California Institute of Technology

Using ODC for Empirical Evaluation of Software Anomalies

Dr. Robyn Lutz and Carmen Mikulski
rlutz at cs.iastate.edu
515-294-3654
http://www.cs.iastate.edu/~rlutz

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The work was sponsored by the NASA Office of Safety and Mission Assurance under the Software Assurance Research Program led by the NASA Software IV&V Facility. This activity is managed locally at JPL through the Assurance Technology Program Office

JSC, 4/14/04

Topics

- Overview of ODC
- Sample Results
  - Mars Exploration Rover
  - Deep Impact & Stardust
  - Technology Infusion
  - Dissemination
- Benefits for Moon/Mars Human Space Flight Missions

Overview

California Institute of Technology

- Goal:
  1. To characterize software anomalies, using data from multiple spacecraft projects, by means of a defect-analysis technology called *Orthogonal Defect Classification (ODC).*
  2. To support transfer of ODC to NASA projects through applications and demonstrations.
- Approach:
  1. Analyze anomaly data using adaptation of *Orthogonal Defect Classification (ODC)* method
     - Developed at IBM; widely used by industry
     - Quantitative approach
     - Used here to detect patterns in anomaly data
     - More information at http://www.research.ibm.com/softeng
  2. Adapt ODC for NASA use and apply to NASA projects

Overview: *Status*

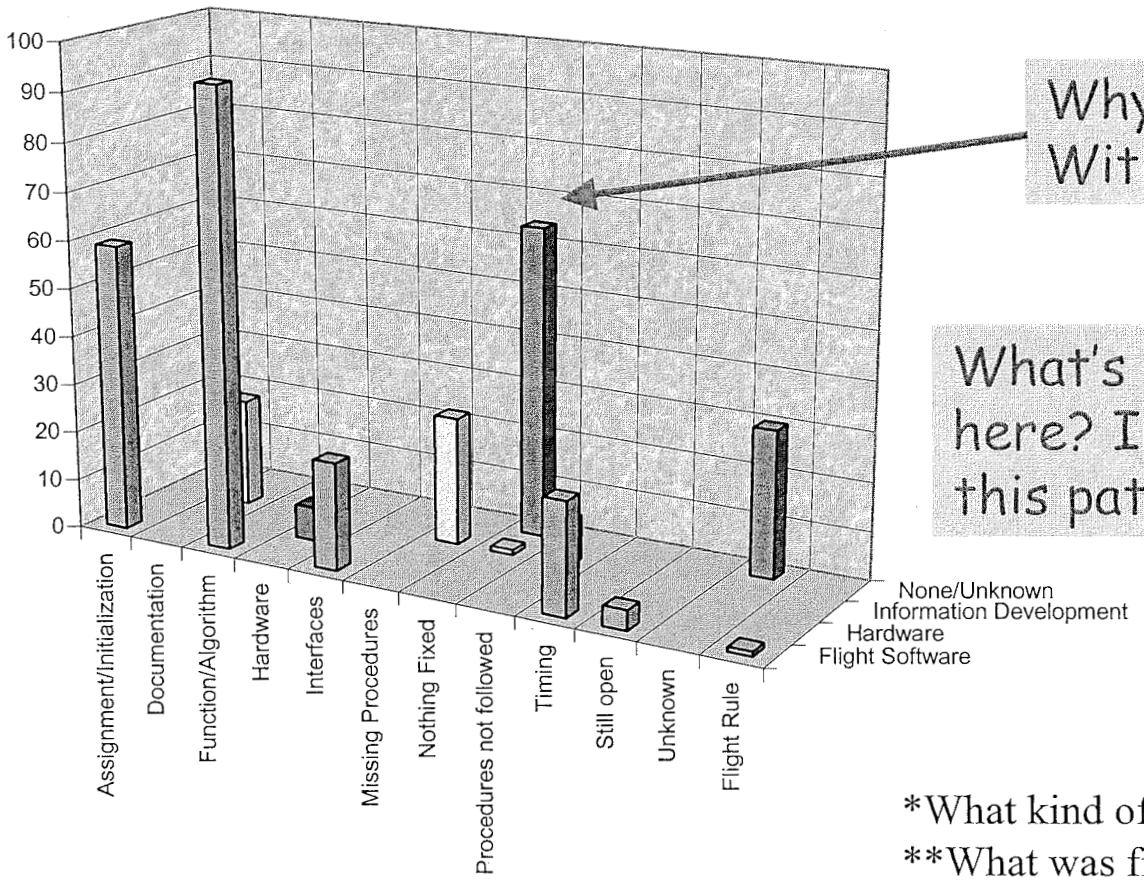
California Institute of Technology

- Previous work used ODC to analyze safety-critical *post-launch* software anomalies on 7 spacecraft.
- FY'03 task extended ODC work to *pre-launch development and testing* (Mars Exploration Rover testing, Deep Impact, contractor-developed software) and supported *technology infusion*
- Adapted ODC categories to spacecraft software at JPL:
  - Activity: what was taking place when anomaly occurred?
  - Trigger: what was the catalyst?
  - Target: what was fixed?
  - Type: what kind of fix was done?

Example: Lesson Learned from ODC

California Institute of Technology

MER software defects; distribution of Types* x Target** (6/03)



Why so many With "No fix"?

What's going on here? Investigate this pattern further!

*What kind of fix?

**What was fixed?

Example: Lesson Learned from ODC

California Institute of Technology

"Crystal Ball":
If software behavior surprised testers, it may surprise operators

- Testing problem 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
- 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

ODC Values for Test Anomalies

	Activity			
Trigger	Build Test	Avionics I&T	ATLO	Unkwn
Capability Invocation	X	X	X	X
Command Execution	X	X	X	X
HW Configuration	X	X	X	X
HW-SW Interaction		X	X	X
Inspection/Review	X	X		
Recovery	X	X	X	X
SW Configuration	X	X	X	X
Special Procedures	X	X	X	X
Start/Restart, Shutdown, Reboot	X	X	X	X
Unknown	X	X	X	X
Workload & Stress			X	

ODC Values for Test Anomalies

California Institute of Technology

Target	Type
	Assignment/Initialization
	Flight Rule
Ground Software	Function/Algorithm
	Interfaces
	Testbed environment
	Timing
	Assignment/Initialization
	Flight Rule
Flight Software	Function/Algorithm
	Interfaces
	Timing

Build/Package	Install Dependency
	Scripts
	Version conflict
Info. Development	Documentation
	Missing procedures
	Procedures not followed
Hardware	Hardware
None/Unknown	Nothing Fixed
	Unknown

Use on MER of ODC* defect analysis

Testing Problem Reports

Problem Report file for MER

Classify MER test problem reports per ODC Categories

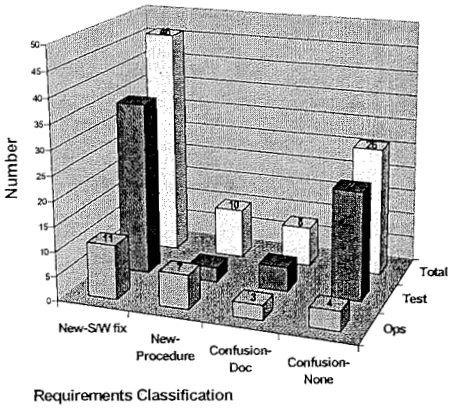
➢Where are the spikes?
➢Phase-by-phase deltas?
➢Activity/Trigger/Target/Type look nominal?

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

Discover defect patterns in testing of interest to MER

➢How many problem reports result in requirements discovery?
➢Improvement release-by-release uneven: why?
➢Many closed with no fix: why?

MER use: Identify patterns of concern for more investigation

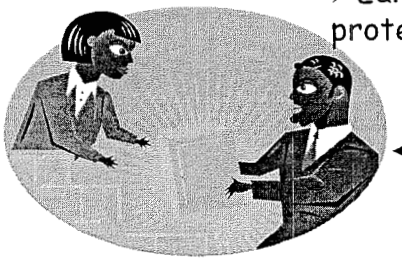


Perform Root Cause Analysis on subset

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

Develop and package recommendations for MER

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



MER use: Implement/defer recommendations

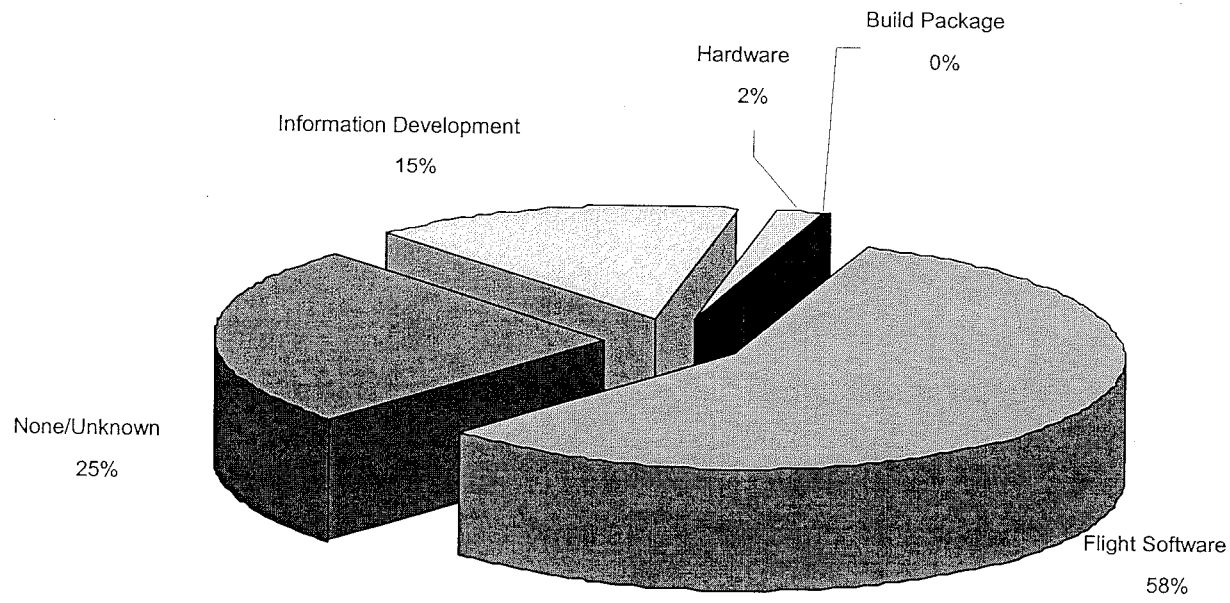
*ODC = Orthogonal Defect Classification technique [IBM]

Results: *MER*

California Institute of Technology

- Collaborated with Mars Exploration Rover to experimentally extend ODC approach to pre-launch software problem/failure testing reports (~800)
  - Adjusted ODC classifications to testing phases
  - Institutional defect database → Access database of data of interest → Excel spreadsheet with ODC categories → Pivot tables with multiple views of data
  - Frequency counts of Activity, Trigger, Target, Type, Trigger within Activity, Type within Target, etc.
  - User-selectable representation of results support tracking trends and progress:
    - Graphical summaries
    - Comparisons of testing phases
  - Provides rapid quantification of data
  - Project provides feedback/inquiries on our monthly deliverables of results and on our draft reports/paper

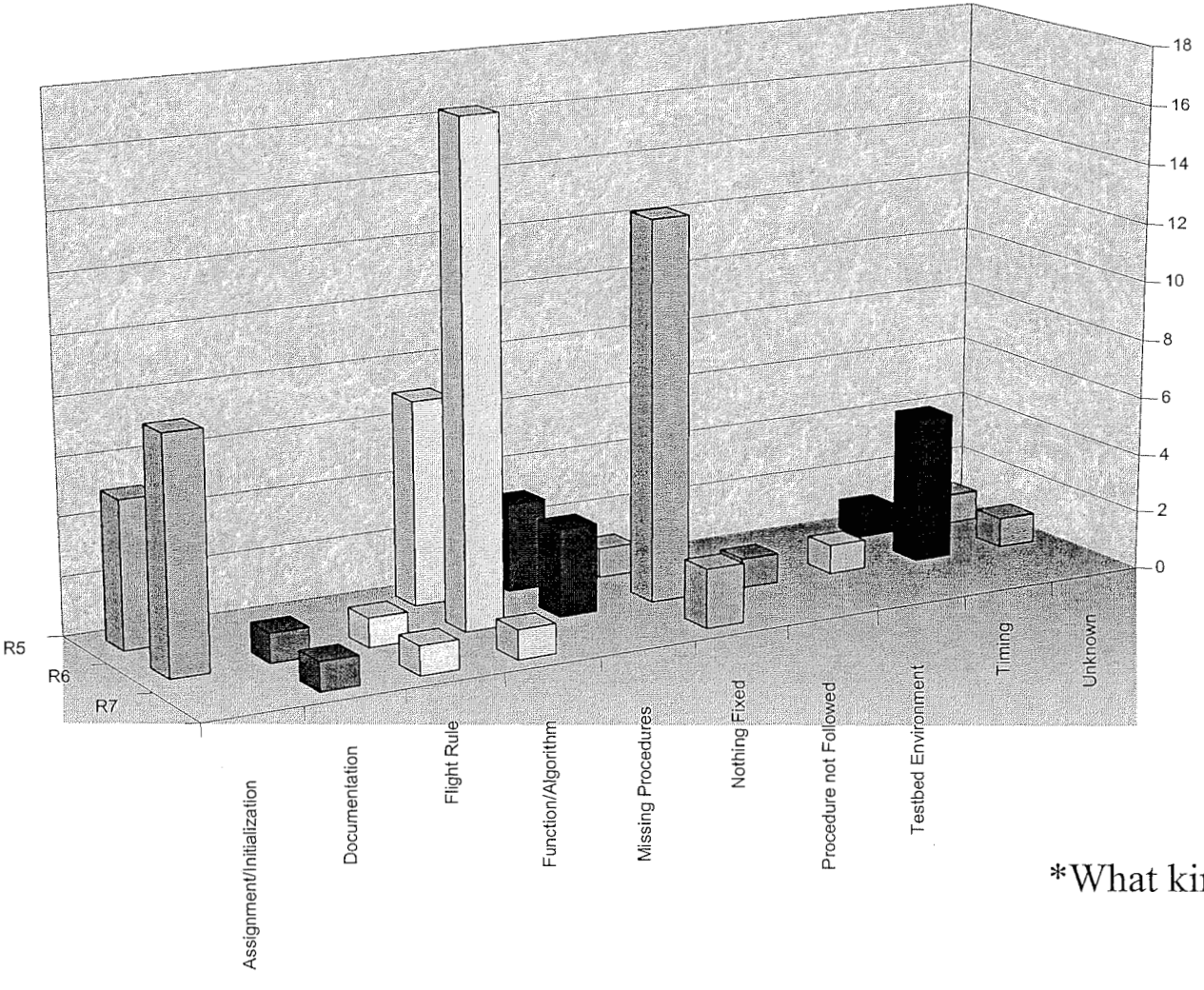
Results: *MER software defects*



Results: *MER software defects*

California Institute of Technology

Distribution of Types* by Build



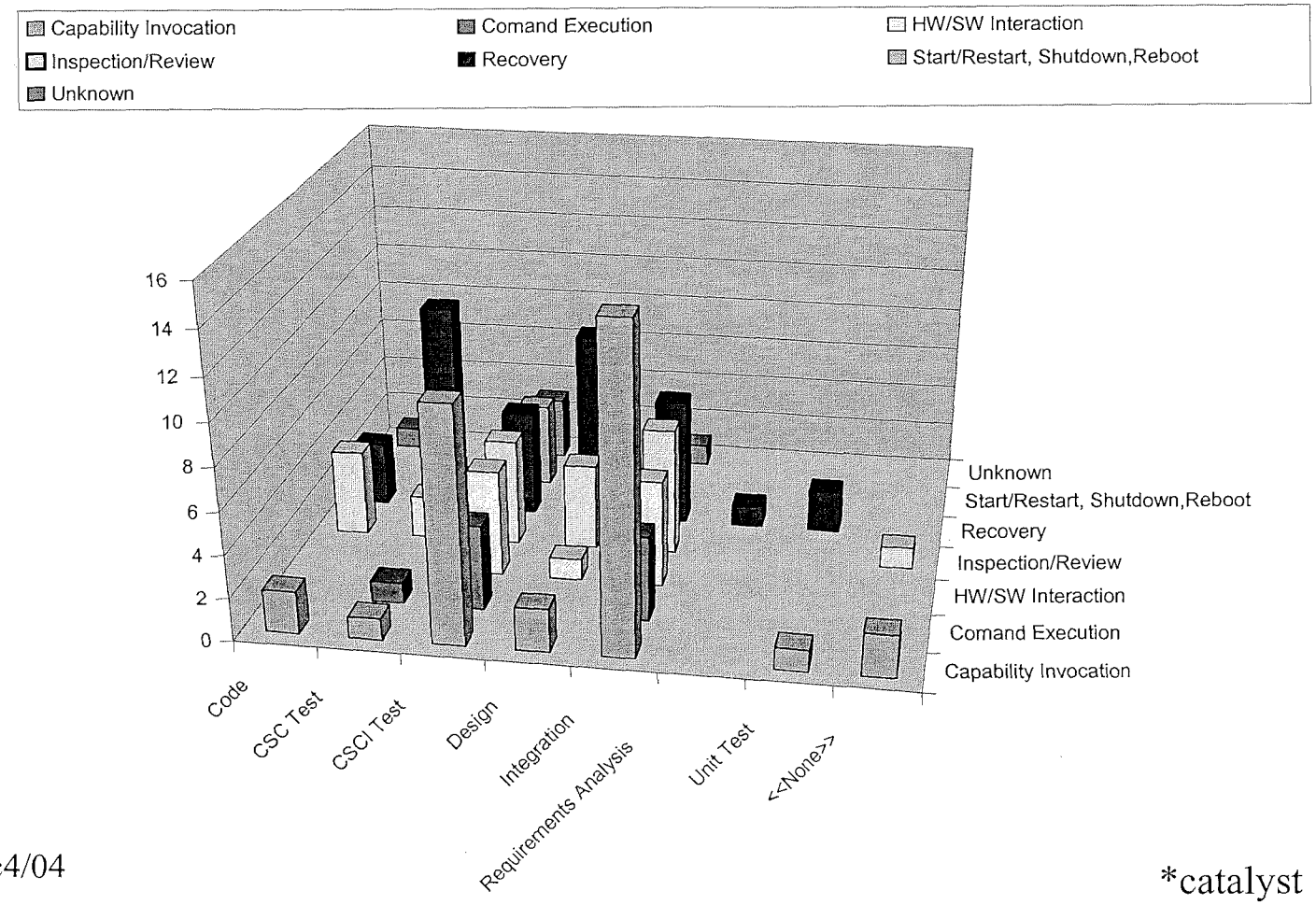
*What kind of fix?

Results: *Deep Impact & Stardust*

- Extended ODC approach to development-phase SCRs (Software Change Reports)
  - Classified initial set of 121 critical DI SCRs written by Ball (with highest cause-corrective action/failure effect ratings)
- Extended ODC approach to contractor software
  - Classified 101 software problem reports written by Lockheed Martin during System/ATLO testing of Stardust.

Results: *Deep Impact*

Distribution of Trigger* by Phase



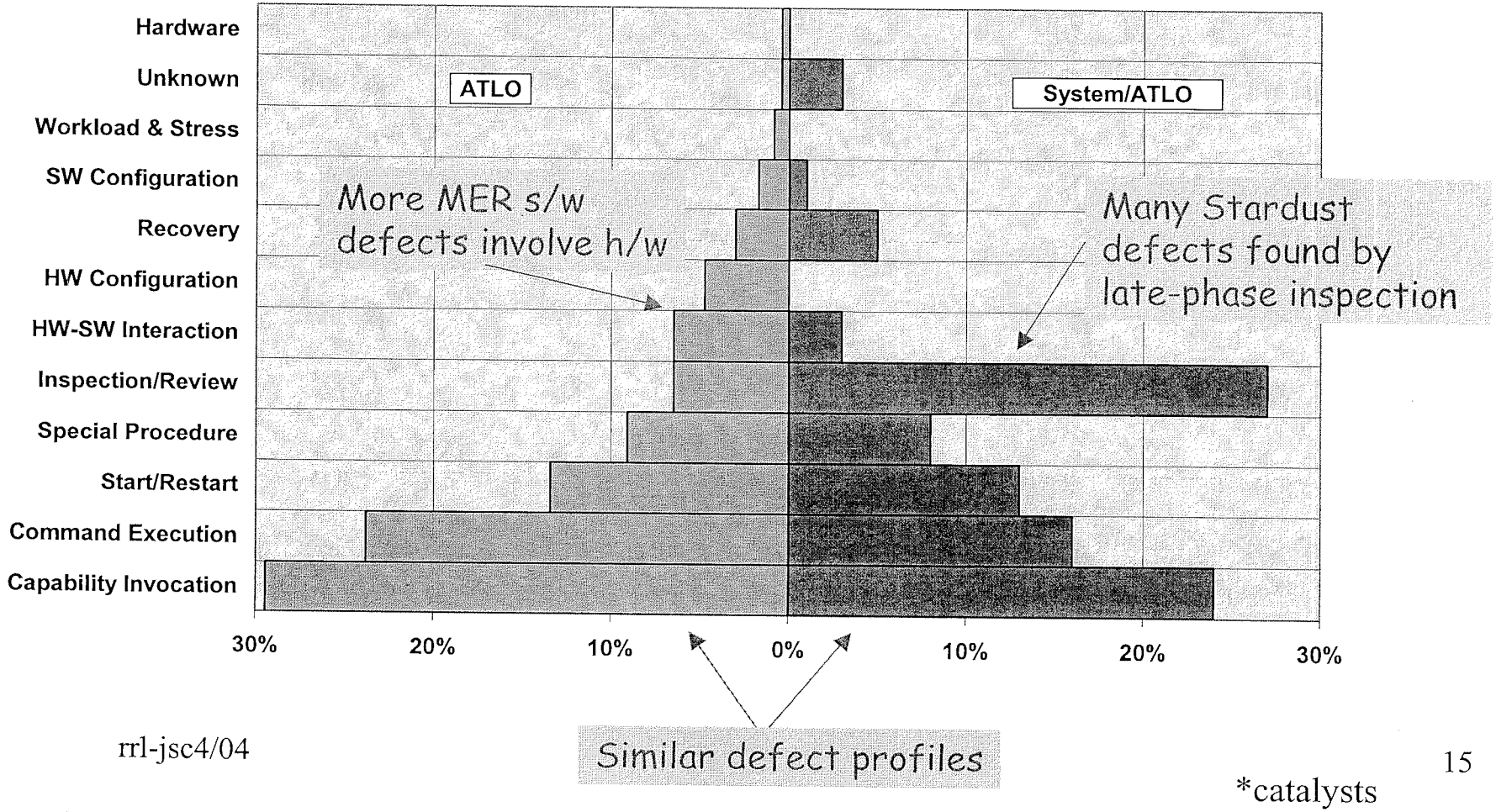
California Institute of Technology

*catalyst

Results: *MER/Stardust*

California Institute of Technology

Comparative Distribution of Triggers*

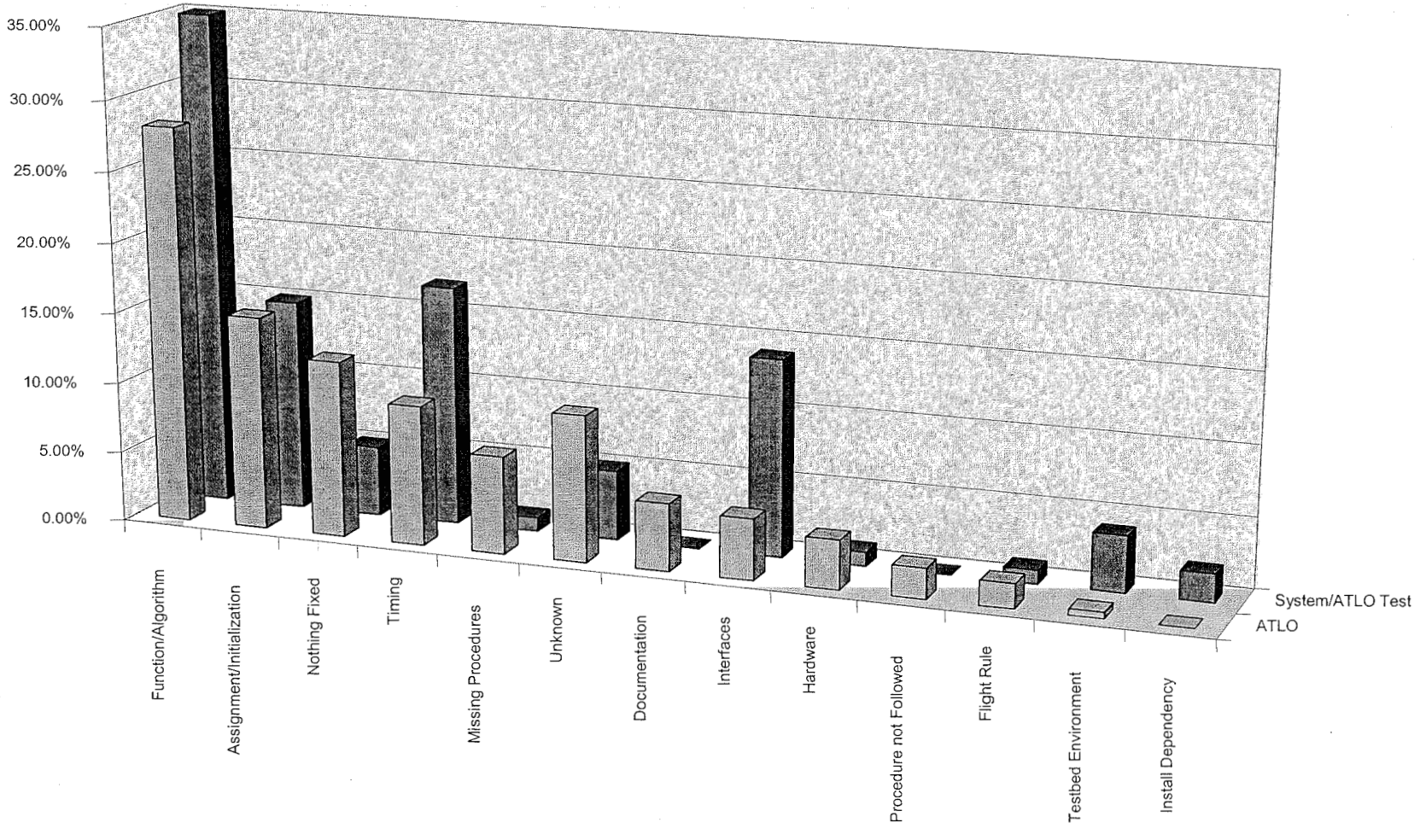


*catalysts

Results: *MER/Stardust*

California Institute of Technology

Comparative Distribution of Types*



*What kind of fix?

Results: *Lessons Learned*

California Institute of Technology

- **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: 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.

Results: *Infusion*

- ODC selected as one of the 2004 NASA Research Infusion projects (ARC, T. Pressburger, Lead)
  - Antenna Beam Waveguide Retrofit project (JPL) (S. Morgan, Proj. Mgr)
- ODC selected as a candidate defect-analysis tool by JPL's Software Quality Improvement task
- Have picked up use of ODC:
  - "Linking features to failures" (Garnett & Lesch, IV&V)
  - Low Temperature Microgravity Physics Facility (JPL); set up to use ODC but project cancelled
  - Common Operating Environment project (Defense Information Infrastructure) (JPL)

Results: *Dissemination*

California Institute of Technology

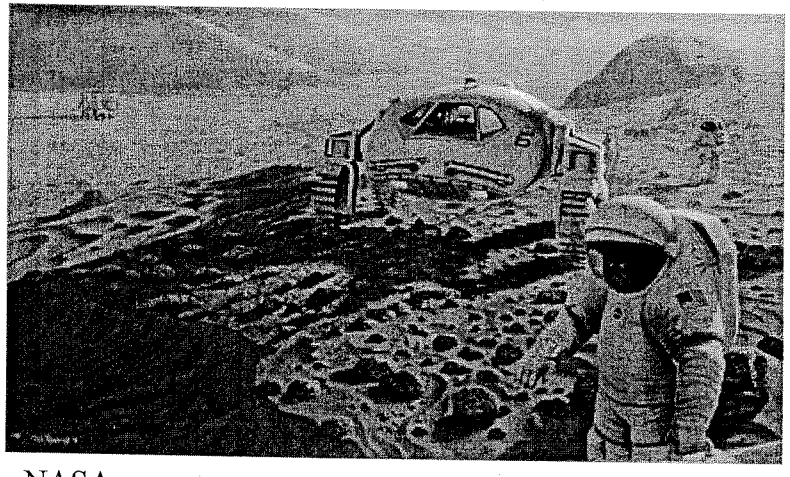
- ODC for operational anomalies:
  - ODC profile of operational defects: *IEEE Transactions on Software Engineering*, March, 2004.
  - How operational anomalies drive requirements evolution: *Journal of Systems and Software*, Feb. 2003
- ODC for testing anomalies:
  - the 4 mechanisms involved in requirements discovery during testing, *ICSE 2003* (Int'l Conf on S/W Eng)
- ODC for anomalies in testing & operations:
  - *IEEE Software*, March/April, 2004
  - Found same 4 mechanisms in operations as in testing, *RE 2003* (Int'l Conf on Requirements Eng); best experience paper
- ODC-based analysis:
  - Patterns of defect data, *SMC-IT* 2003 (Space Mission Challenges)
  - Automated analysis of ODC defect data (T. Menzies) at *SEKE 2003* (Int'l Conf S/W Eng & Knowledge Eng)

Benefits for Moon/Mars Human Space Flight Missions

California Institute of Technology

ODC can contribute to the success and safety of the human flight program by improved understanding of defects during development, testing and operations

NASA



NASA

Sustained human exploration missions will require innovative defect management approaches that feed-forward defect data to related missions

Benefits for Moon/Mars Human Space Flight Missions

California Institute of Technology

Mission Challenges:

1. Sustained missions with dynamic defect profiles
2. Incremental and fleet missions
3. Very large amount of contractor software for integration & testing
4. Rapid evolution in technologies and knowledge

ODC Features:

1. ODC avoids "can't see the forest for the trees" overload
2. ODC supports product line approach
3. ODC provides development-through-operations defect management
4. ODC allows us to continually test our operational assumptions

Conclusion: ODC Strengths

California Institute of Technology

High value of adoption

- Identifies problem patterns
- Multi-project ODC baseline provides guidance to future projects
- Can answer current project's questions regarding defects

- "Smooth adoption curve"
  - High usability
  - Flexible visualization & browsing options
  - Acceptable level of effort; can use existing fields in existing problem-reporting system
  - Partial automation (project selectable)
- Extensive NASA Experience
  - ODC for development, testing, and operations phases
  - ODC for project, contractor software

Conclusion: ODC Strengths

- Quantitative foundation
  - Metrics for process improvement
  - Metrics for tracking project quality
  - Metrics for assessment of reused software
- Future focus
  - Equips us with a methodology to continue to learn as space-exploration missions and processes evolve