Project Title: System Health Assessment
Lead University: University of Denver
Principal Investigator: Anneliese Andrews
Budget: $17,500
Schedule:
Start date: Jun 1, 2016
End date: Dec 31, 2016

Objective(s):
Apply assessment techniques and prediction methods to a very large system to show applicability to a large legacy system.

Problem Statement:
- Assessing the health of aging systems and identifying architectural problems, predict future defects and the maintenance effort to keep these assets performing satisfactorily.
- Identifying whether components reused for new systems fit well or not.

This project aims to apply assessment techniques and prediction methods to a very large system to show applicability to a large legacy system.

Deliverables:
- Techniques
- Tools
- Documentation
- Analysis report
Methodology of The Integrated Method:

1. **Fault-prone Component Analysis**
   - Fault Architecture
   - Release Quality Assessment

2. **Focus Testing**
   - Prioritizing Testing Activities
   - Testing Guidelines

3. **Static & Dynamic Defect Estimation Methods**
   - Release Decisions

4. **Fault-prone Component Analysis**
   - Fault Architecture
   - Multi-Release Quality Assessment

This flowchart illustrates the steps involved in the Methodology of The Integrated Method, starting with fault-prone component analysis and progressing through focus testing, static and dynamic defect estimation methods, and finally, multi-release quality assessment.
TECHNICAL SUMMARY

Integrated Method & Maintenance:

Replicate Case Study to:
- Enhance SW Maintenance effort
- To assess product quality
- Reduce maintenance cost of
  - Corrective
  - Adaptive
  - Perfective
  - Preventive
- (Contribute to the Empirical SW Engineering Knowledge and towards the maturity of external validity)
Fault-prone component analysis:

<table>
<thead>
<tr>
<th>Ranked Components</th>
<th>Development</th>
<th>System Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>component</td>
<td># defects</td>
<td>component</td>
</tr>
<tr>
<td>A</td>
<td>131</td>
<td>B (new)</td>
</tr>
<tr>
<td>B (new)</td>
<td>50</td>
<td>F (new)</td>
</tr>
<tr>
<td>C</td>
<td>40</td>
<td>Q (new)</td>
</tr>
<tr>
<td>D (new)</td>
<td>35</td>
<td>I (new)</td>
</tr>
<tr>
<td>E</td>
<td>28</td>
<td>J</td>
</tr>
<tr>
<td>F (new)</td>
<td>18</td>
<td>D (new)</td>
</tr>
<tr>
<td>G (new)</td>
<td>14</td>
<td>E</td>
</tr>
<tr>
<td>H</td>
<td>13</td>
<td>C</td>
</tr>
<tr>
<td>I (new)</td>
<td>12</td>
<td>Q</td>
</tr>
<tr>
<td>J</td>
<td>10</td>
<td>P</td>
</tr>
<tr>
<td>K</td>
<td>6</td>
<td>R</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>
Comparison in post-release:

- Is system testing finding all problems?
  - Analysis on system test versus post-release

<table>
<thead>
<tr>
<th></th>
<th>Release 1</th>
<th>Post-release Fault-prone</th>
<th>Post-release Normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Test</td>
<td>1</td>
<td>5(5 new)</td>
<td></td>
</tr>
<tr>
<td>Fault-prone</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Test</td>
<td>6</td>
<td>168(2 new)</td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Fault architecture:

• Identify fault-prone components using defect cohesion measures.
• Create a fault component directory structure.
• Identify fault-prone relationships using defect coupling measures.
• Develop a fault architecture.
• Create fault architecture diagrams.
• “Lift” to subsystem level.
Fault Architecture (Release 1: Post-release)
Multi-Release Analysis: Subsystem Level Cumulative Release Diagram:

What kinds of problems occur release after release?

Results:
• More problems internal to components rather than between components.
• Small number of specific components repeatedly have problems.
TECHNICAL SUMMARY

Cumulative Defect Curves (Release 1):

- development
- system test
- system test reordered

Week
Cumulative # defects
0 50 100 150 200 250 300 350 400 450 500
44 47 50 53 56 59 62 65 68 71 74 77 80 83 86 89 92 95 98 101 104

research • deploy • train • evaluate
Why focus on maintenance?

According to Capers Jones (CrossTalk December 2007)

- Post Y2K and Euro mass software modification the 1st decade of 21st century shows more than 50% of software population is engaged in modifying existing software

<table>
<thead>
<tr>
<th>Year</th>
<th>Development Personnel</th>
<th>Maintenance Personnel</th>
<th>Total Personnel</th>
<th>Maintenance Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1950</td>
<td>1,000</td>
<td>100</td>
<td>1,100</td>
<td>9.09%</td>
</tr>
<tr>
<td>1955</td>
<td>2,500</td>
<td>250</td>
<td>2,750</td>
<td>9.09%</td>
</tr>
<tr>
<td>1960</td>
<td>20,000</td>
<td>2,000</td>
<td>22,000</td>
<td>9.09%</td>
</tr>
<tr>
<td>1965</td>
<td>50,000</td>
<td>10,000</td>
<td>60,000</td>
<td>16.67%</td>
</tr>
<tr>
<td>1970</td>
<td>125,000</td>
<td>25,000</td>
<td>150,000</td>
<td>16.67%</td>
</tr>
<tr>
<td>1975</td>
<td>350,000</td>
<td>75,000</td>
<td>425,000</td>
<td>17.65%</td>
</tr>
<tr>
<td>1980</td>
<td>600,000</td>
<td>300,000</td>
<td>900,000</td>
<td>33.33%</td>
</tr>
<tr>
<td>1985</td>
<td>750,000</td>
<td>500,000</td>
<td>1,250,000</td>
<td>40.00%</td>
</tr>
<tr>
<td>1990</td>
<td>900,000</td>
<td>800,000</td>
<td>1,700,000</td>
<td>47.06%</td>
</tr>
<tr>
<td>1995</td>
<td>1,000,000</td>
<td>1,100,000</td>
<td>2,100,000</td>
<td>52.38%</td>
</tr>
<tr>
<td>2000</td>
<td>750,000</td>
<td>2,000,000</td>
<td>2,750,000</td>
<td>72.73%</td>
</tr>
<tr>
<td>2005</td>
<td>775,000</td>
<td>2,500,000</td>
<td>3,275,000</td>
<td>76.34%</td>
</tr>
<tr>
<td>2010</td>
<td>800,000</td>
<td>3,000,000</td>
<td>3,800,000</td>
<td>78.95%</td>
</tr>
<tr>
<td>2015</td>
<td>1,000,000</td>
<td>3,500,000</td>
<td>4,500,000</td>
<td>77.78%</td>
</tr>
<tr>
<td>2020</td>
<td>1,100,000</td>
<td>3,750,000</td>
<td>4,850,000</td>
<td>77.32%</td>
</tr>
<tr>
<td>2025</td>
<td>1,250,000</td>
<td>4,250,000</td>
<td>5,500,000</td>
<td>77.27%</td>
</tr>
</tbody>
</table>
Source: DoD Test and Evaluation Master Plan (TEMP) Guide Book Ver 2.1- May 2013:

The majority of life cycle costs for DoD systems reside in the Operations and Sustainment (O&S) phase, where the single greatest driver of O&S costs is unreliability.

The more reliable the system, the less it costs to operate and sustain in the field. With today’s highly complex systems, a small decrease in reliability can mean additional, substantial cost, but a small investment in reliability growth can significantly decrease O&S costs.
Dan Galorath/SEER for Software estimates SW maintenance typically forms 75% of total ownership cost.
Case Study Research Questions:

- How can historical defect data be used as an actionable information to effectively manage software maintenance projects?
  - Where a balance among scope, cost, schedule, quality is maintained

- Can historical defect data be used to identify:
  - areas for software quality improvement
  - areas to reduce cost of maintenance and cost of defect repair
  - methods to predict number of undetected defects before release
  - components or subsystems as candidates for reengineering
  - areas for productivity improvement
  - methods to improve software development process
Context of a possible Case Study:
**Defect lifecycle and Testing Phases**
Hypothesis: The establishment of *practical link and feedback mechanism* between defect lifecycle and test lifecycle improves software maintenance project effectiveness.
PLAN FOR NEXT REVIEW

- Assess information in database (Jan 1- Mar 1, 2016)
- Adjust existing assessment methods to data available (Jan 1- Jun 1, 2016)
- Prepare data for analysis (May 1- Jul 1, 2016)
- Apply analysis methods, prediction methods, and interpret results (Jul 1- Oct 1, 2016)
- Suggest corrective actions (Oct 1- Dec 31, 2016)