out what the world needs, then I proceed to invent it.”

-Thomas Edison
GE ... unmatched innovation legacy & scale

Discovery ... world's firsts

1879 ELECTRIC LAMP
1896 X-RAY MACHINE
1941 US JET ENGINE
1998 MULTIPLE IMAGE CT

Portfolio ... differentiated IP

✓ Top 10 global patents ... 63,000+
✓ Broad applicability ... cross disciplinary
✓ Market tested ... product deployed

Prosperity ... ecosystem impact

Generating
1/3 of world’s electricity

Powering
Takeoff every 2 seconds

Curing
16,000+ scans every minute

Building on a century of breakthroughs
GE Global Research ... no one has what we have

Capabilities
- Artificial Intelligence
- Biology & Physics
- Controls & Optimization
- Edge Computing
- Electronics
- Electric Power
- Functional Materials
- Mechanics & Design
- Software & Analytics
- Structural Materials
- Thermosciences

600+ PhDs
Unmatched interdisciplinary expertise

Breakouts

SHAPE BUSINESSES
- 3 Cent Wind
- 65% CC GT
- Additive
- Renewables Full Time

SPAN INDUSTRIES
- Castings
- Plug & Play MR
- Renewable Reservoir
- Hybrid - Electric Flight

CREATE THE FUTURE
- Additive
- Artificial Intelligence
- CMC
- Robotics
- Controls/Edge
- Digital Twin
- Materials
- SiC
- Autonomous Imaging
- Bioelectronics
- High Assurance Systems
- Mission Unbreakable
- Trusted Autonomy

Scientific depth & breadth ... delivering real economics

© 2018, General Electric Company. All Rights Reserved.
GE Research High Assurance Systems Team

**Our Mission:** Create technology, tools and process that dramatically changes the cost and cycle time to develop and deploy critical systems software.

**Core Technologies:** Requirements modeling (structured natural language) and analysis (automated theorem proving), automated test case generation (model checkers), compositional verification (SMT Solvers) and synthesis, assurance case based verification

**Accomplishments & Experience:** Developed ASSERT™ toolchain, Developed fault tree modeling & synthesis tool for NASA, 2017 AFRL Summer of Innovation - applied formal verification to UxAS code, participating in DARPA Cyber Assured Systems Engineering (CASE) program – Team Enabling Architecture for Manned-Unmanned Systems (TEAMS). see references in notes page

**Next Steps:** Develop technical solutions that improve efficiency of development and certification high assurance systems. Exploring ideas to formalize and automate assurance cases that include continuous and run time assurance to support autonomy and untrusted components. Provide analysis and corrective action services.
Original motivation & goals from 2012
Software Development Trends (2013)

- Scale and complexity of software is growing in critical systems
- Traditional development methods are becoming unaffordable
- Desire to reuse legacy code
- Cyber security is a growing threat
- Trust of machine learning and artificial intelligence is an emerging challenge

Embedded software scale & complexity growing in critical systems applications
New technology, tools and process to enable next generation of systems

© 2019, General Electric Company
Level of Effort & Potential Benefit Opportunities

**Cost Increases**
1. HLR Modeling

**Cost Benefits**
1. Automated Test Case Generation
2. Automated Code Generation
3. Test Case and Analysis Automation
4. Reduced Rework
Our Process Map

1. System Requirements allocated to Software (Text)
2. Specification Model
3. Customer Model Based Validation
4. Formal Requirements Analysis
5. Requirements Based Test Cases (Auto Generated)
6. Design Model
7. Formal Design Verification
8. Model Coverage Analysis
9. Model Coverage Test Cases (Auto Generated)
10. Test Based Design Model Verification
11. Source Code (Auto Generated)
12. Formal Code Verification
13. Executable Object Code (EOC)
14. Test Based EOC Verification
## Alignment of Technology to Benefits

<table>
<thead>
<tr>
<th>Process Step</th>
<th>Technology</th>
<th>Benefit</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Develop Code</td>
<td>Design Modeling &amp; auto code generation</td>
<td>Reduce coding cost by 50%, overall process 10%</td>
<td>Low technology risk, medium application risk</td>
</tr>
<tr>
<td>LLR Test Case</td>
<td>Automated test case generation &amp; formal analysis</td>
<td>Reduce LLR test case gen by 75 %, overall process 12-17%</td>
<td>Low/Medium technology risk, medium application risk</td>
</tr>
<tr>
<td>High Level</td>
<td>Specification Modeling &amp; Formal Analysis</td>
<td>Increased overall process by 2%, identifies and correct errors early – should reduce rework for net 7%</td>
<td>Medium/High technology and application risk</td>
</tr>
<tr>
<td>HLR Test Case</td>
<td>Automated test case generation &amp; formal analysis</td>
<td>Reduce HLR test case gen by 75%, overall process by 10-15%</td>
<td>Medium/High technology and application risk</td>
</tr>
</tbody>
</table>

### Cost Increases
- 1. HLR Modeling
- 2. Automated Test Case Generation
- 3. Test Case and Analysis Automation
- 4. Reduced Rework

### Cost Benefits
- Automated Test Case Generation
Our Technology Program

**Approach**
- Develop, demonstrate and transition technology and capability to efficiently produce certified aerospace software
- Model based development

**Benefits**
- Identify & resolve problems earlier
- Efficient validation and verification

**Our technical focus**
- Requirements & design modeling
- Test suite generation
- Formal methods

**Tasks, Deliverables & TRL’s**
- Specification Model Design Practice, IP Landscape, Auto Test Generation Algorithms
- Pilot demonstrations of requirements modeling, auto test suite generation, requirements model checking & static code analysis

Model Based, Early Cycle & Automated Validation & Verification
Experience and progress to date
Our Approach

➢ Based on DO-178C & supplements DO-333 (FM) & DO-331 (MBD)
➢ Team Structure
  • GE Research
  • GE Aviation
    – Chief Engineers Office (CEO)
    – Product Line Engineers and Management
    – Certification Team
    – Tools Organization
  • Academia (Professor Panagiotos Manolios, Northeastern)
  • Government Agencies
➢ Execution Plan
  • Develop Technology & Tools
  • Pilot with Product Teams
  • Transition Process & Best Practices Through CEO
  • Transition Technology Through Tools Organization
Alignment with certification process

- RTCA DO-178C: Software Considerations in Airborne Systems and Equipment Certification
- RTCA DO-330: Software Tool Qualification Considerations
- RTCA DO-331: Model-Based Development and Verification Supplement to DO-178C and DO-278A
- RTCA DO-332: Object-Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A
- RTCA DO-333: Formal Methods Supplement to DO-178C and DO-278A
Key technologies to address quality & efficiency

**Specification Modeling**
- Behavioral + formal properties
- Aligned with DO-178C, DO-331 (MBD)
- Identify and remove errors early

**Formal Validation & Verification**
- Model Checking, theorem proving, static analysis
- Aligned with DO-178C, DO-333 (FM)
- Identify and remove errors early
- Reduce dependency on manual reviews & testing
- Requires formal specification model

**Automated test case generation & execution**
- Generate test cases from specification models
- Apply to design models & code
- Aligned with DO-178C, DO-331 (MBD)
- Reduce dependency on manual test case generation and execution
- Requires specification model

Develop & refine specification models early – leverage them throughout process
### Examples from our early certification analysis

<table>
<thead>
<tr>
<th>Objective</th>
<th>Applicability by Software Level</th>
<th>Artifacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>Description</td>
<td>A  B  C  D</td>
</tr>
<tr>
<td>MB.6.3.1.b</td>
<td>High-level requirements are accurate and consistent.</td>
<td>✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MB.6.3.1.d</td>
<td>High-level requirements are verifiable.</td>
<td>✓ ✓ ✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Specification Model formally models functional and safety-related requirements, including the pass/fail criteria. The Specification Model is also used for Formal Requirements Analysis to check that high-level requirements are accurate and consistent. The Specification Model TDR will include a checklist item to verify all criteria have passed.

The Specification Model formally models functional and safety-related requirements declaratively and clearly identifies the monitored and controlled variables. Each high-level requirement has its inputs and outputs identified and the pass/fail criteria are verifiable. The Specification Model TDR will include checklist items to verify that the Specification Model conforms to the Software Modeling Standard and that all criteria have passed.

Legend: ✓ Objective satisfied by Specification Model  ❌ Objective not satisfied by Specification Model  ■ Objective not applied
## Examples from our early certification analysis

<table>
<thead>
<tr>
<th>Objective</th>
<th>Applicability by Software Level</th>
<th>Artifact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test coverage of high-level requirements is achieved.</td>
<td>✓ ✓ ✓ ✓</td>
<td>Software Verification Results (MB.11.14)</td>
</tr>
</tbody>
</table>

**A Specification Model is developed to represent High-Level Requirements. Requirements based test cases and procedures are auto-generated from the Specification Model. The test procedures are first applied to the Design Models in the SCADE test environment, then later in the process on the executable object code. A test coverage report of the high-level requirements is generated and provided as artifact.**

<table>
<thead>
<tr>
<th>Objective</th>
<th>Applicability by Software Level</th>
<th>Artifact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test coverage of low-level requirements is achieved.</td>
<td>✓ ✓ ✓ ✓</td>
<td>Software Verification Results (MB.11.14)</td>
</tr>
</tbody>
</table>

**Derived requirements in the Design Model are identified. Test cases and procedures for derived requirements are auto-generated from the Design Model. The test procedures are first applied to the Design Models in the SCADE test environment, then later in the process on the executable object code. A test coverage report of the low-level requirements is generated and provided as artifact.**

---

**Legend:**
- ✓ Objective satisfied by ATCG
- ✓ Objective not satisfied by ATCG
- Objective not applied

<table>
<thead>
<tr>
<th>Reference</th>
<th>Description</th>
<th>Applicability by Software Level</th>
<th>Artifact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test coverage of software structure (decision coverage) is achieved.</td>
<td>✓ ✓ ✓</td>
<td>Software Verification Results (MB.11.14)</td>
<td></td>
</tr>
</tbody>
</table>

**Start with requirements based testing coverage data in the SCADE Model Test Coverage (MTC) testing environment – Identify Design Model elements that were not covered by requirements based tests. Analyze uncovered model elements to identify the reason for lack of coverage. The reason may be an error in the requirements based test case, an error in the Specification Model, derived requirements, or an error in the Design Model such as dead or unintended code. Auto generate test cases and procedures from Design Model for derived requirements. Apply the test cases and procedures to the Design Model using the SCADE MTC tool to verify decision coverage. For cases other that derived requirements, use the ATCG tool to generate test cases and procedures to help the engineer understand the design and appropriately correct the error. Once the error is resolved, execute the test case and procedure on the Design Model in the SCADE MTC tool to verify decision coverage. Translate the test cases and procedures to the appropriate format for use on the Instrumented (via LDRA or other tool) executable object code. Apply test cases and procedures to Instrumented EOC to verify structural coverage. Identify errors and iterate until structural coverage is satisfied.**

**Legend:**
- ✓ Objective satisfied by Design Model
- ✓ Objective not satisfied by Design Model
- Objective not applied
Tool Qualification Levels from RTCA DO-178C 12.2.2

Criteria 1: A tool whose output is part of the airborne software and thus could insert an error

Criteria 2: A tool that automates verification process(es) and thus could fail to detect an error, and whose output is used to justify the elimination or reduction of:

1. Verification process(es) other than that automated by the tool, or
2. Development process(es) that could have an impact on the airborne software

Criteria 3: A tool that, within the scope of its intended use, could fail to detect an error.

<table>
<thead>
<tr>
<th>Software Level</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>TQL-1</td>
</tr>
<tr>
<td>B</td>
<td>TQL-2</td>
</tr>
<tr>
<td>C</td>
<td>TQL-3</td>
</tr>
<tr>
<td>D</td>
<td>TQL-4</td>
</tr>
</tbody>
</table>

Qualification of a tool is needed when process(es) are eliminated, reduced or automated by use of the software tool without its output being verified ...

Source: RTCA DO-178C
Focus on human and machine readable structured English requirements - formal analysis to check for requirements errors, auto generate requirements based tests
Illustrative example
Overall Process

- **Generate Reqts in DOORS**
- **Use DXL Script to Export Functional Requirements to RCE (Eclipse)**
- **Use RCE Error Checking to Identify and Fix Errors (manually modified in DOORS and re-exported)**
- **Run RAE to Validate Requirements (completeness, independence, conflict, contingency)**
- **Change Reqts to Resolve Errors Raised by RAE (manually modified in DOORS and re-exported)**

**Requirements Capture**

**Link to Requirements**

- **Execute Test Scripts on Test Bench**
- **Use DXL Script to Import Test Cases into DOORS, ITCs include TP refs and DXL creates links from TCs to Reqts**
- **Generate Test Scripts with ATPT**
- **Generate Test Procedures with ATPG**
- **Generate Test Cases with ATCG**

**Requirements Analysis & Validation**

**Translation to Test Equipment (ATPT)**

**Automated Test Procedure Generation**

**Automated Test Case Generation**
### Requirements in DOORS

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirements for the Braking System Example</th>
<th>Requirement</th>
<th>Comment</th>
<th>Ignore in SADL</th>
<th>Inlinks (All modules)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS-REQ-1</td>
<td><strong>1 BSCU_Mode_Command</strong></td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYS-REQ-2</td>
<td>import &quot;http://Ontology_BSCU/BSCU&quot;</td>
<td>Context: BSCU and Hydraulic_Hysteresis is 2.</td>
<td>Context: Time units for Hysteresis are &quot;seconds&quot;. BSCU with pilot_controls PILOT_CONTROLS and BSCU with hyd_status HYD_STATUS and Hydraulic_Hysteresis is 2.</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>SYS-REQ-3</td>
<td>BSCU shall set mode of BSCU to Normal when (fall of Hyd_1 has been Not_Asserted for the past Hydraulic_Hysteresis and BSCU_mode_command of PILOT_CONTROLS is Normal).</td>
<td>Derived</td>
<td>Derived: Set BSCU mode to Normal when pilot selects Normal and Hyd 1 is not failed for the duration of hysteresis.</td>
<td>False</td>
<td>SYS-TC-Mode-90</td>
</tr>
<tr>
<td>SYS-REQ-4</td>
<td>BSCU shall set mode of BSCU to Alternate when (BSCU_mode_command of PILOT_CONTROLS is Alternate and fail of Hyd_2 has been Not_Asserted for the past Hydraulic_Hysteresis) or (BSCU_mode_command of PILOT_CONTROLS is Normal and fail of Hyd_1 was asserted in the past Hydraulic_Hysteresis and fail of Hyd_2 has been Not_Asserted for the past Hydraulic_Hysteresis).</td>
<td>Derived</td>
<td>Derived: Set BSCU mode to Alternate when pilot selects Alternate or Hyd 1 is failed while Hyd 2 has not failed for the duration of hysteresis.</td>
<td>False</td>
<td>SYS-TC-Mode-94</td>
</tr>
<tr>
<td>SYS-REQ-5</td>
<td>BSCU shall set mode of BSCU to Emergency when (fall of Hyd_1 has been Not_Asserted for the past Hydraulic_Hysteresis and fail of Hyd_2 was Asserted in the past Hydraulic_Hysteresis and BSCU_mode_command of PILOT_CONTROLS is Alternate) or (fall of Hyd_1 was Asserted in the past Hydraulic_Hysteresis and fail of Hyd_2 was Asserted in the past Hydraulic_Hysteresis) or (BSCU_mode_command of PILOT_CONTROLS is Emergency).</td>
<td>Derived</td>
<td>Derived: Set BSCU mode to Emergency when pilot selects Emergency or Hyd 1 and Hyd 2 have both failed within the hysteresis (or more).</td>
<td>False</td>
<td>SYS-TC-Mode-102</td>
</tr>
<tr>
<td>SYS-REQ-6</td>
<td>BSCU shall set fail of HYD_STATUS to Asserted when pressure of HYD_STATUS &lt; threshold of HYD_STATUS.</td>
<td>Derived</td>
<td>Set Hyd 1/2 to fail when Hyd 1/2 pressure is below threshold.</td>
<td>False</td>
<td>SYS-TC-Mode-111</td>
</tr>
<tr>
<td>SYS-REQ-7</td>
<td>BSCU shall set fail of HYD_STATUS to Not_Asserted when pressure of HYD_STATUS &gt;= threshold of HYD_STATUS.</td>
<td>Derived</td>
<td>Set Hyd 1/2 to not fail when Hyd 1/2 pressure is greater than or equal to threshold.</td>
<td>False</td>
<td>SYS-TC-Mode-130</td>
</tr>
</tbody>
</table>
Requirements in ASSERT™

url "http://Requirements_BSCU/SYS_REQ_1__BSCU_Mode_Command" alias SYS_REQ_1__BSCU_Mode_Command.
//ObjectId=SYS-REQ-2
//ParentId=SYS-REQ-1
import "http://Ontology_BSCU/BSCU".

©Context:

- BSCU and
- Hydraulic_Hysteresis is 2.
//EndObject

- Requirement SYS-REQ-3:
  BSCU shall set
  mode of BSCU to Normal
when (fail of Hyd_1 has been Not_Asserted for the past Hydraulic_Hysteresis and
  BSCU_mode_command of PILOT_CONTROLS is Normal).
//EndObject

- Requirement SYS-REQ-4:
  BSCU shall set
  mode of BSCU to Alternate
when (BSCU_mode_command of PILOT_CONTROLS is Alternate and
  fail of Hyd_1 has been Not_Asserted for the past Hydraulic_Hysteresis) or
  (BSCU_mode_command of PILOT_CONTROLS is Normal and
  fail of Hyd_1 was Asserted in the past Hydraulic_Hysteresis and
  fail of Hyd_2 has been Not_Asserted for the past Hydraulic_Hysteresis).
//EndObject
## Ontology in DOORS

<table>
<thead>
<tr>
<th>ID</th>
<th>Ontology for the BSCU Example</th>
<th>Object Type</th>
<th>Class</th>
<th>Cardinal</th>
<th>Max Cardinal</th>
<th>Instances</th>
<th>Physical Mapping</th>
<th>Input/Output</th>
<th>Comments</th>
<th>Units</th>
<th>Functional Range Max</th>
<th>Functional Range Min</th>
<th>Physical Range Max</th>
<th>Physical Range Min</th>
<th>Resolution</th>
<th>Tolerance</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS-OMT-52</td>
<td><strong>1.7 HYD_STATUS</strong></td>
<td>Class Definition</td>
<td>base</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYS-OMT-53</td>
<td></td>
<td>pressure</td>
<td>Property Definition</td>
<td>integer</td>
<td>1</td>
<td></td>
<td>Hyd_1</td>
<td>Input</td>
<td>hydraulic pressure</td>
<td>psi</td>
<td>4000</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SYS-OMT-67</td>
<td></td>
<td>threshold</td>
<td>Property Definition</td>
<td>integer</td>
<td>1</td>
<td></td>
<td>Hyd_2</td>
<td>Input</td>
<td>minimum hydraulic pressure</td>
<td>psi</td>
<td>3000</td>
<td>2000</td>
<td>2000</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SYS-OMT-68</td>
<td></td>
<td>fail</td>
<td>Property Definition</td>
<td>DISCRETE</td>
<td>_VALUE</td>
<td></td>
<td></td>
<td>Neither</td>
<td>failure when hydraulic pressure is below minimum hydraulic pressure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYS-OMT-55</td>
<td><strong>1.8 MONITOR</strong></td>
<td>Class Definition</td>
<td>base</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYS-OMT-56</td>
<td></td>
<td>Validity</td>
<td>Property Definition</td>
<td>VALIDITY</td>
<td>1</td>
<td></td>
<td></td>
<td>Neither</td>
<td>BSCU validity status</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYS-OMT-60</td>
<td><strong>1.9 COMMAND</strong></td>
<td>Class Definition</td>
<td>base</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYS-OMT-15</td>
<td></td>
<td>anti_skid_cmd</td>
<td>Property Definition</td>
<td>integer</td>
<td>1</td>
<td></td>
<td>Command_1</td>
<td>Output</td>
<td>anti-skid command</td>
<td>% of full range</td>
<td>100</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SYS-OMT-16</td>
<td></td>
<td>anti_skid_shutoff</td>
<td>Property Definition</td>
<td>integer</td>
<td>1</td>
<td></td>
<td>Command_2</td>
<td>Output</td>
<td>anti-skid shutoff</td>
<td>% of full range</td>
<td>100</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SYS-OMT-57</td>
<td><strong>1.10 TRUCK</strong></td>
<td>Class Definition</td>
<td>base</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYS-OMT-7</td>
<td></td>
<td>wheel_data</td>
<td>Property Definition</td>
<td>WHEEL_1YA</td>
<td>8</td>
<td></td>
<td></td>
<td>Output</td>
<td>interface with main landing gear trucks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Ontology in ASSERT™
Requirements Syntax Checking in ASSERT™

Type Checking
RCE type checking flags this as an error
• “failure” is not defined in our ontology
• User needs to change to “fail”
User needs to fix the error in DOORS
Completeness Error
SYS-REQ-6 and -7 also properly set fail indication based upon pressure relative to threshold
  • Except when pressure = threshold
RAE raises this as a Completeness error
  • Behavior not defined for counter-example state
  • pressure = threshold
User needs to fix the error in DOORS
Conflict Errors
SYS-REQ-6 and -7 also properly set fail indication based upon pressure relative to threshold
- Except when pressure = threshold
RAE raises this as a Conflict error
- Conflict identified with counter-example
  - pressure = threshold
User needs to fix the error in DOORS
Formal Analysis with ACL2s happens in the background

**Conflict Analysis.** A set of requirements conflict if there is some input value for the monitored variables from which the requirement cannot be satisfied, no matter what value is given to the controlled variables. When one set of requirements states that a controlled variable must have one value, while another set of requirements states that it must have a different value, we have a conflict.

**Analysis:** Check $\forall M :: \exists C :: R(M, C)$ where $R$ is the conjunction of all requirements and $M, C$ are the monitored and controlled variables.

**Error isolation:** Return concrete values for $M$ and a requirements core $R'$ such that the above check fails.

**Completeness Analysis.** A set of requirements is complete if they uniquely specify the value of the controlled variables, given concrete values for the monitored variables. For functional requirements, completeness is needed.

**Analysis:** Check $\forall M, C_1, C_2 :: R(M, C_1) \wedge R(M, C_1) :: C_1=C_2$ where $R$ is the conjunction of all requirements, $M$ are the monitored variables, and $C_1, C_2$ are two copies of the controlled variables.

**Error isolation:** Return concrete values for $M, C_1, C_2$ such that the above check fails.

Requirements Analysis function is used as a design aid – not a qualified tool
Automated Test Generation with ASSERT™

Test Strategy Summary

Selection of strategies is performed automatically and independently for each requirement

- Results tailored for content of each requirement

Main strategies

- Functional Completeness Analysis
- Logic Condition Analysis
- Equivalence Class Analysis
- Boundary Value Analysis

---

2. TEST STRATEGY

---

// This section defines the ATCG test strategy breakdown to verify the requirements.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS-REQ-3</td>
<td>X</td>
<td>X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SYS-REQ-4</td>
<td>X</td>
<td>X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SYS-REQ-5</td>
<td>X</td>
<td>X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SYS-REQ-6</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>X</td>
</tr>
<tr>
<td>SYS-REQ-7</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>X</td>
</tr>
</tbody>
</table>

(KEY: 'X'=fully covered, 'P'=limited coverage, '='=no coverage, 'N/A'=N/A, 'J'=justified)
Equivalence Class Analysis Summary

All combinations of low and high range of the two inputs are listed

- Requirement provides two input variables with unique low and high ranges
- Ranges defined in ontology
- Results in two equivalence classes

---

**Equivalence Class Table**

<table>
<thead>
<tr>
<th></th>
<th>EQP1</th>
<th>EQP2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mv1</td>
<td>SYS-REQ-6-TC003</td>
<td>SYS-REQ-7-TC003</td>
</tr>
<tr>
<td>Mv2</td>
<td>SYS-REQ-6-TC008</td>
<td>SYS-REQ-7-TC008</td>
</tr>
</tbody>
</table>

(KEY: 'TCxxx'-Test Case. 'N/A.'—no coverage. 'J'=justified)
Boundary Value Test Cases

All combinations of low and high range of the inputs and resolutions are listed

• Requirement provides two input variables with unique low and high ranges
• Ranges and resolutions defined in ontology
• Results in 6 boundary values for SYS-REQ-6
Test Procedure Generation

Boundary Value Analysis Test Procedures

Solver used to convert test cases into test procedures

Meta-data included that links each test step back to the test case it satisfies
Test Generation is based on SMT Solver technology.

Auto Test Generation has TQL – 5 Qualification Kit.
<table>
<thead>
<tr>
<th>ID</th>
<th>Test Case Description</th>
<th>Linked Test Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS-TC-Mode-111</td>
<td><strong>3.1.4 Requirement SYS-REQ-6</strong></td>
<td>SYS-REQ-6</td>
</tr>
</tbody>
</table>
| SYS-TC-Mode-112 | **3.1.4.1 SYS-REQ-6-TC001_0**  
Verify  
Fall of HYD_STATUS of hyd_status of BSCU is Asserted  
When  
purpose of HYD_STATUS of hyd_status of BSCU < threshold of HYD_STATUS of hyd_status of BSCU | SYS-REQ-6, Context 0  
SYS-REQ-6, Context 0  
SYS-REQ-6, Context 0  
SYS-REQ-6, Context 0 |
| SYS-TC-Mode-113 | **3.1.4.2 SYS-REQ-6-TC002_0**  
Verify  
Fall of HYD_STATUS of hyd_status of BSCU is Asserted  
When  
purpose of HYD_STATUS of hyd_status of BSCU < threshold of HYD_STATUS of hyd_status of BSCU | SYS-REQ-6, Context 0  
SYS-REQ-6, Context 0  
SYS-REQ-6, Context 0  
SYS-REQ-6, Context 0 |
| SYS-TC-Mode-114 | **3.1.4.3 SYS-REQ-6-TC003_0**  
Verify  
Fall of HYD_STATUS of hyd_status of BSCU is Asserted  
When  
(purpose of HYD_STATUS of hyd_status of BSCU < threshold of HYD_STATUS of hyd_status of BSCU) AND  
(purpose of HYD_STATUS of hyd_status of BSCU > 1.0) AND  
(purpose of HYD_STATUS of hyd_status of BSCU < 3999.0) | SYS-REQ-6, Context 0  
SYS-REQ-6, Context 0  
SYS-REQ-6, Context 0  
SYS-REQ-6, Context 0 |
| SYS-TC-Mode-115 | **3.1.4.4 SYS-REQ-6-TC004_0**  
Verify  
Fall of HYD_STATUS of hyd_status of BSCU is Asserted  
When  
(purpose of HYD_STATUS of hyd_status of BSCU < threshold of HYD_STATUS of hyd_status of BSCU) AND  
(purpose of HYD_STATUS of hyd_status of BSCU is 0.0) | SYS-REQ-6, Context 0  
SYS-REQ-6, Context 0  
SYS-REQ-6, Context 0  
SYS-REQ-6, Context 0 |
| SYS-TC-Mode-116 | **3.1.4.5 SYS-REQ-6-TC005_0**  
Verify  
Fall of HYD_STATUS of hyd_status of BSCU is Asserted  
When  
(purpose of HYD_STATUS of hyd_status of BSCU < threshold of HYD_STATUS of hyd_status of BSCU) AND  
(purpose of HYD_STATUS of hyd_status of BSCU is 1.0) | SYS-REQ-6, Context 0  
SYS-REQ-6, Context 0  
SYS-REQ-6, Context 0  
SYS-REQ-6, Context 0 |
Recent Publications


<table>
<thead>
<tr>
<th>Title</th>
<th>Patent Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method of Software Specification Modeling</td>
<td>US 9,747,079</td>
</tr>
<tr>
<td>Scalable Methods for Analyzing Formalized Requirements and Localizing Errors</td>
<td>US 9,639,450</td>
</tr>
<tr>
<td>System and Methods for Safety-Critical Software Architecture Design and Automated Requirements-based Test Case Generation</td>
<td>US 9,940,222</td>
</tr>
<tr>
<td>Coverage-based Automated Test Case Augmentation for Design Models</td>
<td>US 9,792,204</td>
</tr>
<tr>
<td>Methods and tools for mission planning via formal verification and supervisory controller synthesis</td>
<td>US 10,249,197</td>
</tr>
<tr>
<td>System and Method for Automatic Test Cases Generation from Software Specifications that Contain Nonlinear Arithmetic Constraints over Real Numbers</td>
<td>US 10,169,217</td>
</tr>
</tbody>
</table>
Lessons learned

- Formal methods are usable today to provide measurable benefits in quality, consistency and efficiency for developing safety critical software.

- Clarify your process, understand the projected impact, expected artifacts and certification implications – review them with your DER and certification authority early.

- Understand the product application, and the alignment of your tool. Use your tool where you are confident that it will work well even if it is only a portion of the overall project.

- Plan your team. Include application domain, tool development, technology and certification experts. Plan your transition from research to product.

- Have a strategy and goals – all programs face short term volatility – stay positive and long term focused.

- Use new technology as a design aid to build confidence and credibility.

- There is a gap in clear definition of quality requirements. Our tool helps clarity levels and structure.

- Usability is critical. Think about how your tool will be used early. Who will be using your tool? What artifacts will you create and where will they be used?
What’s next?

• Capture and analyze security requirements and policies
• Capture and analyze requirements for AI\ML applications
• Extend state based test generation capability
• Develop capability to automatically interpret legacy requirements\applications and map them into ASSERT™
• Explore opportunities with assurance case based certification approach
Questions?