Using Automated Testing Tools to Facilitate Affordable Designs

CAPT Paul Van Benthem, Dr. Valdis Berzins, Christopher Johnson, Brian Womble
DoD Testing

• The testing challenge
• Cheating to find critical faults
• Software and hardware is never finished
• Automated testing can improve affordability
• Testing is a design requirement
• Experiences with automated testing
• Conclusions
The Test Space is Infinite

- There are hundreds of systems on warships
- All systems must interoperate seamlessly
- The environment is harsh and boundless
- Human factors (faults) must be prevented
- Participants rely on simultaneous correct execution
- Lives are at stake
- Testing is sparse relative to the entire space
Critical Faults

• There will be faults left behind
• Some faults are more severe than others
  • Risk
  • Difficulty of detection
  • Probability of occurrence
  • Severity
Critical Faults

• Finding critical faults may require cheating
  • Statistically invisible = impossible to detect by black box testing
  • Clear box testing can do better
    • Use constraint solvers to synthesize test inputs for majority of cases
Systems Are Never Finished

• There are always faults to be fixed
• There are always upgrades desired
• Added features create more demand signal
Automation Can Improve Testing

• Faster development time
• Stable and consistent quality systems
• Lower costs
• Allow fast regression testing
• Changes in approach are required
Hardware Testing

• Easier than software testing
  • Uniform state representation
  • Known expected outputs
  • Effective error models
Software Testing

- More complex failure patterns
- Complete test sets not algorithmically computable in the general case
# Testability levels

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>inadequate</td>
</tr>
<tr>
<td></td>
<td>Does not meet requirements for any of the higher levels</td>
</tr>
<tr>
<td>1</td>
<td>syntactic</td>
</tr>
<tr>
<td></td>
<td>All services and data elements provided by each procurable component have published interfaces/data models that provide names and type signatures.</td>
</tr>
<tr>
<td>2</td>
<td>semantic</td>
</tr>
<tr>
<td></td>
<td>Published interfaces include precise definitions of the meaning of the services/data, including units, connection to real world objects, and requirements on outputs and final states resulting from all services</td>
</tr>
<tr>
<td>3</td>
<td>robust</td>
</tr>
<tr>
<td></td>
<td>Published interfaces include all assumptions and restrictions on inputs and states, triggering conditions for all exceptions, and expected results after exceptions</td>
</tr>
<tr>
<td>4</td>
<td>observable</td>
</tr>
<tr>
<td></td>
<td>All system attributes relevant to checking the requirements are observable either via the published operational interfaces or published augmented testing interfaces</td>
</tr>
<tr>
<td>5</td>
<td>measurable</td>
</tr>
<tr>
<td></td>
<td>All properties needed to check the requirements have clearly defined measurement and evaluation procedures</td>
</tr>
<tr>
<td>6</td>
<td>decidable</td>
</tr>
<tr>
<td></td>
<td>Pass/fail decisions for all test cases can be made entirely by automated procedures, without need for subjective human judgment</td>
</tr>
<tr>
<td>7</td>
<td>unbounded</td>
</tr>
<tr>
<td></td>
<td>Any number of random test inputs can be automatically generated and corresponding test results can be automatically checked for all services</td>
</tr>
</tbody>
</table>
### Agile vs. Waterfall Automated Testing

<table>
<thead>
<tr>
<th>Agile</th>
<th>Waterfall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allows for fluid requirements shifts and changes</td>
<td>Does not accommodate changes in requirements easily</td>
</tr>
<tr>
<td>Typically requires smaller teams of dedicated developers focused on smaller applications</td>
<td>Typically is based on larger teams working more at the system level</td>
</tr>
<tr>
<td>Requires that work be time boxed into Sprints with a working product demonstrable at the end of each time box</td>
<td>Does not have to provide any working components until delivery at the end of the development cycle</td>
</tr>
</tbody>
</table>
Testing is a Design Requirement

• Programs Approach Testing differently
• Common Instrumentation of SW could allow formalization of Automated Testing
• Using similar Technical Reference Frameworks allows common tools to be utilized
• DoN is considering sponsoring standards for testing
Experience with Automated Testing

• Rapid Integration and Test Environment (RITE)
  • SPAWAR initiative
  • Fundamental change to DoD integration activities
  • Graduated set of tests

• Focused testing accomplished in three phases is a fundamental aspect

• Continuous integration process
RITE Continuous Integration Process
Conclusions

- Automated testing has an important role in achieving affordability
- Depends on valid and sufficiently defined requirements
- Complementary quality assurance processes are needed