Empirical Analysis of Safety-Critical Software Anomalies Post-Launch

Burton Sigal (JPL)
Robyn Lutz and Carmen Mikulski (JPL)

NASA Code Q Software Program Center Initiative UPN 323-08,
Supported by JPL Assurance Technology Program Office (ATPO)

NASA GRC Assurance Technology Conference
May 29, 2002
Topics

- Overview
- Results
  - Quantitative analysis
  - Evolution of requirements
  - Visualization tools
- Work-in-progress
- Contribution
Overview: Goal

To reduce the number of safety-critical software anomalies that occur during flight by providing a quantitative analysis of previous anomalies as a foundation for process improvement.
Overview: Approach

- Analyzed anomaly data using *Orthogonal Defect Classification (ODC)* method
  - Developed at IBM; widely used by industry
  - Quantitative approach
  - Used here to detect patterns in anomaly data

- Evaluated ODC using *Formalized Pilot Study*
  - R. Glass ['97] detailed rigorous process to get valid results
  - 35 steps divided into 5 phases
  - Used here to evaluate ODC for NASA use
Overview: What is ODC?

- ODC is a measurement technology for software engineering that uses defects found as a source of information to understand and improve:
  - The software product
  - The software process
- Defect is described as a required change, necessary to fix the program or product

More information can be found at http://www.research.ibm.com/softeng
Overview: ODC Defect Classification

- **Activity:** what was taking place when anomaly occurred?
  - Based on the activity performed when the defect was recorded

- **Trigger:** what was the catalyst?
  - Catalyst that causes defect to manifest itself as a failure
  - Different triggers for each activity

- **Target:** what was fixed?
  - The highest-level identity of the entity that was fixed

- **Type:** what kind of fix was done?
  - The actual correction that was made to fix the problem
Topics

- Overview
- Results
  - Quantitative analysis
  - Evolution of requirements
  - Visualization tools
- Work-in-progress
- Contribution
Results: Summary of Activities

- Started with a small set of 89 critical Incident/Surprise/Anomaly reports (ISAs)
  - Adapted ODC classification scheme for post-launch ISAs
  - Preliminary results indicated post-launch software requirements evolution
- Expanded set and analyzed 199 of high criticality ISAs
  - 7 spacecraft: Cassini, Deep Space 1, Mars Global Surveyor, Galileo, Mars Polar Lander, Mars Climate Orbiter, Stardust
- Institutional defect database → Access database of data of interest → Excel spreadsheet with ODC categories → Pivot tables with multiple views of data
- 2-D and 3-D frequency counts of Activity, Trigger, Target, Type, Trigger within Activity, Type within Target, etc.
- Found patterns → formulated hypotheses → provided recommendations
Results: Summary of Activities

- Started with a small set of 89 critical Incident/Surprise/Anomaly reports (ISAs)
  - Adapted ODC classification scheme for post-launch ISAs
  - Preliminary results indicated post-launch software requirements evolution

- Expanded set and analyzed 199 of high criticality ISAs
  - 7 spacecraft: Cassini, Deep Space 1, Mars Global Surveyor, Galileo, Mars Polar Lander, Mars Climate Orbiter, Stardust

- Institutional defect database → Access database of data of interest → Excel spreadsheet with ODC categories → Pivot tables with multiple views of data

- 2-D and 3-D frequency counts of Activity, Trigger, Target, Type, Trigger within Activity, Type within Target, etc.

- Found patterns → formulated hypotheses → provided recommendations
## Results: ODC Adaptation to JPL

- Adapted ODC classification to post-launch spacecraft Incident Surprise Anomalies (ISAs)

### Activities vs. Triggers

<table>
<thead>
<tr>
<th>Activities</th>
<th>Triggers</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Test</td>
<td>Software Configuration</td>
</tr>
<tr>
<td></td>
<td>Hardware Configuration</td>
</tr>
<tr>
<td></td>
<td>Start/Restart, Shutdown</td>
</tr>
<tr>
<td></td>
<td>Command Sequence Test</td>
</tr>
<tr>
<td></td>
<td>Inspection/Review</td>
</tr>
<tr>
<td>Flight Operations</td>
<td>Recovery</td>
</tr>
<tr>
<td></td>
<td>Normal Activity</td>
</tr>
<tr>
<td></td>
<td>Data Access/Delivery</td>
</tr>
<tr>
<td></td>
<td>Special Procedure</td>
</tr>
<tr>
<td></td>
<td>Hardware Failure</td>
</tr>
<tr>
<td>Unknown</td>
<td>Unknown</td>
</tr>
</tbody>
</table>

### Targets vs. Types

<table>
<thead>
<tr>
<th>Targets</th>
<th>Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ground Software</td>
<td>Function/Algorithm</td>
</tr>
<tr>
<td></td>
<td>Interfaces</td>
</tr>
<tr>
<td></td>
<td>Assignment/Initialization</td>
</tr>
<tr>
<td></td>
<td>Timing</td>
</tr>
<tr>
<td>Flight Software</td>
<td>Function/Algorithm</td>
</tr>
<tr>
<td></td>
<td>Interfaces</td>
</tr>
<tr>
<td></td>
<td>Assignment/Initialization</td>
</tr>
<tr>
<td></td>
<td>Timing</td>
</tr>
<tr>
<td></td>
<td>Flight Rule</td>
</tr>
<tr>
<td>Build /Package</td>
<td>Install Dependency</td>
</tr>
<tr>
<td></td>
<td>Packaging Scripts</td>
</tr>
<tr>
<td>Ground Resources</td>
<td>Resource Conflict</td>
</tr>
<tr>
<td>Info. Development</td>
<td>Documentation</td>
</tr>
<tr>
<td></td>
<td>Procedures</td>
</tr>
<tr>
<td>Hardware</td>
<td>Hardware</td>
</tr>
<tr>
<td>None/Unknown</td>
<td>Nothing Fixed</td>
</tr>
<tr>
<td></td>
<td>Unknown</td>
</tr>
</tbody>
</table>

02/28/02
Results: Profiling Analysis

Sample Question: What is the typical signature of a post-launch critical software anomaly?

Metrics:
- Activity = Flight Operations
- Trigger = Data Access/Delivery
- Target = Information Development
- Type = Procedures

Example: Star Scanner anomaly
- Activity = occurred during flight
- Trigger = star scanner telemetry froze
- Target = fix was new description of star calibration
- Type = procedure written
Results: Sample Analyses

Ground/Flight S/W vs. Type within Activity

Trigger vs. Target
Results: *Evolution of Safety-Critical Requirements Post-Launch*

- Anomalies sometimes result in changes to software requirements
- Found that requirements changes are not due to earlier requirement errors
- Instead, requirements changes are due to:
  - Need to handle rare event or scenario (software adds fault tolerance)
  - Need to compensate for hardware failure or limitations (software adds robustness)

- **Metrics:**
  - Activity = Flight Operations
  - Trigger = Hardware failure
  - Target = Flight Software
  - Type = Function Algorithm

- **Example:** Damaged Solar Array Panel cannot deploy as planned
  - Activity = occurred during flight
  - Trigger = Solar Array panel incorrect position (broken damper which had rotated into the SAM hinge-line which prevented latching)
  - Target = Changes to flight software
  - Type = Add a solar array powered hold capability in the FSW
Results: *Evolution of Safety-Critical Requirements Post-Launch*

- Confirms value of requirements completeness for fault tolerance
- Confirms value of contingency planning to speed change
- Contradicts assumption that “what breaks is what gets fixed”
- Suggests need for better requirements engineering for maintenance
Results:

Web-based Visualization Tool

- Results of Peter Neubauer (ASU), Caltech/JPL Summer Undergraduate Research Fellow, 2001
- Developed alternative visualizations of data results to support users’ analyses
- Web-based tool assists distributed users
- Sophisticated tool architecture builds on existing freeware
- Demo at QA Section Manager’s meeting (FAQ: Would this work for our project?)
- Demo to D. Potter’s JPL group developing next-generation Failure Anomaly Management System
Work-In-Progress

- Assembling process recommendations tied to specific findings and unexpected patterns
  - Ex: Create checklist of “missing” procedures that were needed during operations in previous, similar missions
  - Ex: Use early Contingency Planning to anticipate and support requirements evolution

- Incorporate standardized ODC classifications into next-generation problem-failure reporting database to support automation and visualization

- Profile by mission phase: are there more anomalies during critical mission phases?

- Pursuing funding for FY’03 extension of ODC work to pre-launch and additional applications
**Work-In-Progress**

- Collaborating with Mars Exploration Rover to experimentally extend ODC approach to pre-launch software problem reports
  - Adjusting ODC classifications to testing phase
  - Feedback from Project has been noteworthy
  - Analyzing Problem Reports per ODC classification
  - Results can support tracking trends and progress:
    - Graphical data summaries will be delivered to Project
  - Results can support better understanding of typical problem signatures:
    - Hypothesis testing results will be delivered to Project
Contribution

- User selects preferred representation (e.g., 2-D bar graph) and set of projects to view
- Data mines historical and current databases of anomaly and problem reports to feed-forward into future projects
- Uses metrics information to identify unexpected patterns and focus on problem areas
- Provides rapid quantitative foundation for process improvement
- Equips us with a methodology to continue to learn as projects and processes evolve