Model-Based Testing of Spacecraft Flight Software

Maria Hernek
Virtual
12/09/2013
Objective:
To present the result and achievements of ESA study “Model Based Testing of flight SW” and discuss strengths and weaknesses of the method

Outline:
• ESA ISVV process short intro of applicable parts
• Background of ESA study
• Method and approach
• Test object
• Implementation
• Conclusion
Independent Software Verification & Validation (ISVV) by ESA

1. ISVV is required for Mission and Safety Critical software, (ECSS-E-40/ECSS-Q-80)

2. ISVV tasks are additional and complementary to the nominal SW supplier’s verification and validations tasks

3. ISVV tasks cover verification and validation of software requirements, design, code and tests (typically starting at SW-SRR and finishing before the SW-QR)

4. ISVV supplier is required to be an organization independent of the software supplier as well as the prime/system integrator (full technical, managerial, and financial independence)

5. Most ESA projects implement the ISVV process as an industrial contract placed by the Prime contractor
ESA ISVVV Process overview

- **6 activities/STAGES**: Management (MAN), Verification (IVE) and Validation (IVA)
- Activities are composed of **TASKS**, and these are further split into **SUBTASKS**

1. **Management** (MAN.PM and MAN.VV) is concerned with issues such as ISVV objectives and scope, planning, roles, responsibilities, budget, communication, competence, confidentiality, schedule and ISVV level definition (to limit the scope of ISVV)

2. **Technical Specification Analysis** (IVE.TA) is verification of the software requirements

3. **Design Analysis** (IVE.DA) is verification of the SW Architectural Design and the Software Detailed Design

4. **Code Analysis** (IVE.CA) is verification of the SW source code

5. **Validation** (IVA) is testing of the SW to demonstrate that the implementation meets the technical specification
Example of a Task/Subtask description

- **Activity**: Technical Specification Analysis
- **Task**: SW Requirements Verification
- **Subtasks**: T1.S1, T1.S2 ... T1.S11
- **Start/End Events**
- **Inputs/Outputs**
- **Methods** are identified for each subtask

### Some numbers:

- IVE.TA → 1 task → 11 subtasks
- IVE.DA → 3 tasks → 15/12/5 subtasks
- IVE.CA → 3 tasks → 10/5/3 subtasks
- IVA → 3 tasks → 3/3/3 subtasks

---

**Example of a Task/Subtask description**

<table>
<thead>
<tr>
<th>TASK DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title:</td>
</tr>
<tr>
<td>Task ID:</td>
</tr>
<tr>
<td>Activity:</td>
</tr>
<tr>
<td>Start event:</td>
</tr>
<tr>
<td>End event:</td>
</tr>
<tr>
<td>Responsible:</td>
</tr>
<tr>
<td>Objectives:</td>
</tr>
</tbody>
</table>

**Inputs**

- From the ISVV Customer:
  - System Requirements allocated to Software [RB; SRR]
  - Software Requirements Specification [TS; PDR]
  - Software Logical model [TS; PDR]
  - Interface Control Document [ICD(TS); PDR]
  - Hardware-Software Interface Requirements [RB; SRR]
  - Software Criticality Analysis Report [PAF; SRR]

- From the ISVV Supplier:
  - Critical Software Requirements List (refer to [MAN.VV.T2])
  - ISVV level definition (refer to MAN.VV.T2)

**Sub Tasks (per ISVV Level):**

- **ISVV Level 1 and Level 2**
  - IVE.TA.T1.S1: Verify the Software Requirements external consistency with the system requirements (by Inspection - reviewing the traceability matrices produced by the software supplier):
    Ensure that all system requirements allocated to software are traceable to software requirements (forward traceability).
    Ensure that every software requirement is traceable to a system requirement (backward traceability).
    Ensure that the relationships between the software requirements and their originating system requirements are specified in a uniform manner (in terms of level of detail and format).
    Ensure that the characteristics specified in the system requirements allocated to software are accurately specified by the traced software requirements.
  - IVE.TA.T1.S2: Verify the Interface Requirements external consistency with the system requirements (by Inspection - reviewing the traceability matrices produced by the software supplier and in case they do not exist, by producing them):
  - IVE.TA.T1.S11: Verify software requirements conformance with applicable standards (by Inspection)
    Ensure that the software requirements are compliant to applicable standards, references, regulations, policies, physical laws, and business rules.

**Outputs**

- Technical Specification Verification Report (including the ISVV findings)
- Contribution to Independent Validation
**TA.T1: Software Requirements Verification**

**Subtasks:** To verify

- **Software Requirements external consistency** with the system requirements
- **Interface Requirements external consistency** with the system requirements
- Software requirements **correctness**
- **consistent documentation** of the software requirements
- Software requirements **completeness**
- **dependability and safety** requirements
- **readability** of the software requirements
- **timing and sizing budgets** of the software requirements
- Identify **test areas and test cases for Independent Validation**
- that software requirements are **testable**
- Software requirements **conformance** with applicable standards
IVA: Independent Validation

Subtasks:

- Identification of Test Cases
- Construction of Test procedures
- Execution of Test Procedures
Background

- Software Verification & Validation important phase
- Software Validation phase – critical path
- Software Validation labor intensive
- Software versioning – impact on Validation – regression need

- Source of problem Validation? and/or Specification?
- More automation possible?
- Specification incomplete, incorrect
Method and Approach

- Sequence-Based Specification (SBS) combined with
- Statistical-Based Testing (SBT)

**Sequence-Based Specification**
- develop Black Box specification
- depends solely on external stimuli and responses

**Statistical-Based Testing**
- Test values randomly generated using a probabilistic profile
Approach

- Test lib
- Requirements
- TestScript: Java, php, python
- Test log
- SimOps
- SCOS
- SVF
- TC
- TM
- Test Object
Test Object

Spacecraft Flight Software –
Earth observation mission

Main functions:
- Thermal control
- Power control and distribution unit management
- Attitude and Orbit Control System and mode management
- Telemetry memory management
- Mass memory and formatting unit management
- On-Board time management
- Remote interface unit management

Functions suited for SBS Modeled
- State machine based
- Complex combination of input sequences
- Possible to identify stimuli, predicate and responses

Predicate represent a stimuli which can be expressed in form of regular expression rule
Test Object

Requirements metrics

FSW requirements

- Irrelevant
- Relevant and worth doing
- Relevant but too simple for SBS
- Relevant but only for stimuli response
- Relevant and selected

Values:
- 500
- 350
- 200
- 225
Sequence-Based Specification, steps

Model system’s state transition and responses according to all possible sequences of system inputs

Requirements

Sequence-based Specification

Mealy Machine

Annotated Mealy Machine

Test Model

Tool support for all steps
Sequence-Based Specification, steps

- Identify boundaries
  - Define interfaces
- Identify stimuli
- Identify responses
- Enumerate sequences of stimuli
  - Identify canonical sequences
- Define state variables
Statistical Based Testing, steps

Test Model

Abstract Test Cases

Generate abstract test cases covering transitions of Mealy Machine

Concrete Test Cases

Mapping abstract stimuli to test scripts libraries

What parameter and values of stimuli?

What parameter and values of predicate?

Where to read responses and expected values.

Code

Test Oracle

Test Results

Compare results with Oracle

Tool support:
JUMBL
CONVAC-TC
Test case criteria:

- cover every arc of Mealy Machine (every state is covered)
- weighted arcs used reflecting frequency, importance, risk of the likelihood of arc to happen
- Patch coverage, typically infinite number of paths but basis path coverage can be achieved
Tool support

1. SBS Super
   a. Protoseq with some additions, dedicated build:
      - Extended predicate support
      - More responses supported  Support dynamic sequence adjustment
      - Forward traceability
      - Interface smoothly with JUMBL

2. JUMBL
   a. By Software Quality Research Laboratory, SQRL
   b. Generate abstract tests, a path in a usage model

3. CONVAC TC
   a. Custom made tool
   b. Converts abstracts test cases to executable

Protoseq considered but not used
## Implementation, characteristics

<table>
<thead>
<tr>
<th>Function</th>
<th>Number of stimuli</th>
<th>Max length of seq</th>
<th>Number of predicates</th>
<th>Response type</th>
<th>Num of states</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermal cntrl</td>
<td>14</td>
<td>8</td>
<td>1</td>
<td>Simple</td>
<td>19</td>
</tr>
<tr>
<td>Power cntrl</td>
<td>30</td>
<td>2</td>
<td>2</td>
<td>Simple</td>
<td>4</td>
</tr>
<tr>
<td>AOCS mode mng</td>
<td>15</td>
<td>8</td>
<td>0</td>
<td>Simple</td>
<td>21</td>
</tr>
<tr>
<td>MMM</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>Complex</td>
<td>9</td>
</tr>
<tr>
<td>MMFU mng</td>
<td>11</td>
<td>4</td>
<td>0</td>
<td>Complex</td>
<td>5</td>
</tr>
<tr>
<td>OBT mng</td>
<td>12</td>
<td>6</td>
<td>0</td>
<td>Complex</td>
<td>6</td>
</tr>
</tbody>
</table>
## SBS, Effectiveness

<table>
<thead>
<tr>
<th>Function</th>
<th>hrs spent SBS models</th>
<th>hrs spent refining model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermal cntrl</td>
<td>18 (60)</td>
<td>16</td>
</tr>
<tr>
<td>Power cntrl</td>
<td>18 (60)</td>
<td>28</td>
</tr>
<tr>
<td>AOCS mode mng</td>
<td>15 (51)</td>
<td>16</td>
</tr>
<tr>
<td>MMM</td>
<td>25 (86)</td>
<td>35</td>
</tr>
<tr>
<td>OBT</td>
<td>12 (42)</td>
<td>35</td>
</tr>
<tr>
<td>MMFU mng</td>
<td>8</td>
<td>10</td>
</tr>
</tbody>
</table>

Explain stimuli, predicate and responses, revising model.
### Discrepancies found applying SBS

<table>
<thead>
<tr>
<th>Problem</th>
<th>Major</th>
<th>Minor</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inconsistency</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Completeness</td>
<td>1</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>Testability</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Understandability</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>Sum</strong></td>
<td><strong>2</strong></td>
<td><strong>6</strong></td>
<td><strong>14</strong></td>
</tr>
</tbody>
</table>
Conclusions

+ Method will discover issues related to completeness and consistency of requirements
+ Testability is addressed: identify stimuli and means to generate it
+ Traceability forward and backward is supported

- Data management requirement are difficult to address
- Implementation details are not addressed (black box)

- In the current version time dependent functions are not modeled
Thanks for your attention!

For more information, please contact:
Maria Hernek, **European Space Agency**
Maria.hernek@esa.int