Establishing a Framework and Testbed for Evaluating and Infusing Software Assurance Tools

NASA OSMA Software Assurance Symposium
August, 2012

PI: Allen Nikora: Jet Propulsion Laboratory, California Institute of Technology
Co-Is: Prof. Dan Port: University of Hawaii
       Joel Wilf: Jet Propulsion Laboratory, California Institute of Technology
Intern: Denise Shigeta: Jet Propulsion Laboratory, California Institute of Technology

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration. The work was sponsored by the NASA Office of Safety and Mission Assurance under the Software Assurance Research Program. This activity is managed locally at JPL through the Assurance and Technology Program Office.
Topics

- Problem
- Approach
- Status and Results
- Future Work
- Community Involvement
Problem

• There are a large number of tools both research and commercial that may be of useful for software assurance
  – Investigation at JPL and NASA SAWG revealed most assurance is preformed “manually” and is perceived to be inefficient and ineffective for some tasks
  – Investigation at JPL revealed that there are impediments to tool use

• There is significant research interest in assurance tool development and evaluation research
  – Many SARP projects are tool related

• There is a gap between research in tools and their use on projects
  – Both commercial (COTS) and research-developed tools (“ROTS”)
Example: OSATE (AADL IDE)

- **AADL = Architectural Analysis and Design Language**
  - Text and graphical based
  - Models run-time systems
  - Similar to UML/SysML
- **OSATE can perform analyses on AADL model using properties specified**
  - Latency
  - Schedulability
  - Processor capability
- **OSATE is open source**
Example: JIRA

- Used by SQEs to track SQA issues and tasks for projects
  - Types of issues
  - Status of issues
  - Priority level of issues

- Features:
  - Produce charts and graphs of progress or work still left to do
  - Create filters
  - Create your own dashboard

- Also used by software developers to track tasks
I. Survey available tools
II. Develop tool evaluation criteria
III. Evaluate tools under controlled conditions
IV. Develop a specification for the functionality, behavior, and structure of the tool evaluation framework
V. Evaluate a subset of the tools examined in Stage III on real development efforts
VI. Provide tool evaluations and framework to the assurance community…

Ultimate goal: encourage infusion of valuable tools
I. Survey available tools

- List of candidate tools DB
- Research on tool notes and summaries
- Mapping of tools to assurance activity areas
- Degree of coverage of areas by candidate tools (strengths and gaps)
SA Tool Database

- Information on tools include:
  - Vendor, version, description, cost, license type, platform, dependencies, etc.

- Will be used as the source of information for the online resource

<table>
<thead>
<tr>
<th></th>
<th>A tool_name</th>
<th>B tool_type</th>
<th>C vendor_name</th>
<th>D tool_version</th>
<th>E tool_version_date</th>
<th>F tool_description</th>
<th>G tool_cost</th>
<th>H tool_license</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Code Collaborator</td>
<td>Collaborative Development Support</td>
<td>Smart Bear Software</td>
<td>6.5</td>
<td>10/25/2011</td>
<td>Code Collaborator is a collaborative tool to support code review. Among other features, it highlights code for review, keeps review metrics, and integrates with bug tracking systems.</td>
<td>moderate</td>
<td>commercial</td>
</tr>
<tr>
<td>3</td>
<td>COCOMO</td>
<td>Cost and Schedule Creation/Analysis</td>
<td>USC Center for Systems and Software Engineering</td>
<td>ll.2000.0</td>
<td>2000</td>
<td>CO Constructive Cost Model II (COCOMO™ II) is a model that allows one to estimate the cost, effort, and schedule when planning a new software development activity. COCOMO™ II can be used for the following major decision situations: - Making investment or other financial decisions involving a software development effort - Setting project budgets and schedules as a basis for planning and control - Deciding on or negotiating tradeoffs among software cost, schedule, functionality, performance or quality factors</td>
<td>free</td>
<td>academic</td>
</tr>
<tr>
<td>4</td>
<td>COCOTS</td>
<td>Cost and Schedule Creation/Analysis</td>
<td>USC Center for Systems and Software Engineering</td>
<td>NA</td>
<td>2002</td>
<td>The CO Constructive COTS (COCOTS) model is intended to capture the true cost of integrating COTS software components into a larger system. This includes traditional costs associated with new software development such as the cost of requirements definition, design, code, test, and software maintenance, as well as the cost of licensing and redistribution rights, royalties, effort needed to understand the COTS software, pre-integration assessment and evaluation, post-integration certification of compliance with</td>
<td>free</td>
<td>academic</td>
</tr>
<tr>
<td></td>
<td>COSYSMO</td>
<td>Cost and Schedule Creation/Analysis</td>
<td>Lean Advancement Initiative Center for Technology</td>
<td>2.0</td>
<td>2010</td>
<td>The COSYSMO (Constructive Systems Engineering Cost Model) model is used to estimate the Systems Engineering effort for</td>
<td>free</td>
<td>academic</td>
</tr>
</tbody>
</table>
# Tool Application Survey

## Life-Cycle View

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Collaborative Development Support</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>Code Collaborator</td>
<td>F</td>
<td>D</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>B</td>
<td>D</td>
<td>B</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>Cost and Schedule Creation/Analysis</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>COCOMO</td>
<td>A</td>
<td>D</td>
<td>B</td>
<td>D</td>
<td>B</td>
<td>D</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>D</td>
<td>F</td>
</tr>
<tr>
<td>COCOTS</td>
<td>A</td>
<td>D</td>
<td>F</td>
<td>A</td>
<td>B</td>
<td>D</td>
<td>B</td>
<td>C</td>
<td>C</td>
<td>D</td>
<td>B</td>
</tr>
<tr>
<td>COSYSMO</td>
<td>A</td>
<td>D</td>
<td>F</td>
<td>C</td>
<td>B</td>
<td>D</td>
<td>B</td>
<td>C</td>
<td>C</td>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>SCAT</td>
<td>A</td>
<td>D</td>
<td>F</td>
<td>F</td>
<td>B</td>
<td>D</td>
<td>B</td>
<td>C</td>
<td>C</td>
<td>D</td>
<td>F</td>
</tr>
<tr>
<td>Formal Specification/Analytical Verification</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>Alloy</td>
<td>D</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>D</td>
<td>C</td>
<td>C</td>
<td>F</td>
<td>B</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>Java Pathfinder (ARC)</td>
<td></td>
<td>D</td>
<td>F</td>
<td>B</td>
<td>A</td>
<td>D</td>
<td>C</td>
<td>C</td>
<td>F</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>PVS</td>
<td>D</td>
<td>B</td>
<td>A</td>
<td>F</td>
<td>B</td>
<td>D</td>
<td>D</td>
<td>C</td>
<td>F</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>SAL</td>
<td>D</td>
<td>B</td>
<td>A</td>
<td>F</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>C</td>
<td>F</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>SCR</td>
<td>D</td>
<td>B</td>
<td>A</td>
<td>F</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>C</td>
<td>F</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>SPIN</td>
<td>D</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>C</td>
<td>F</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>
## Example Survey Results

- Evaluations from 9 assurance practitioners – Commonly used tools
- Tools below had similar evaluations
  - Support = looked for all A’s and B’s
  - No support = looked for all D’s and F’s

<table>
<thead>
<tr>
<th>Tool</th>
<th>Assurance areas supported</th>
<th>Assurance areas not supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bugzilla</td>
<td>Code, test, dependability assurance</td>
<td>Cost, planning, architecture, performance, resource assurance</td>
</tr>
<tr>
<td>Coverity</td>
<td>Code, test, product, safety, risk assurance</td>
<td>Planning, requirements, architecture, process, cost, schedule, assurance management</td>
</tr>
<tr>
<td>DOORS</td>
<td>Requirements, test, delivery, product, risk assurance</td>
<td>Code, cost, schedule assurance</td>
</tr>
<tr>
<td>JIRA</td>
<td>Code, test, delivery, operations &amp; maintenance, process, product, risk, contractor assurance, assurance management</td>
<td>Performance, cost assurance</td>
</tr>
<tr>
<td>JPLs PRS</td>
<td>Code, test, delivery, dependability assurance</td>
<td>Cost, architecture assurance</td>
</tr>
<tr>
<td>SLIC</td>
<td>Product, cost, schedule assurance</td>
<td>Planning, requirements, architecture, test, safety, security, performance, dependability, resource assurance</td>
</tr>
</tbody>
</table>

Note: Evaluations were given by assurance practitioners on common tools used in the field.
I. Survey available tools

II. Develop tool evaluation criteria
   – Candidate evaluation criteria
   – Mappings of tools to decisions and evidence supplied

What tools might be useful for assurance?

How do these tools support assurance?

What is the value of using particular tools? (testbed)

How can we share and evolve information about value of tools?
Preliminary Assessment Criteria

• **Applicability**
  – From tool survey data

• **Effectiveness**
  – Scalability ratio = \{\text{max amount handleable with tool} / \text{max amount handleable manually}\}
  – Assurance productivity efficiency = \{\text{average amount assured per function point with tool} / \text{average amount assured per function point manual}\}
  – Accuracy ratio = \{\text{average number errors with tool} / \text{average number error manual}\}
  – Average accuracy = \{\text{average errors with tool}\}
  – Accuracy variance = \{\text{variance of errors with tool}\}
  – Coverage fraction \([0-1]\) = \{\text{amount tool covers} / \text{total amount}\}

• **Tool Availability**

• **Usability**
  – As per Seffah et al consolidated usability model [Software Qual J (2006) 14: 159–178]

• **Relationship To other Tools**
I. Survey available tools
II. Develop tool evaluation criteria
III. Evaluate tools under controlled conditions
   – Example: ODASA Static Code Analyzer (in progress)
IV. Develop a specification for the functionality, behavior, and structure of the tool evaluation framework
   – Top-down framework found to be too constricting
   – Use common criteria + user criteria + user experience
V. Evaluate a subset of the tools examined in Stage III on real development efforts (III and V done in concert)
   – Example: ODASA Static Code Analyzer (in progress)
   – Evaluating on SMAP, MGSS IOS, ICX (DoD project) and others (in progress)
I. Survey available tools
II. Develop tool evaluation criteria
III. Evaluate tools under controlled conditions
IV. Develop a specification for the functionality, behavior, and structure of the tool evaluation framework
V. Evaluate a subset of the tools examined in Stage III on real development efforts (III and V done in concert)

VI. Provide tool evaluations and framework to the assurance community – In Progress
   – Setting up JPL externally-facing site by end of FY
   – Will include Assurance Tool Survey, Database, and WIKI

What tools might be useful for assurance? | How do these tools support assurance? | What is the value of using particular tools? (testbed) | How can we share and evolve information about value of tools?
Future Work

• Collect more data
  – SARP TIM next week – extract knowledge from NASA assurance researchers while they’re here!

• Analyze/interpret the data
  – Statistical hypothesis testing
  – Distribution of “grades”
  – Determine which tools provide strong support
  – Are there any areas lacking tool support where we should develop new tools?
  – What opportunities for tool use are there?

• Set up online resource (before end of FY12)
  – Survey: to collect more data and evolve tool/applicability matrix
  – Database: to communicate basic tool information
  – WIKI: to comment on and learn about tools experience
Community Involvement

• Get interviewed!
  – Contribute your assessment on tools applied to assurance
• Use, evaluate, and infuse tools
• Contribute to the tools WIKI when it goes online