How Requirements Knowledge Grows in Built Systems

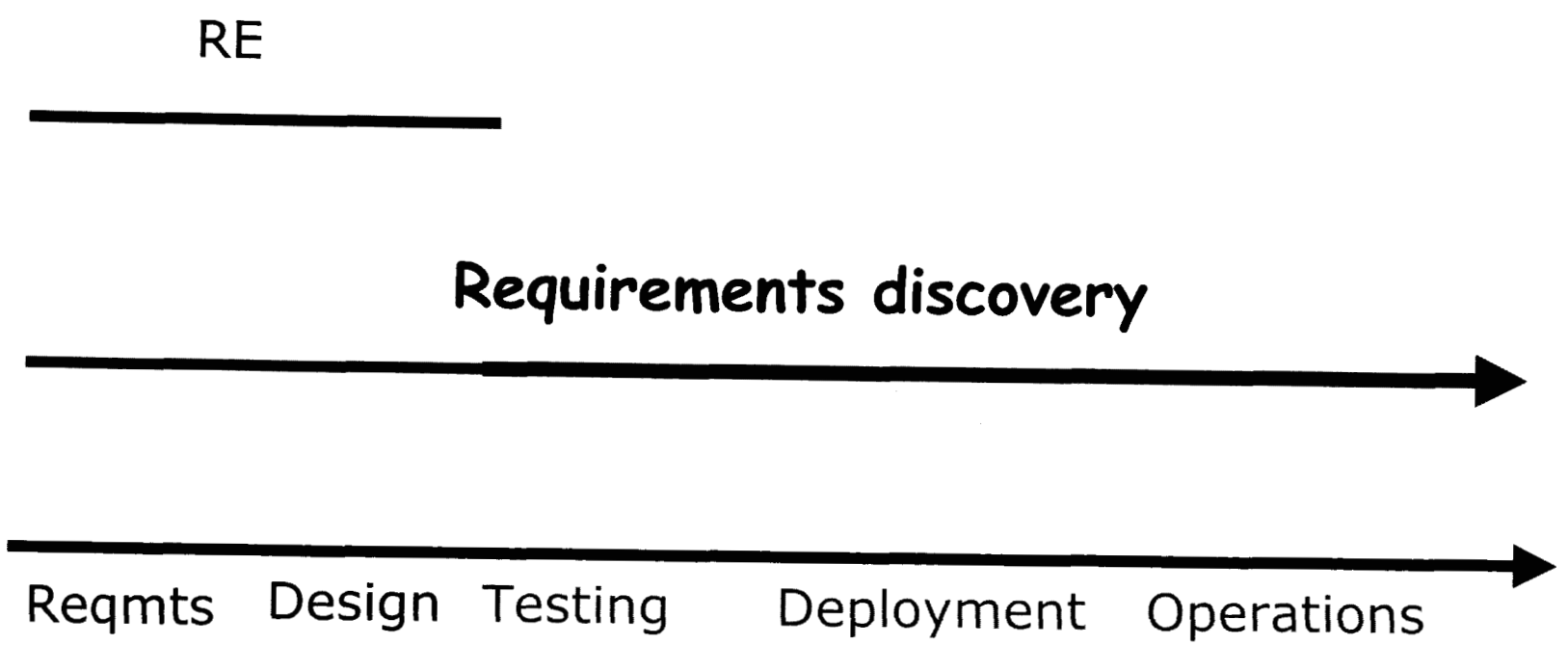
Robyn Lutz
Jet Propulsion Lab/Iowa State University
Joint work with Carmen Mikulski, JPL

IFIP WG2.9

February 16-19, 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.

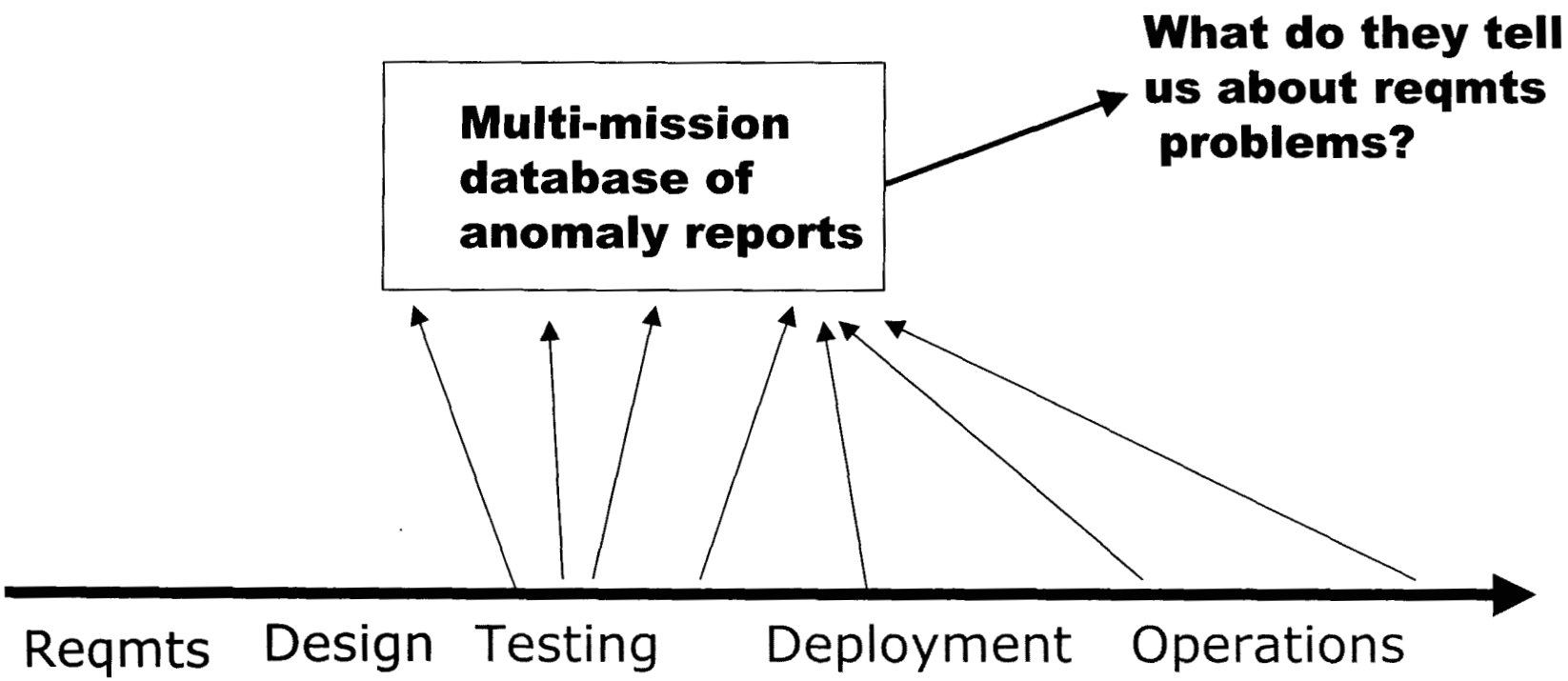
Problem


RE
Requirements discovery
Reqmts
Design
Testing
Deployment
Operations

Overview

- Incomplete requirements and requirements misunderstandings cause many serious anomalies in testing and operations
- Current work: To better understand and validate requirements discovery in these later phases
- Context: Extend RE techniques that support requirements discovery to later phases
- Goal: Reduce critical anomalies post-launch

Approach


Multi-mission database of anomaly reports
What do they tell us about reqmts problems?
Reqmts
Design
Testing
Deployment
Operations

Approach

- Analyze software anomaly reports from integration testing, system testing, and operations
  - Testing: 326 reports from Mars Exploration Rovers Integration & System testing (launch 5/03, 6/03)
  - Operations: 189 reports from 7 launched spacecraft
- Adapted ODC (Orthogonal Defect Classification) [Chillarege et al., 92] to spacecraft domain
  - Describes Activity, Trigger, Target, and Type of each anomaly

Approach

- Caveat: not maintenance, not requirements evolution: requirements are here *essential* for *current* system
- Caveat: analyzed critical reports (project-assigned rating) from operational spacecraft; no criticality rating available for most MER reports

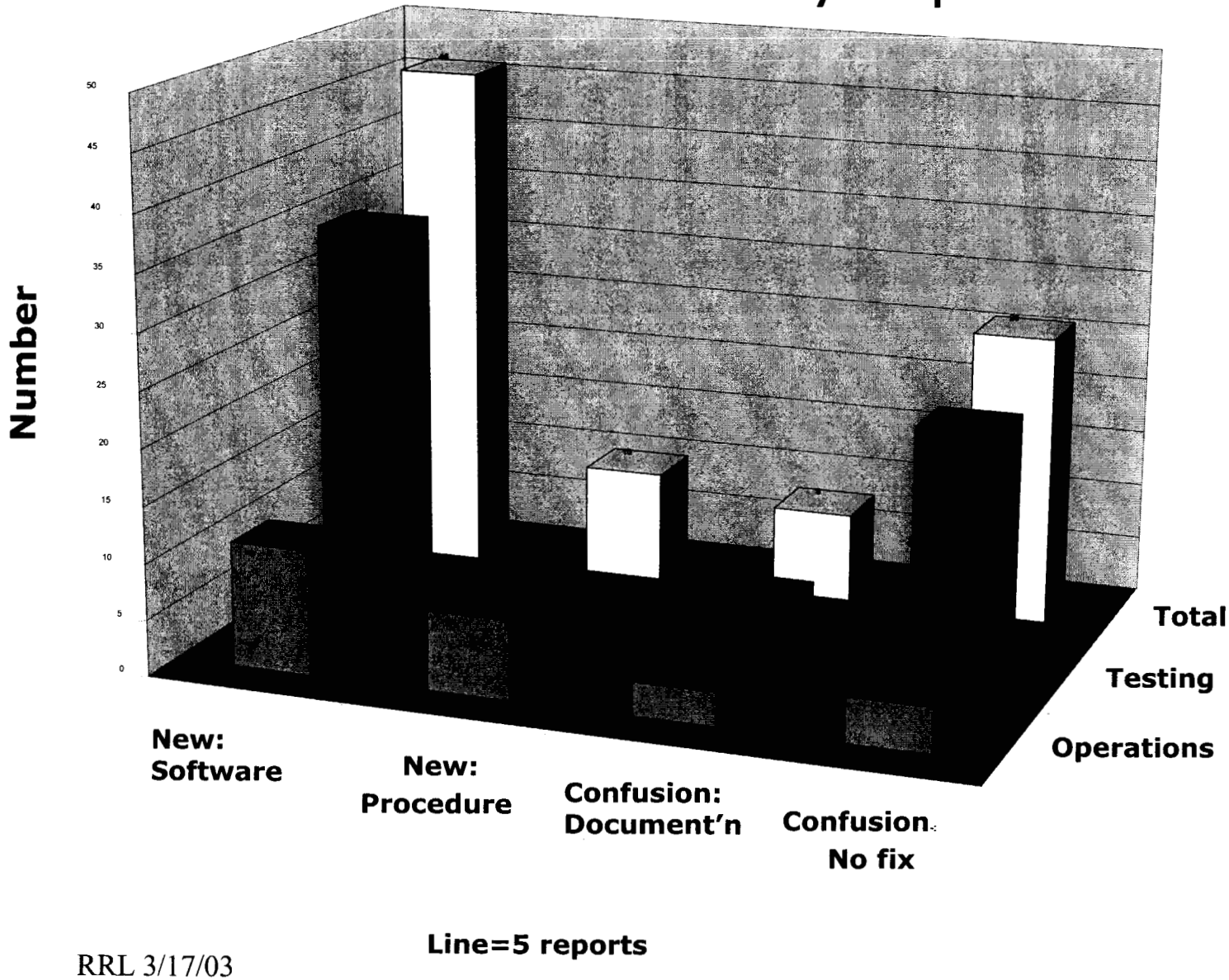
Related work

- Requirements evolution
  - Early identification of latent or changing requirements pays off [Anton & Potts, Fickas & Feather, Zowghi et al.]
- Maintenance
  - Focuses on management of proposed, usually optional, requirements changes [Bennett et al., Harker et al., Lam & Loomes]
- Defect analysis
  - Many defects are requirements-related [Leszak et al., Lutz, Lauesen & Vinter]
- High-assurance requirements
  - Misunderstood requirements cause accidents [Leveson et al., Knight et al]

What the anomaly reports show

- 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)

What the anomaly reports show



Examples (1)

- Incomplete requirements, resolved by change to software:
  - Testing anomaly: new requirement became evident for initial state of a component's state machine to wait for completion of the associated motor's initial move to complete
  - Operational anomaly: new requirement became evident for software to compensate for noisy transducers that were causing frequent component resets

Examples (2)

- Unexpected requirements interactions, resolved by changes to operational procedures:
  - Testing anomaly: 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
  - Operational anomaly: when aerobraking maneuver erroneously performed twice, discovered that due to software being loaded to memory "too soon"; fixed by adding a procedure to prevent recurrence of configuration problem.

Examples (3)

- Requirements confusion, resolved by changes to documentation
  - Testing anomaly: Testing personnel incorrectly thought heaters would stay on as software transitioned from pre-separation to Entry/Descent mode; clarified in documentation.
  - Operational anomaly: Drop in battery power occurred when operational personnel misunderstood required behavior initiated by a command; clarified required behavior and associated command in operational flight rule.

Examples (4)

- Requirements confusion, resolved without change
  - Testing anomaly: Testers assumed commands issued when component was off would be rejected, but commands executed upon reboot. No fix needed; behavior correct.
  - Operational anomaly: Operational personnel assumed "stow" meant "close instrument cover when instrument not in use" and "deploy" meant "open instrument cover when instrument will be used." In fact, "stow" opens the cover, and "deploy" closes it. No fix needed.

Requirements confusion

- Does "no fix" suffice?
  - Mismatch between correct and expected behavior
  - Possible recurrence in operations with serious consequences
  - Long-lived systems→ more turnover, loss of requirements knowledge

Related work

- Goal-obstacle analysis [van Lamsweerde & Letier]
  - Anomaly reports document obstacles encountered in trying to achieve requirements
  - Requirements confusions are instances of their "Wrong Belief" class of obstacles
  - Found instances of 4 of 6 "Wrong Belief" subclasses
  - Found possible additional "Wrong Belief" subclass: "Information Not Used"
  - "no change needed" anomalies were instances of their "do-nothing" strategy

"Wrong Belief" instances (from 23 critical ops anomalies with "nothing fixed")

Wrong Info Provided

Info Corrupted

Info Outdated

Info Forgotten

Wrong Inference

Info Confusion

Also: Info Not Used

Recurring sources of requirements confusion

- Counters
  - High-water marks, persistence, timers (relative vs. absolute); resetting of counters
- States
  - Unavailable vs. unresponsive; relationship between component states & software modes
- Redundant data
  - Effect on system state
- Warning messages
  - Passive (info only) or active (initiating response) indications

Results --> Recommendations to projects

1. Similar discovery mechanisms in testing and operations

1. View testing as dry-run of ops

2. Agent = procedure for newly discovered requirements on software interactions

2a. Recognize tradeoff and risk

2b. Provide traceability

2c. Make sure procedures needed in past are in place

3. Feed-forward focus in reporting

3. Capture/Store/Retrieve

Results --> Recommendations to projects

4. False-positive reports provide unique insight

4. Raise bar for closure: if situation can recur & confusion can recur & fault-protection or critical phase → document, train

5. Some requirements confusion types recur

5. Identify, resist, anticipate effect

Questions

- What RE techniques can help in later phases?
  - Given: incomplete specifications,
  - No modeling of current system exists,
  - Testing **is** requirements elicitation,
  - Integration testing for build i drives requirements for build i+1, and
  - Requirements emerge from operations
- How best to reuse requirements knowledge (rationale, interactions, constraints, dependencies) on similar systems?
  - Domain-specific checklists extracted from defect reports?
  - Testbed of historically troublesome scenarios from defect reports?
  - **Risk: inappropriate reuse**

Questions

- Do requirements confusions that occur in testing recur in operations?
  - Anomaly reports suggest yes ("rediscovery of problem")
  - How to trace from testing into operations?
- Can we anticipate (pre-identify) sources of requirements confusions?
  - Anomaly reports suggest some recurring patterns of misunderstanding
  - How to avoid in practice?