Model-based testing of NASA systems

Dharma Ganesan,
Mikael Lindvall,
Charles Song,
Christoph Schulze
Problems in NASA projects

- Test cases are often developed manually
- Some test execution is automated (e.g., JUnit)
- Test cases miss valid "corner" cases
- Difficult to summarize what was tested

Approach: **Test Automation and Model-based Test Generation and Execution**

- Supported by NASA’s SARP program
Motivation

• Software bugs can lead to deaths, injuries, or financial loss
• Software testing consumes 50% - 75% of the development effort
• Many NASA projects could benefit from test automation
• Demonstrated several times that regular testing is not enough (defects remain undetected) – and that MBT can detect several of these defects.
Currently Targeted Projects

• **GMSEC** – Reusable framework for ground systems
  – Modeled the Core API and Applications
  – Generated executable test cases from the model
  – Confirmed defects/violations reported and fixed
  – Test cases delivered to the team

• **Core Flight Software** – Reusable framework for flight systems
  – Modeled the OS abstraction layer (OSAL)
  – Generated executable test cases from the model
  – Confirmed defects/violations reported and fixed
Currently Targeted Projects

• **Space Network – White Sands**
  – Developed an initial framework for GUI testing
  – Demonstrated the benefits of the framework
  – More work is in progress
Abstract Test Cases

Models

Traversal models

Instantiator

Concrete Test Cases

Mapping

<table>
<thead>
<tr>
<th>State/Transition label</th>
<th>API Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>click Clear</td>
<td>myWin.clear(&quot;Clear&quot;);</td>
</tr>
<tr>
<td>click World</td>
<td>myWin.clear(&quot;World&quot;);</td>
</tr>
<tr>
<td>Hello=false, World=true</td>
<td>Assert(myWin.hasText(&quot;World&quot;));</td>
</tr>
</tbody>
</table>
Advanced MBT

• Explicitly modeling the state space leads to scalability problem
  – Difficult to model all states manually
  – Difficult to slice the model for different scenarios

• We use Spec Explorer for sophisticated MBT
  – Models are C# programs (model programs)
  – State machines are generated from model programs
  – Better scenario control
Advanced MBT …

• Explicit state space modeling is easier to use for small models but is less powerful

• Advanced MBT is very powerful, requires real programming skills
Current Results

• An end-to-end approach for test automation
• Approach found specification and runtime errors
  – Teams fixed those errors!
• Approach works well on different levels:
  – API (Module interface) level testing
  – GUI testing
• Easy to infuse - e.g. GMSEC interns picked up immediately, developed models, found defects.
Sample discovered defects on GMSEC

- Sometimes results in **extra message**:
  - sub(x), pub(x), getNextMsg(), getNextMsg()
- Sometimes results in **missing message**:
  - sub(x), pub(x), unsub(x), getNextMsg()
- Sometimes results in **failure**:
  - connect(), disconnect(), connect()
Sample defects using MBT on OSAL

Issues found when running model based tests on the Posix implementation of OSAL:

- **File-descriptors issue after removing the file-system:**
  - After somewhat long tests we would run out of file-descriptors
  - This would even happen with a newly created file-system
  - Cause: OSAL does not remove file-descriptors when the file-system is removed
  - Effect: inability to create and open files.

- **Wrong error codes returned and unimplemented features:**

<table>
<thead>
<tr>
<th>Test scenario</th>
<th>Error message</th>
<th>Expected</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>checkFileSystemNullName()</td>
<td>Expected 'invalid pointer' error</td>
<td>OS_FD_ERR_INVALID_POINTER(-2)</td>
<td>OS_FS_UNIMPLEMENTED(-5)</td>
</tr>
<tr>
<td>checkFileSystemOsCallFails()</td>
<td>Expected 'filesystem' error</td>
<td>OS_FS_ERROR(-1)</td>
<td>OS_FS_UNIMPLEMENTED(-5)</td>
</tr>
<tr>
<td>checkFileSystemValid()</td>
<td>Filesystem Not Checked</td>
<td>OS_FS_SUCCESS(0)</td>
<td>OS_FS_UNIMPLEMENTED(-5)</td>
</tr>
<tr>
<td>copyFileLongSourceFilename()</td>
<td>Filesystem error code expected</td>
<td>OS_FS_ERROR(-1)</td>
<td>OS_FS_ERR_NAME_TOO_LONG(-4)</td>
</tr>
<tr>
<td>copyFileNonExistingSourceFile()</td>
<td>Filesystem error code expected</td>
<td>OS_FS_ERROR(-1)</td>
<td>OS_FS_SUCCESS(0)</td>
</tr>
<tr>
<td>readDirectoryValid()</td>
<td>Expected a valid pointer</td>
<td>OS_FS_ERROR(-1)</td>
<td>OS_FS_ERR_NAME_TOO_LONG(-4)</td>
</tr>
<tr>
<td>renameFileLongSourceFilename()</td>
<td>Filesystem error code expected</td>
<td>OS_FS_ERROR(-1)</td>
<td>OS_FS_ERR_NAME_TOO_LONG(-4)</td>
</tr>
</tbody>
</table>
MBT – some limitations

• Modeling requires specification of SUT
  – start with available spec and find spec. issues

• Developers are typically not used to modeling and abstraction

• Difficult to document individual test cases
  – Note: Models summarize all test cases
  – Some customers require document of each test case
"The GMSEC API provides an abstraction for message oriented middleware and support for multiple programming languages.

Fraunhofer has developed a sophisticated, programming language independent, model of GMSEC API behavior. Tests generated from that model have high-lighted cases where the behavior was not adequately specified, or varied between languages or middleware.

The value of the model was recently demonstrated after the addition of a new C# binding of the GMSEC API. Fraunhofer generated a large suite of test cases for the new language in one day. The remarkable turn-around was possible because only the mapping from the language independent test elements to the C# language was needed."

– Developer, NASA GMSEC Team
Summary and Next Steps

• We’re building a practical approach that
  – Helps in test automation for NASA projects
  – Has been demonstrated to be
    • effective and efficient,
    • “easy” to infuse
    • applicable to many different types of systems
  – Contact Dharma (next slide) if you are interested in more info about MBT
Contact

• Dharma Ganesan
  – (dganesan@fc-md.umd.edu)

• Mikael Lindvall
  – mlindvall@fc-md.umd.edu