Orthogonal Defect Classification for Projects

Robyn Lutz and Carmen Mikulski

robyn.lutz@jpl.nasa.gov carmen.mikulski@jpl.nasa.gov

QMSW contact: Burt Sigal

Jet Propulsion Laboratory

Quality Mission Software Workshop

May 13-15, 2003

The research described in this presentation was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. It was funded by NASA's Office of Safety and Mission Assurance, Center Initiative UPN 323-08. The first author's research is supported in part by National Science Foundation Grants CCR-0204139 and CCR-0205588.

Orthogonal Defect Classification (ODC) for Projects

- **What are the objectives of defect measurement?**
  - Reduce post-launch critical software anomalies
  - Provide quantitative foundation for process improvement
- **Results on projects**
  - Provide defect patterns that can be used as leading indicators of software problems
  - Provide process improvement recommendations based on typical patterns that should be avoided
- **What do we do in the short run?**
  - ODC has been piloted on 9 spacecraft projects, can be expanded to include more projects
  - The existing institutional defect database is used
- **What should we do in the long run?**
  - Partial automation and tool development

Current Status

- **ODC is an Industry standard; mature technique; developed at IBM**
  - ODC gives signature of defects
  - ODC gives high-level patterns of defects
  - Adapted ODC to spacecraft domain
- **Applications at JPL:**
  - 7 launched spacecraft: (critical post-launch incident/surprise/anomaly reports
  - MER: testing problem/failure reports
  - Deep Impact: software change request reports
- **Attributes characterize each defect:**
  - Activity: when defect surfaced, e.g., integration test
  - Trigger: situation that allowed defect to appear; e.g., testing a single command
  - Target: what got fixed; e.g., flight software
  - Type: nature of the fix, e.g., assignment/initialization
- **Analysis of patterns**
  - Are results confirmatory or unexpected? OK or not?
  - Defect models are typical patterns – so far, ODC has patterns from 9 spacecraft
  - For unexpected patterns causal analysis is done on that specific subset
  - Resulting recommendations extracted by analyst from ODC results, iterated with project

Testing Problem Reports

Problem Report file for MER

[redacted]

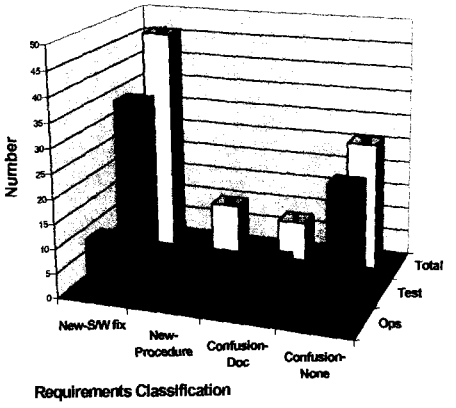
➢Where are the spikes?
➢Phase-by-phase deltas?
➢Activity/Trigger/Target/
➢Type look nominal?

MER use: Browse pivot chart (Excel) for overview/closer look at testing

[redacted]

➢Focus on problem reports that involve requirements
➢Improvement release-by-release uneven: why?
➢Many closed with no fix: why?

MER use: Identify patterns of concern for more investigation

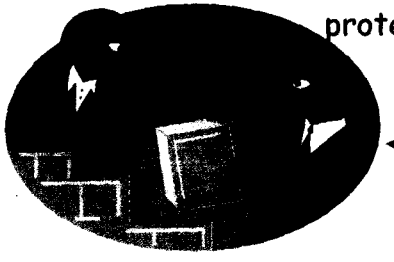


[redacted]

MER use: Improved understanding of data, underlying causes, defect mechanisms

[redacted]

Recommendations for MER and future projects:
➢Earlier assignment of criticality ratings
➢If software's behavior confused testers, enhance documentation
➢Earlier testing of fault-protection



[redacted]

MER use: Implement/defer recommendations

*ODC = Orthogonal Defect Classification technique [IBM]

What can be done in the short run?

- **ODC has been piloted on 9 projects and can be expanded to include more projects**
- **The existing defect database can be used**
- **Current estimated effort**
  - ODC ~ 4 minutes/defect vs. Root cause ~ 19 (Leszak & Perry 2003)
  - ODC requires little/no additional project time (uses existing fields)
  - Data manipulation and analysis ~ 4 days/month
  - Reduces effort on causal analysis to just unexpected patterns of interest
- **Benefits**
  - Visualization & browsing options (Excel pivot tables and charts)
  - Gives immediate results to projects
  - Provides guidance to future projects
- **Analysis of patterns**
  - Incorporates project results into multi-project baseline patterns
  - Can answer project's questions regarding defects
  - Feeds forward into process recommendations

What should we do in long run?

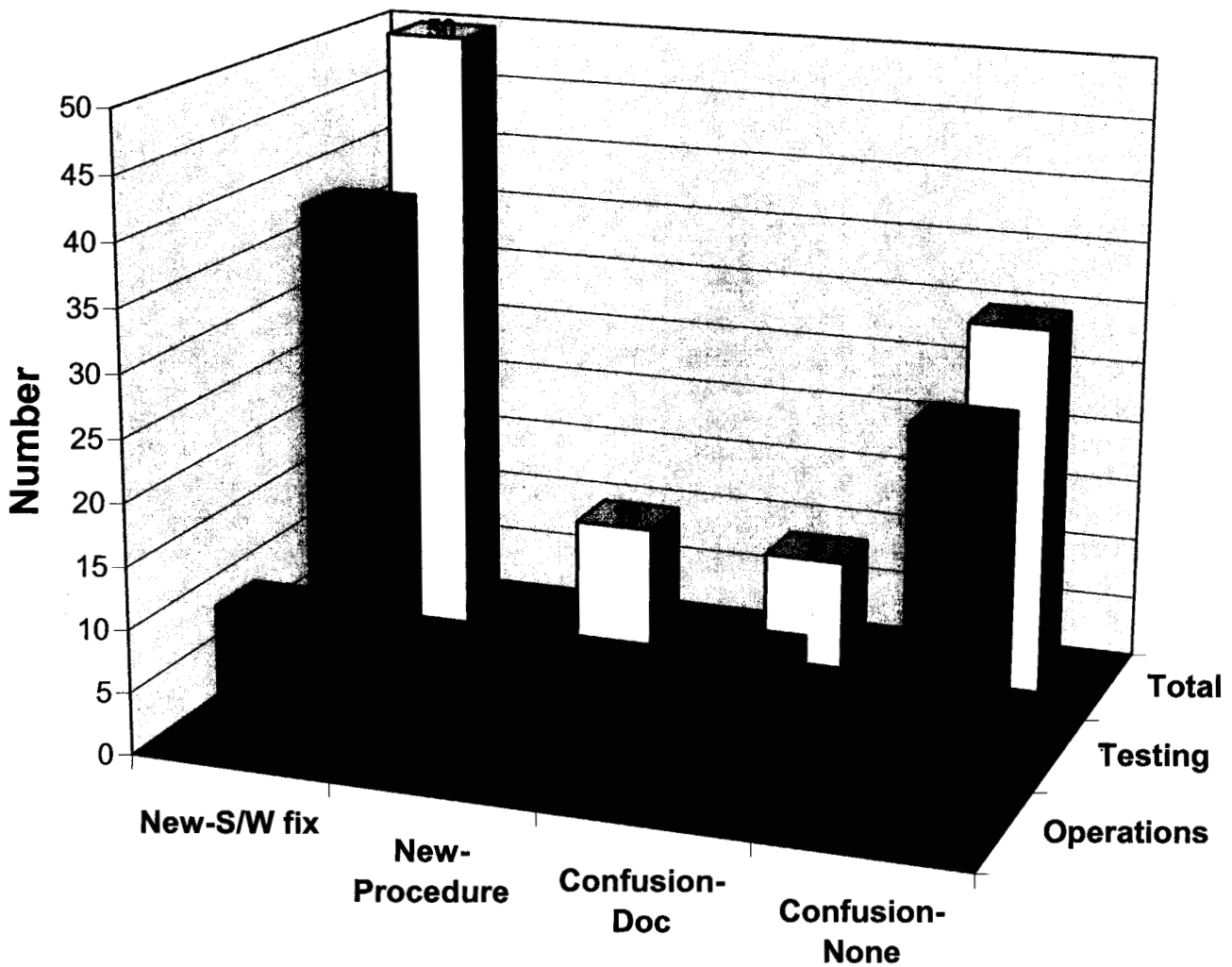
- **Partial automation of classification is possible:**
  - Customize pull-down menus (Pick-Lists) of the problem reporting system
  - Train users on ODC:
    - When the problem is found (originators)
    - When the problem is analyzed and fixed (S/W engineers)
  - Improve fidelity of raw data
  - Automation supports timely feedback to projects
- **Procedures/Tools to generate patterns from PFR database**

ODC Classification Sample

Activities	Triggers
System Test	Software Configuration Hardware Configuration Start/Restart, Shutdown Command Sequence Test Inspection/Review
Flight Operations	Recovery Normal Activity Data Access/Delivery Special Procedure Hardware Failure
Unknown	Unknown

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

Sample Pattern from ODC



Process Recommendations Sample

Examples of Unexpected ISA patterns:	*Process Recommendation:*	*Example (from spacecraft):*
22% of critical ISAs had ground software as Target (fix)	Software QA for ground software	Unable to process multiple submissions. Fixed code.
23% of critical ISAs had procedures as Type	Assemble checklist of needed procedures for future projects	Not in inertial mode during star calibration. Additions made to checklist to prevent in future.
Of these, 41% had Data access / delivery as Trigger	Better communication of changes and updates to operations	Multiple queries for spacecraft engineering and monitor data failed. Streamlined notification to operators of problems.
34% of critical ISAs involving system test had software configuration as Trigger (cause) ; 24% had hardware configuration as Trigger	Additional end-to-end configuration testing	OPS personnel did not have a green command system for the uplink of two trajectory-correction command files. Problems resulted from a firewall configuration change.

Sample Lessons Learned from ODC

- **Testing reports give "crystal ball" into operations**
  - False-positive testing problem reports (where software behavior is correct but unexpected) provide insights into requirements confusions on the part of users
  - If software behavior surprised testers, it may surprise operators
- **Closing problem reports with "No-Fix-Needed" decision can waste opportunity to document /train/ change procedure**
  - Avoid potentially hazardous recurrence
  - Important in long-lived systems with turnover, loss of knowledge
- **Need traceability from testing into operations**
  - Some testing PRs resolved by changes to operational procedures
  - Capture rationale for change to use in ops & maintenance

For More Information

- "Operational Anomalies as a Cause of Safety-Critical Requirements Evolution," R. Lutz and C. Mikulski, *The Journal of Systems and Software*, to appear (available on-line now at sciencedirect.com)
- "Requirements Discovery During the Testing of Safety-Critical Software," R. Lutz and C. Mikulski, *Proc. 25th International Conference on Software Engineering (ICSE'03)* , May 3-10, 2003, Portland, OR.
- "Resolving Requirements Discovery in Testing and Operations," R. Lutz and C. Mikulski, *Proc. 11th IEEE Requirements Engineering Conference*, Sept. 8-12, 2003, Seattle, WA.
- "Better Analysis of Defect Data at NASA," T. Menzies, R. Lutz, and C. Mikulski, *Proc. 15th International Conference on Software Engineering and Knowledge Engineering*, July 1-3, 2003, San Francisco, CA.