



Patterns of Software Defect Data on Spacecraft

Robyn Lutz and Carmen Mikulski

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

Jet Propulsion Laboratory

Space Mission Challenges for Information Technology

(SMC-IT) July 13-16, 2003

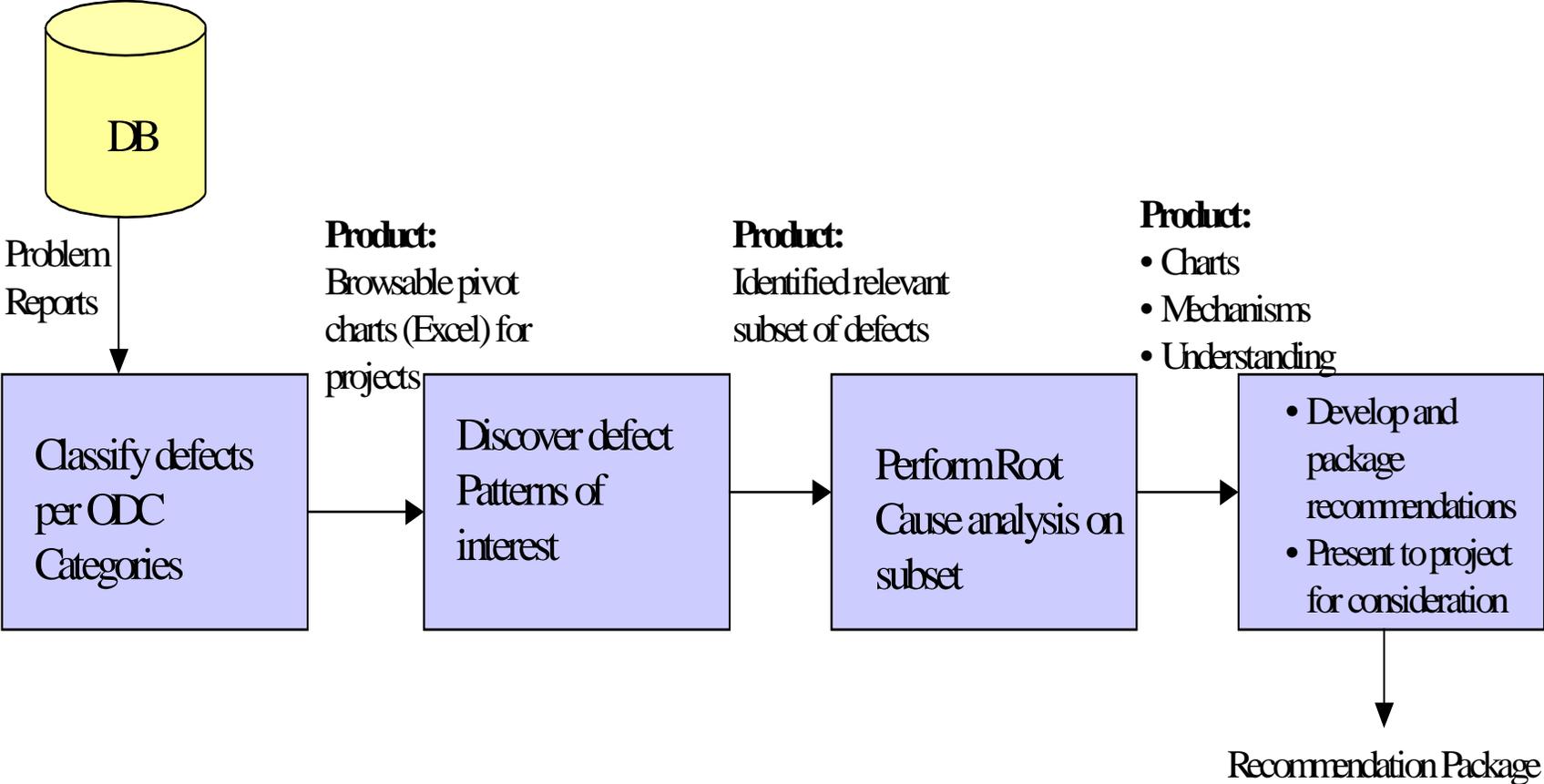
The research described in this presentation was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. It was funded by NASA's Office of Safety and Mission Assurance, Center Initiative UPN 323-08. The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

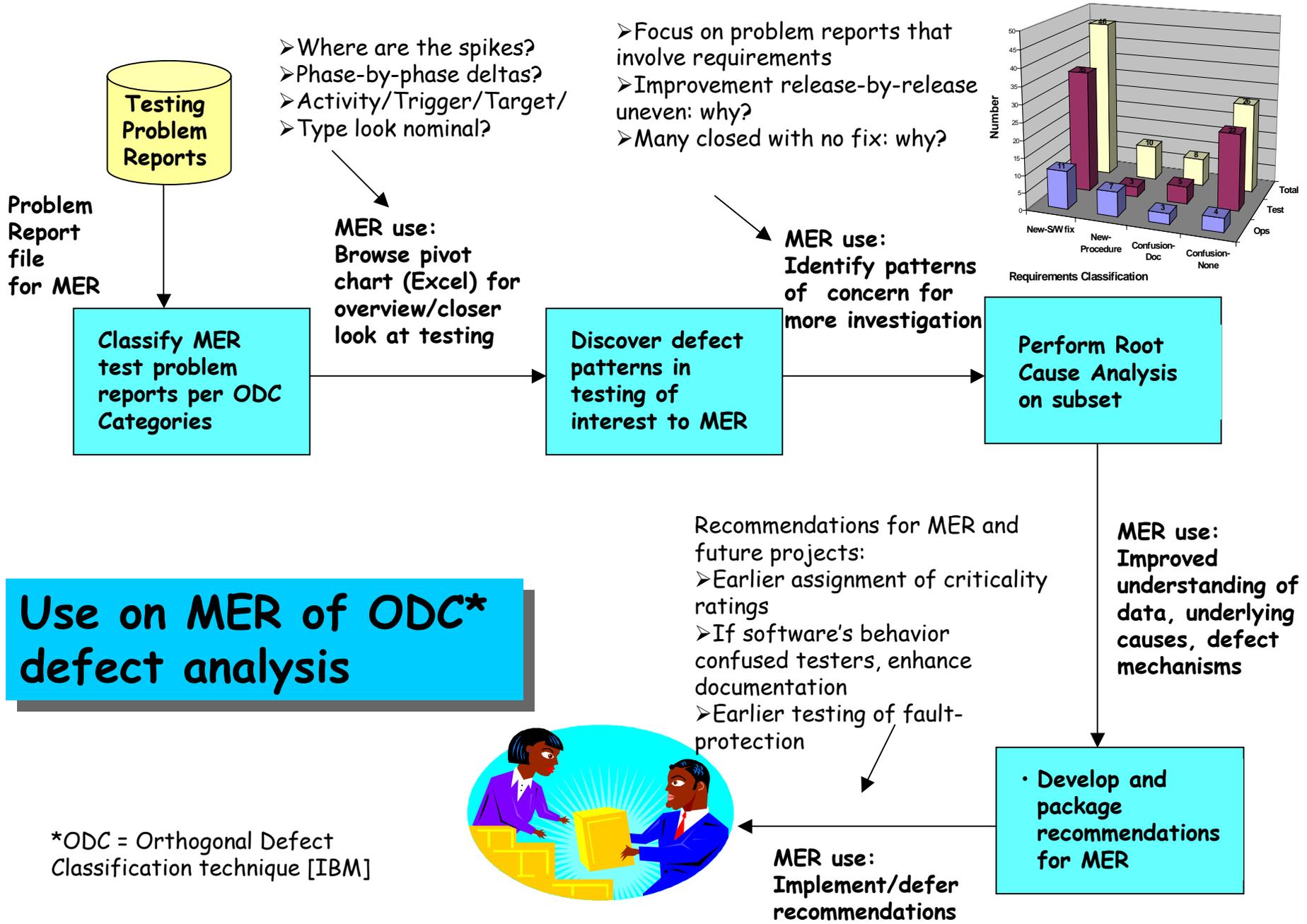
Overview of ODC



- **ODC is a software defect analysis technique**
 - 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

ODC Approach

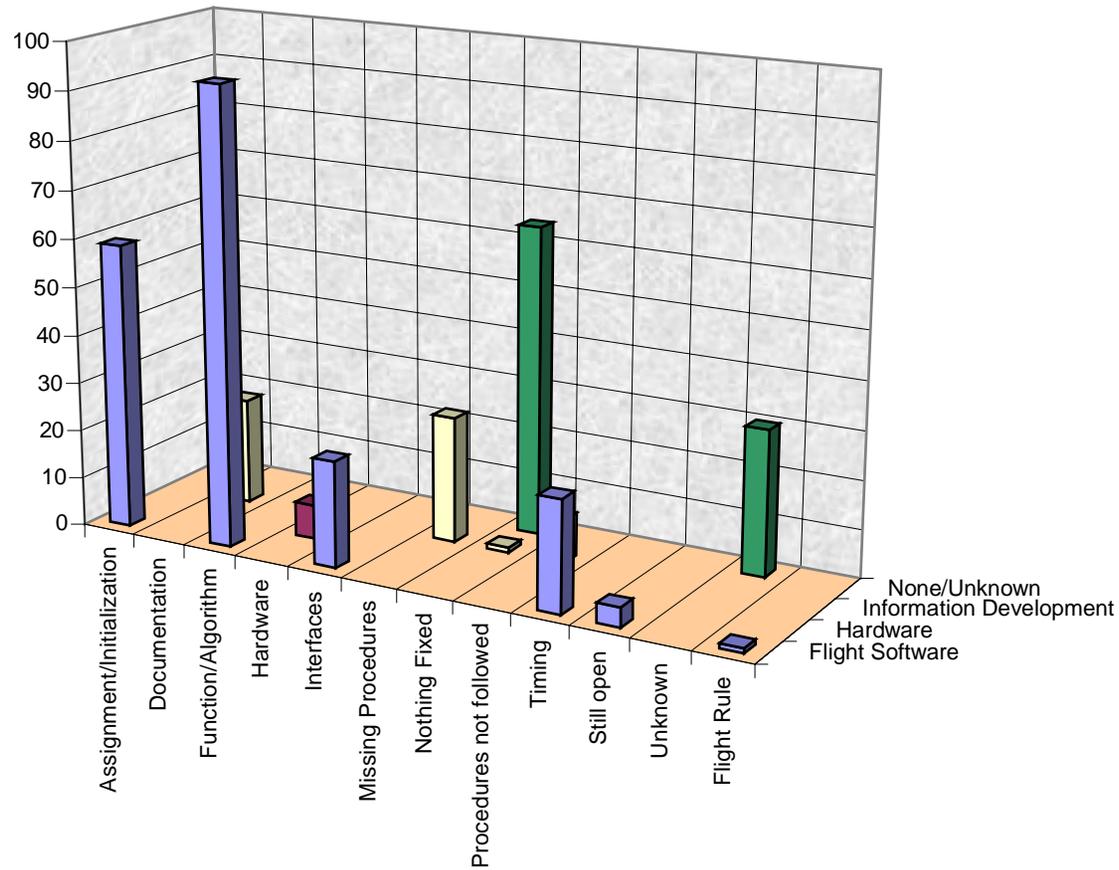




Defect Patterns in Testing



Distribution of Types by Target



Defect Patterns in Testing



- **2 basic kinds of requirements discovery:**
 - Discovery of new (previously unrecognized) requirements or requirements knowledge
 - Discovery of misunderstandings of (existing) requirements
- **Reflected in ODC Target (what gets fixed) and ODC Type (nature of the fix)**
 - Software change (new requirement allocated to software)
 - Procedural change (new requirement allocated to operational procedure)
 - Document change (requirements confusion addressed via improved documentation)
 - No change needed (works OK as is; user was just confused)

1. Incomplete requirements, resolved by change to software:

New software requirement became evident: initial state of a component's state machine must wait for the associated motor's initial move to complete

2. Unexpected requirements interaction, resolved by changes to operational procedures:

Software fault monitor issued redundant off commands from a particular state (correct but undesirable behavior). Corrective action was to prevent redundant commands procedurally by selecting limits that avoid that state in operations

3. Requirements confusion, resolved by changes to documentation

Testing personnel incorrectly thought heaters would stay on as software transitioned from pre-separation to Entry/Descent mode; [clarified in documentation](#).

4. Requirements confusion, resolved without change

Testers assumed commands issued when component was off would be rejected, but commands executed upon reboot. [No fix needed](#); behavior correct.

Defect Patterns in Operations



<i>Examples of Unexpected ISA patterns:</i>	<i>Process Recommendation:</i>	<i>Example (from spacecraft):</i>
22% of critical ISAs had <u>ground software</u> as Target (fix)	Software QA for ground software	Unable to process multiple submissions. Fixed code.
23% of critical ISAs had <u>procedures</u> 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 <u>Data access / delivery</u> 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.

JPL *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

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 problem reporting system 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)
 - 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



Challenges Ahead

What should we do in the long run?

- **Partial automation of classification is possible:**
 - Customize pull-down menus (Pick-Lists) of the problem reporting system
 - Train users on ODC
 - Improve fidelity of raw data
 - Automation supports timely feedback to projects
- **Product line perspective**
 - Problem Reports predict problems in future similar systems
 - How can we better mine the problem database to prevent defect recurrence
- **Integration with run-time monitoring**
 - ODC identifies patterns of concern
 - Run-time monitoring can use these patterns
 - Automate defect prevention



Backup Slides

For More Information



- **“Operational Anomalies as a Cause of Safety-Critical Requirements Evolution,” R. Lutz and C. Mikulski, *The Journal of Systems and Software*, 65 (2003) (available on-line 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.**

ODC Classification Sample



Activities	Triggers	Targets	Types
System Test	<ul style="list-style-type: none"> Software Configuration Hardware Configuration Start/Restart, Shutdown Command Sequence Test Inspection/Review 	Ground Software	<ul style="list-style-type: none"> Function/Algorithm Interfaces Assignment/Initialization Timing
Flight Operations	<ul style="list-style-type: none"> Recovery Normal Activity Data Access/Delivery Special Procedure Hardware Failure 	Flight Software	<ul style="list-style-type: none"> Function/Algorithm Interfaces Assignment/Initialization Timing Flight Rule
Unknown	Unknown	Build /Package	<ul style="list-style-type: none"> Install Dependency Packaging Scripts
		Ground Resources	Resource Conflict
		Info. Development	<ul style="list-style-type: none"> Documentation Procedures
		Hardware	Hardware
		None/Unknown	<ul style="list-style-type: none"> Nothing Fixed Unknown