



Requirements Discovery During the Testing of Safety-Critical Software

Robyn Lutz
Jet Propulsion Lab
and Iowa State University
rlutz@cs.iastate.edu

Inés Carmen Mikulski
Jet Propulsion Lab
ines.c.mikulski@jpl.nasa.gov

ICSE 2003
Portland, OR May 8, 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 authors' research is supported in part by National Science Foundation Grants CCR-0204139 and CCR-0205588.



Topics

- Motivation & related work
- Problem
- Approach
- Results and examples
- Lessons Learned



Motivation and Related Work

- Goal: reduce critical anomalies after launch
- Known:
 - Incomplete or misunderstood requirements cause testing defects [Gardiner, '99; Lauesen, Vinter, '01; Leszak, Perry, Stoll, '00; Lutz, '93]
 - Incomplete or misunderstood requirements cause accidents [Hanks, Knight, Strunk '01; Weiss, Leveson, Lundqvist, Farid, Stringfellow '01]

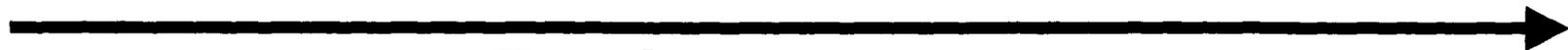


Problem

Previous focus



Requirements discovery



Reqmts Design **Testing** Deployment Operations



Approach

- Analyze problem reports from integration and system testing to better understand how requirements are discovered; use findings to reduce anomalies post-launch
- Mars Exploration Rovers
 - Launch June, 2003
 - ~300 K LOC flight software
 - ~400 software requirements
- Problem Reports (PRs)
 - Written by test teams
 - Standard form
 - Mined institutional, web-based database of PRs
 - 171 PRs analyzed in ICSE paper; now ~450



Approach

- Adapted Orthogonal Defect Classification (ODC) [Chillarege et al., 92] to spacecraft domain
- “Extracts signatures from defects”
- 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

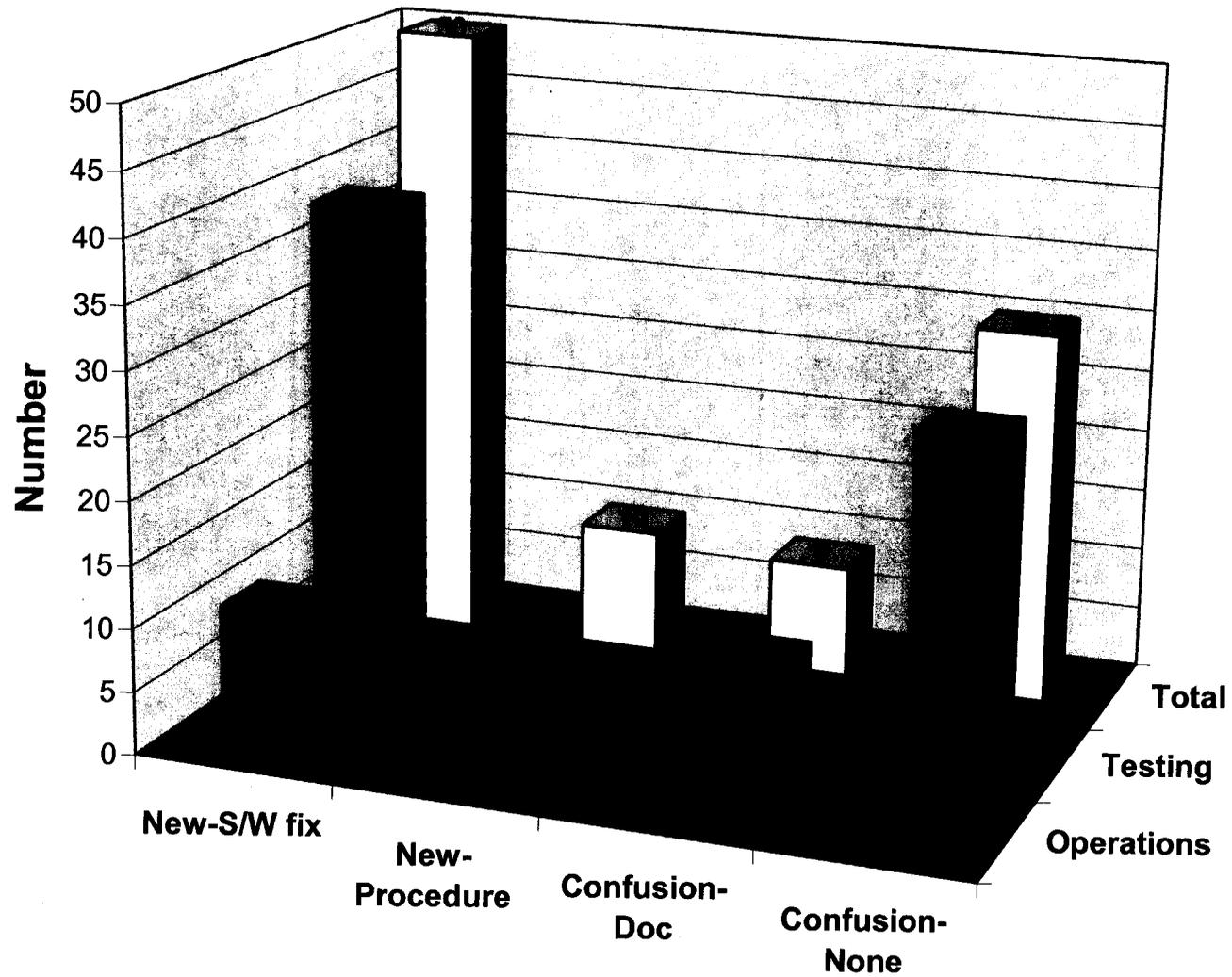


Results

- 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):
 - 1. Software change** (new requirement allocated to software)
 - 2. Procedural change** (new requirement allocated to operational procedure)
 - 3. Document change** (requirements confusion addressed via improved documentation)
 - 4. No change needed**



Results: What the PRs show





Results: Examples

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



Results: Examples

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.



Lessons Learned

- Testing is “crystal ball” into operations
 - False-positive PRs (behavior correct but unexpected) provide insights into requirements confusions
 - If software behavior surprised testers, it may surprise operators
- “No-Fix” decision may 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