

Formalized Pilot Study of Safety-Critical Software Anomalies
Final Report
December 30, 2002

Robyn R. Lutz
Inés Carmen Mikulski
Jet Propulsion Laboratory

This work 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 work is managed locally at JPL through the Assurance Technology Program Office.

Summary

This report describes both the research techniques and the application results from the analysis of safety-critical software anomalies recorded post-launch on seven spacecraft: Galileo, Mars Global Surveyor, Cassini/Huygens, Deep Space 1, Mars Climate Orbiter, Mars Polar Lander, and Stardust. The process involved the adaptation of an existing defect-analysis methodology, Orthogonal Defect Classification, to spacecraft applications. This report also describes initial results from a feasibility study of adapting this ODC technique to analyze software problem reports generated during testing on the Mars Exploration Rovers. Both the approach and the results are presented here. The goal is to reduce the number of safety-critical software anomalies that occur during flight by providing a quantitative analysis of previous anomalies as a foundation for process improvement.

Introduction

Advances in NASA's capability to produce quality software that contributes to safe, reliable systems depend in part on our ability to more precisely characterize areas needing improvement. Analysis of software anomalies is such an area. This work characterizes the common causes of safety-critical in-flight software anomalies using operational data from seven spacecraft. The analysis was done using an adaptation of a defect-analysis technology, called Orthogonal Defect Classification (ODC), developed by IBM and used by industries such as Bellcore and Motorola. Since the goal of the research is to provide a sound, quantitative foundation to enable improvements, a formalized pilot study approach (the rigorous Glass criteria) was used.

Results from analysis of the safety-critical software anomalies occurring post-launch on seven spacecraft were delivered to the IV&V Facility. In response to queries regarding whether a similar adaptation of ODC could provide analysis of pre-launch software anomalies, a feasibility study began in collaboration with the Mars Exploration Rover project. In this study problem reports generated during integration and system testing were analyzed using a similar adaptation of ODC.

This final report provides the following results not previously delivered to the IV&V Facility:

1. **Definition of ODC Classification Scheme** Adapted to Post-Launch Anomalies
2. **Definition of ODC Classification Scheme** Adapted to Software Testing Problem Reports
3. **Excel Database** of ODC Classification of 199 post-launch critical software Incident/Surprise/Anomaly reports
4. **Excel Database** of ODC Classification of testing software developmental problem failure reports and software problem failure reports
5. **Pivot Table Summary of Results** from Analysis of Post-Launch Anomalies
6. **Pivot Table Summary of Results** from Analysis of Some Testing Problem Reports
7. **Paper Describing Results** from Preliminary Analysis of Some Testing Problem Reports (accepted to IEEE International Conference on Software Engineering, 2003)
8. **Process Recommendations** Resulting from ODC Analysis of Safety-Critical Post-Launch Software Anomalies from Seven Spacecraft

Section 1. Definition of ODC Classification Scheme Adapted to Post-Launch Anomalies

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

Section 2. Definition of ODC Classification Scheme Adapted to Software Testing Problem Reports

This document contains the definitions of PFR classification, which apply the Orthogonal Defect Classification (ODC) to the MER PFRs encountered during the three level of testing (Build, I&T, and ATLO).

The classifications are being customized for MER during the tests phases in an effort to provide information to the different test levels ...

Classification Summary:

Trigger	Activity			
	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	

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

Target	Type
	Version conflict
Info. Development	Documentation
	Missing procedures
	Procedures not followed
Hardware	Hardware
None/Unknown	Nothing Fixed
	Unknown

ACTIVITY

Activity: Is the test level being performed when the defect was found?

Activity	Description of Activity
Build Test	
Avionics I&T Test	
ATLO Test	
Unknown	This activity will be selected when the description of the problem does not give a clear indication on the type of activity being performed.

TRIGGER

Trigger: The environment or condition that had to exist for the defect to surface. The environment or condition that was the catalyst for the anomaly [ODC].

Suggested description by Karen: "Test objective of the test being run when the defect was discovered. This objective is not necessarily related to the defect or the cause of the anomaly."

Trigger	When to Select this Trigger
Capability Invocation	The error was detected while testing the response from a series of related commands. This includes lack of or incorrect telemetry
Command Execution	The error was detected while testing the response from a single command.
Inspection/Review	A problem have been found out by inspection of: <ul style="list-style-type: none"> • Code • Test results
HW Configuration	A S/W or H/W error was detected as a result of testing a particular H/W configuration, or the hardware failed. Denied connectivity? Electronics
HW-SW Interaction	When the expectation of the H/W and S/W are inconsistent, e.g. the wrong switch closes or the data appears on the wrong channel or in the wrong directory.
Recovery	When a recovery action or fault protection uncovers a problem.
Special Procedures	Testing a specific mission maneuver.

Trigger	When to Select this Trigger
SW Configuration	If the defect is triggered by: <ul style="list-style-type: none"> • Missing or incorrect files • Denied access to files • Wrong S/W versions • Routine build of the S/W version not proceeding as expected. • Incorrect delivery
Start and Restart	The error was detected while the system/subsystem was being: <ul style="list-style-type: none"> • initialized, reboot, shutdown, • powered up, or • restarted following an earlier shutdown or complete system or subsystem failure
Unknown	The trigger can not be determined
Workload & Stress	The error was detected after the system was placed under a heavy task load or run for a very long period of time. Types of problems under test include possible memory leaks, buffer overflows, file and queue full conditions, and delays such as late arrival of responses, crossed arrival of responses, loss of responses, and late execution of timed commands.

TARGET

Target: “Represents the high-level identity of the entity that was fixed” [ODC web page]. The target can be the identity that was changed to avoid the problem in the future (e.g., software that is updated to avoid a future problem with hardware) or the identity that was used to fix a problem (e.g., a contingency file or command that is sent to solve a hardware problem). Usually found in “Corrective Action” description.

Target	When to Select this Target
Ground software	<p>If the corrective action was in ground software. Ground software includes:</p> <ul style="list-style-type: none"> • Mission and Science Planning, • Operations, Operations Analysis Software, • Data Management Software, • Sequence Development Software, • Ground Data Transport Software, • Simulator Software, • Pre-launch Integration and Test Software, • Modeling Software, • Bench Test Equipment Software, • Commercial Software supporting development [Ref: OP/SP software classification]
Flight software	<ol style="list-style-type: none"> 1. Selected if the corrective action was in software residing on the spacecraft, either in the flight control computer or in the instrument computer, etc. Includes commercial software (e.g., operating systems) 2. Selected if the corrective action was: <ul style="list-style-type: none"> • A real-time command, • Flight code update/patch.
Information Development	If the corrective action was to a procedure, documentation, or technical information.
Build/Package	If the corrective action involved proper installation of files in the expected directory
None/ Unknown	<p>Nothing was targeted to be fixed and the problem was closed.</p> <p>Or</p> <p>Insufficient information to determine what was done to correct the problem or the problem was closed due to insufficient data/information to be able to determine what needs to be fixed.</p>
Hardware	The hardware is targeted to fix the problem.

TYPE

Type: Represents the actual correction that was made. "is the meaning of the fix" (R. Chillarege). Collectively, the defect types describe the nature of work.

Target	Type	When to Select this Type
Ground Software	Function/Algorithm	<ol style="list-style-type: none"> 1. When the defect is the result of the omission or incorrect implementation of: <ul style="list-style-type: none"> • significant capability, or • requirements, or • end-user interfaces, or • global data structure(s). 2. When the defect is the result of an efficiency or correctness problem that affects the task and was fixed by (re)implementing an algorithm or a local data structure 3. When the defect is the result of omission or incorrect validation of parameters or data in conditional statements.
	Interfaces	<p>When the defect is the result of a communication problem between: Modules, components, device drivers, function. Note: The defect that is the result of passing the wrong type of variable is an I/F problem, while the defect that is the result of passing the wrong value, is an assignment.</p>
	Assignment/Initialization	<p>When the defect is the result of:</p> <ul style="list-style-type: none"> • value(s) assigned incorrectly, or • not assigned at all, or • wrong calibration value. <p>Note that a fix involving multiple assignments corrections may be type Algorithm.</p>
	Timing	<ol style="list-style-type: none"> 1. When the defect is the result of timing error between: <ul style="list-style-type: none"> • system/subsystem, • modules, • S/W-H/W, etc. or 2. When the defect is the result of the omission or an incorrect use of serialization for controlling access to a shared resource.
	Flight Rule	<ol style="list-style-type: none"> 1. When the defect is the result of an incorrect or missing ground Flight Rule.
	Testbed Environment	<p>When the testbed does not support the executed command or capability.</p>

Target	Type	When to Select this Type
Flight Software	Function/Algorithm	<p>2. When the defect is the result of the omission or incorrect implementation of:</p> <ul style="list-style-type: none"> • significant capability, or • requirements, or • end-user interfaces, or • global data structure(s) <p>3. When the defect is the result of an efficiency or correctness problem that affects the task and was fixed by (re)implementing an algorithm or a local data structure, or</p> <p>4. When the defect is the result of omission or incorrect validation of parameters or data in conditional statements .</p>
	Interfaces	<p>When the defect is the result of a communication problem between: Modules, components, device drivers, function.</p> <p>Note: The defect that is the result of passing the wrong type of variable is an I/F problem, while the defect that is the result of passing the wrong value, is an assignment.</p>
	Assignment/Initialization	<p>When the defect is the result of a value(s):</p> <ul style="list-style-type: none"> • assigned incorrectly or • not assigned at all, or • a wrong calibration value. <p>Note than a fix involving multiple assignments corrections may be type Algorithm.</p>
	Timing	<p>1. When the defect is the result of timing error between:</p> <ul style="list-style-type: none"> • system/subsystem, • modules, • S/W-H/W, etc. <p>2. When the defect is the result of the omission or an incorrect use of serialization for controlling access to a shared resource.</p>
	Flight Rule	<p>When the defect is the result of an incorrect or missing spacecraft Flight Rule.</p>
Build/Package	Install dependency	<p>When the defect is encounter during installation, if needed files were missing or misplaced. Also if at execution, files were missing due to an installation problem.</p>
	Packaging/Scripts	<p>When the defect is encountered during the system build process and was the result of:</p> <ul style="list-style-type: none"> • the library system, or • faulty change management, or • version control.

Target	Type	When to Select this Type
Information Development	Documentation	When the problem is the result of an error in the: <ul style="list-style-type: none"> • written description contained in user guides, • installation manuals, or • prologues, or • code comments. Note , this should not be confused with an error or omission in the requirements or design that might be a Function or Interface defect type.
	Missing Procedures	When the problem is the result of a missing, out of date, or incomplete procedure.
	Procedure not followed	When the problem is the result of using the wrong procedure or not following the documented procedure.
None/Unknown	Nothing Fixed	The problem could not be fixed or decided did not need to be fixed; the problem was closed without indication that anything was fixed.
	Unknown	Insufficient information provided.
Hardware	Hardware	The hardware was fixed to resolve the problem (installation, connectivity problems?)

Excel Database of ODC Classification of post-launch critical software Incident/Surprise/Anomaly reports

Activity	Trigger	Target	Type
Flight Operations	Special Procedure	Ground Software	Assignment/Initialization
Flight Operations	Special Procedure	Flight Software	Assignment/Initialization
System Test	Inspection/Review	Information Development	Procedures
Flight Operations	Hardware Failure	Information Development	Procedures
System Test	Cmd Seq Test	Flight Software	Function/Algorithm
Flight Operations	Hardware Failure	Flight Software	Function/Algorithm
Flight Operations	Special Procedure	Information Development	Procedures
Flight Operations	Special Procedure	None/Unknown	Nothing Fixed
Flight Operations	Normal Activity	Ground Resources	Resource Conflict
Flight Operations	Data Access/Delivery	Ground Software	Function/Algorithm
Flight Operations	Normal Activity	Ground Software	Interfaces
System Test	Cmd Seq Test	Ground Software	Assignment/Initialization
System Test	Cmd Seq Test	Ground Software	Function/Algorithm
System Test	Cmd Seq Test	Ground Software	Function/Algorithm
Flight Operations	Normal Activity	Information Development	Procedures
System Test	Cmd Seq Test	Ground Software	Function/Algorithm
System Test	Software Configuration	Information Development	Procedures
System Test	Cmd Seq Test	Flight Software	Assignment/Initialization
System Test	Cmd Seq Test	Ground Software	Function/Algorithm
Flight Operations	Data Access/Delivery	Information Development	Procedures
Unknown	Unknown	None/Unknown	Nothing Fixed
Flight Operations	Normal Activity	Flight Software	Assignment/Initialization

Activity	Trigger	Target	Type
System Test	Software Configuration	Information Development	Procedures
Flight Operations	Data Access/Delivery	Information Development	Procedures
System Test	Software Configuration	Ground Software	Interfaces
System Test	Hardware Configuration	None/Unknown	Nothing Fixed
Flight Operations	Special Procedure	Information Development	Procedures
System Test	Inspection/Review	Information Development	Procedures
Flight Operations	Recovery	Flight Software	Flight Rule
Flight Operations	Recovery	Flight Software	Function/Algorithm
System Test	Inspection/Review	Ground Software	Function/Algorithm
Flight Operations	Recovery	Information Development	Documentation
Flight Operations	Normal Activity	Flight Software	Assignment/Initialization
Unknown	Unknown	None/Unknown	Nothing Fixed
Flight Operations	Recovery	Flight Software	Function/Algorithm
Flight Operations	Recovery	Flight Software	Function/Algorithm
Flight Operations	Hardware Failure	None/Unknown	Nothing Fixed
Flight Operations	Special Procedure	Information Development	Procedures
Flight Operations	Normal Activity	Information Development	Procedures
Flight Operations	Special Procedure	Flight Software	Function/Algorithm
Flight Operations	Normal Activity	None/Unknown	Unknown
Flight Operations	Special Procedure	Information Development	Procedures
Flight Operations	Special Procedure	Flight Software	Timing
Flight Operations	Data Access/Delivery	Information Development	Procedures
Flight Operations	Hardware Failure	Hardware	Hardware
System Test	Hardware Configuration	Hardware	Hardware

Activity	Trigger	Target	Type
Flight Operations	Data Access/Delivery	Ground Software	Function/Algorithm
Flight Operations	Data Access/Delivery	Ground Software	Function/Algorithm
System Test	Software Configuration	Flight Software	Assignment/Initialization
System Test	Inspection/Review	Flight Software	Assignment/Initialization
System Test	Hardware Configuration	Information Development	Procedures
Unknown	Unknown	Information Development	Procedures
Flight Operations	Data Access/Delivery	Information Development	Procedures
System Test	Hardware Configuration	Information Development	Documentation
Unknown	Unknown	Information Development	Procedures
System Test	Start/Restart/Shutdown	Ground Software	Assignment/Initialization
Flight Operations	Data Access/Delivery	Information Development	Procedures
System Test	Software Configuration	Build Package	Installation dependency
Flight Operations	Special Procedure	Information Development	Procedures
System Test	Inspection/Review	Ground Software	Function/Algorithm
System Test	Software Configuration	Ground Software	Assignment/Initialization
System Test	Software Configuration	None/Unknown	Nothing Fixed
System Test	Hardware Configuration	Hardware	Hardware
System Test	Start/Restart/Shutdown	Ground Software	Assignment/Initialization
System Test	Software Configuration	Information Development	Procedures
System Test	Software Configuration	Ground Software	Function/Algorithm
Flight Operations	Data Access/Delivery	Flight Software	Function/Algorithm
Flight Operations	Data Access/Delivery	None/Unknown	Nothing Fixed
System Test	Hardware Configuration	Hardware	Hardware
Flight Operations	Data Access/Delivery	Information Development	Procedures

Activity	Trigger	Target	Type
Flight Operations	Data Access/Delivery	Ground Software	Timing
Flight Operations	Data Access/Delivery	Ground Software	Timing
Flight Operations	Data Access/Delivery	Information Development	Procedures
Flight Operations	Recovery	Flight Software	Timing
Flight Operations	Normal Activity	Flight Software	Timing
System Test	Inspection/Review	Flight Software	Timing
Flight Operations	Data Access/Delivery	None/Unknown	Nothing Fixed
Flight Operations	Data Access/Delivery	None/Unknown	Nothing Fixed
Flight Operations	Normal Activity	None/Unknown	Nothing Fixed
Flight Operations	Hardware Failure	None/Unknown	Nothing Fixed
Flight Operations	Data Access/Delivery	None/Unknown	Nothing Fixed
Flight Operations	Special Procedure	Flight Software	Timing
System Test	Hardware Configuration	Flight Software	Assignment/Initialization
Flight Operations	Normal Activity	Information Development	Procedures
Flight Operations	Data Access/Delivery	Information Development	Procedures
Flight Operations	Data Access/Delivery	Ground Software	Interfaces
Flight Operations	Data Access/Delivery	None/Unknown	Nothing Fixed
Flight Operations	Data Access/Delivery	Ground Software	Assignment/Initialization
Flight Operations	Data Access/Delivery	Information Development	Procedures
Flight Operations	Data Access/Delivery	None/Unknown	Nothing Fixed

Activity	Trigger	Target	Type
Flight Operations	Data Access/Delivery	Ground Software	Interfaces
System Test	Hardware Configuration	Flight Software	Interfaces
Flight Operations	Recovery	None/Unknown	Nothing Fixed
Flight Operations	Data Access/Delivery	Information Development	Procedures
Flight Operations	Data Access/Delivery	Ground Software	Interfaces
Flight Operations	Hardware Failure	Flight Software	Function/Algorithm
Flight Operations	Data Access/Delivery	Information Development	Documentation
Flight Operations	Data Access/Delivery	None/Unknown	Unknown
System Test	Software Configuration	Ground Software	Assignment/Initialization
Unknown	Unknown	None/Unknown	Nothing Fixed
System Test	Software Configuration	Build Package	Install dependency
System Test	Hardware Configuration	Ground Software	Function/Algorithm
System Test	Hardware Configuration	Ground Software	Function/Algorithm
Flight Operations	Special Procedure	Flight Software	Assignment/Initialization
Flight Operations	Hardware Failure	Flight Software	Assignment/Initialization
Flight Operations	Recovery	Flight Software	Function/Algorithm
Flight Operations	Recovery	Flight Software	Assignment/Initialization
Flight Operations	Normal Activity	Flight Software	Assignment/Initialization
System Test	Software Configuration	Ground Resources	Resource Conflict
System Test	Start/Restart/Shutdown	Ground Software	Function/Algorithm
System Test	Hardware Configuration	Ground Software	Assignment/Initialization

Activity	Trigger	Target	Type
Flight Operations	Normal Activity	Information Development	Procedures
Flight Operations	Normal Activity	Information Development	Procedures
System Test	Start/Restart/Shutdown	Ground Software	Assignment/Initialization
System Test	Software Configuration	Information Development	Procedures
System Test	Software Configuration	Information Development	Procedures
Flight Operations	Hardware Failure	None/Unknown	Nothing Fixed
Flight Operations	Special Procedure	Flight Software	Assignment/Initialization
Flight Operations	Hardware Failure	Information Development	Procedures
Flight Operations	Hardware Failure	Information Development	Procedures
System Test	Software Configuration	Information Development	Procedures
Flight Operations	Hardware Failure	Ground Resources	Resource Conflict
Flight Operations	Normal Activity	Ground Software	Timing
System Test	Software Configuration	None/Unknown	Unknown
System Test	Software Configuration	Build Package	Packaging Script
Flight Operations	Data Access/Delivery	Ground Software	Function/Algorithm
Flight Operations	Data Access/Delivery	Information Development	Procedures
Flight Operations	Data Access/Delivery	Ground Software	Interfaces
Flight Operations	Data Access/Delivery	Ground Software	Interfaces
Flight Operations	Data Access/Delivery	Information Development	Procedures

Activity	Trigger	Target	Type
Flight Operations	Data Access/Delivery	Information Development	Procedures
System Test	Software Configuration	Ground Software	Function/Algorithm
System Test	Cmd Seq Test	Ground Software	Assignment/Initialization
Flight Operations	Data Access/Delivery	Information Development	Procedures
System Test	Hardware Configuration	Hardware	Hardware
Flight Operations	Data Access/Delivery	Information Development	Procedures
Flight Operations	Data Access/Delivery	Information Development	Procedures
Flight Operations	Data Access/Delivery	Information Development	Procedures
Flight Operations	Data Access/Delivery	Information Development	Procedures
Flight Operations	Data Access/Delivery	Information Development	Procedures
Flight Operations	Normal Activity	Information Development	Procedures
System Test	Hardware Configuration	Information Development	Procedures
Flight Operations	Data Access/Delivery	Ground Software	Assignment/Initialization
Flight Operations	Normal Activity	Information Development	Procedures
Flight Operations	Special Procedure	Flight Software	Assignment/Initialization
Flight Operations	Hardware Failure	Flight Software	Assignment/Initialization
System Test	Software Configuration	Build Package	Install dependency
Flight Operations	Normal Activity	Information Development	Procedures
Flight Operations	Data Access/Delivery	None/Unknown	Nothing Fixed

Activity	Trigger	Target	Type
Flight Operations	Data Access/Delivery	Information Development	Procedures
Flight Operations	Special Procedure	None/Unknown	Nothing Fixed
System Test	Software Configuration	Ground Software	Interfaces
Flight Operations	Hardware Failure	Flight Software	Function/Algorithm
System Test	Inspection/Review	Flight Software	Assignment/Initialization
Flight Operations	Recovery	Flight Software	Function/Algorithm
Flight Operations	Hardware Failure	Flight Software	Assignment/Initialization
Flight Operations	Recovery	Flight Software	Timing
System Test	Hardware Configuration	None/Unknown	Nothing Fixed
Flight Operations	Recovery	Flight Software	Assignment/Initialization
Flight Operations	Recovery	Flight Software	Function/Algorithm
Flight Operations	Recovery	Flight Software	Function/Algorithm
System Test	Inspection/Review	None/Unknown	Nothing Fixed
Flight Operations	Hardware Failure	Hardware	Hardware
Unknown	Unknown	None/Unknown	Unknown
Flight Operations	Data Access/Delivery	None/Unknown	Unknown
Flight Operations	Data Access/Delivery	None/Unknown	Unknown
Flight Operations	Data Access/Delivery	None/Unknown	Nothing Fixed
Flight Operations	Data Access/Delivery	Flight Software	Function/Algorithm
Flight Operations	Data Access/Delivery	Flight Software	Function/Algorithm
Flight Operations	Data Access/Delivery	Flight Software	Function/Algorithm
Flight Operations	Special Procedure	None/Unknown	Nothing Fixed

Section 4. Excel Database of ODC Classification of testing software problem reports

Activity	Release	Trigger	Target	Type
I & T	R2	HW-SW Interaction	Flight Software	Interfaces
ATLO	R2	Start/Restart	Information Development	Missing Procedures
ATLO	R2	Command Execution		
ATLO	R2	Capability Invocation		
ATLO	R2	Start/Restart		
ATLO	R2	Capability Invocation	Flight Software	Assignment/Initialization
Build Test	R2	Command Execution	Flight Software	Interfaces
ATLO	R2	Command Execution	Information Development	Documentation
ATLO	R2	HW Configuration	Information Development	Documentation
Build Test	R3	Command Execution	Ground Software	Assignment/Initialization
Build Test	R3	Inspection/Review	Flight Software	Interfaces
Build Test	R3	Command Execution	Information Development	Procedures not followed
Build Test	R3	Command Execution	Flight Software	Assignment/Initialization
Build Test	R3	Command Execution		
ATLO	R2	SW Configuration	Ground Software	
ATLO	R2	HW Configuration		
ATLO	R2	Capability Invocation	Flight Software	Timing
Build Test	R3	Start/Restart	Flight Software	Interfaces
Build Test	R3	Command Execution		
ATLO	R2	HW-SW Interaction		
ATLO	R2	Command Execution		
ATLO	R2	Capability Invocation		
ATLO	R2	Capability Invocation		
I & T	R3	HW-SW Interaction	Information Development	Documentation
I & T	R3	Command Execution	Flight Software	Timing
I & T	R3	Capability Invocation		
I & T	R3	Capability Invocation	Ground Software	Assignment/Initialization
I & T	R3	Capability Invocation		
I & T	R3	Capability Invocation	Ground Software	Function/Algorithm
I & T	R3	Command Execution		
ATLO	R2	Capability Invocation	None/Unknown	Nothing Fixed
ATLO	R2	Command Execution	Flight Software	Assignment/Initialization
ATLO	R2	HW Configuration	Hardware	Hardware
ATLO	R2	Command Execution	Flight Software	Assignment/Initialization
ATLO	R2	HW Configuration	None/Unknown	Nothing Fixed
I & T	R3	Command Execution	None/Unknown	Nothing Fixed
I & T	R3	Capability Invocation	Flight Software	Interfaces
I & T	R3	Command Execution	Flight Software	Function/Algorithm
I & T	R3	Start/Restart	None/Unknown	Nothing Fixed
I & T	R3	Start/Restart	Ground Software	Assignment/Initialization
I & T	R3	Command Execution	Flight Software	
I & T	R3	Start/Restart	Information Development	Missing Procedures
I & T	R3	Command Execution	Flight Software	Function/Algorithm
I & T	R3	Command Execution	Ground Software	Assignment/Initialization
I & T	R3	SW Configuration	Ground Software	Interfaces
I & T	R3	SW Configuration	Flight Software	Interfaces
ATLO	R2	Command Execution	Flight Software	Function/Algorithm
ATLO	R2	HW Configuration	Hardware	Hardware
I & T	R3	Start/Restart	Ground Software	
I & T	R3	Inspection/Review	Ground Software	Assignment/Initialization
Other	R3	Inspection/Review	Flight Software	Timing
I & T	R3	Capability Invocation	Flight Software	Function/Algorithm

Activity	Release	Trigger	Target	Type
I & T	R3	Command Execution	Flight Software	Function/Algorithm
I & T	R3	Start/Restart	Flight Software	Assignment/Initialization
Other	R4	Capability Invocation	Flight Software	Assignment/Initialization
Other	R4	Capability Invocation	Flight Software	Assignment/Initialization
Other	R4	Start/Restart	Flight Software	Assignment/Initialization
I & T	R3	Capability Invocation	Flight Software	Function/Algorithm
I & T	R3	Command Execution	Flight Software	Function/Algorithm
I & T	R3	Start/Restart	None/Unknown	Nothing Fixed
ATLO	R2	HW Configuration	Information Development	Procedures not followed
ATLO	R2	HW Configuration	Hardware	Hardware
ATLO	R2	Capability Invocation	Ground Software	Assignment/Initialization
ATLO	R2	HW Configuration	Hardware	Hardware
ATLO	R2	HW-SW Interaction	None/Unknown	Nothing Fixed
I & T	R3	Command Execution	None/Unknown	Nothing Fixed
Build Test	R3	Unknown	Ground Software	Assignment/Initialization
Build Test	R3	Command Execution	Ground Software	Assignment/Initialization
Other	R3	Capability Invocation		
Other	R3	Start/Restart	Hardware	Hardware
Build Test	R3	Command Execution	Flight Software	Function/Algorithm
ATLO	R2	Capability Invocation	Flight Software	Function/Algorithm
ATLO	R2	Capability Invocation	Information Development	Documentation
I & T	R3	Capability Invocation	Information Development	Documentation
I & T	R3	Command Execution	Flight Software	Function/Algorithm
Other	R3	Command Execution	Ground Software	Assignment/Initialization
I & T	R3	Command Execution	Hardware	Hardware
I & T	R3	Capability Invocation	Hardware	Hardware
I & T	R3	HW-SW Interaction	Flight Software	Interfaces
I & T	R3	Command Execution	Flight Software	Assignment/Initialization
I & T	R3	HW-SW Interaction	Flight Software	Assignment/Initialization
I & T	R3	Command Execution	Flight Software	Interfaces
I & T	R3	Start/Restart	None/Unknown	Nothing Fixed
ATLO	R2	HW-SW Interaction	Ground Software	Assignment/Initialization
ATLO	R2	Capability Invocation	Information Development	Missing Procedures
I & T	R3	Start/Restart	Ground Software	Timing
I & T	R3	Command Execution	Information Development	Documentation
I & T	R3	HW-SW Interaction	None/Unknown	Nothing fixed
I & T	R3	Capability Invocation	Information Development	Missing Procedures
ATLO	R3	Capability Invocation	Ground Software	Timing
Build Test	R3	Command Execution	Flight Software	
ATLO	R3	HW-SW Interaction	Ground Software	Function/Algorithm
I & T	R3	Unknown		
I & T	R3	SW Configuration	None/Unknown	Nothing Fixed
Build Test	R3	Command Execution	Ground Software	Assignment/Initialization
Build Test	R3	Start/Restart		
I & T	R4	Capability Invocation	None/Unknown	Nothing Fixed
Build Test	R3	Command Execution	Ground Software	Assignment/Initialization
I & T	R4	Start/Restart	Ground Software	Function/Algorithm
I & T	R4	Capability Invocation	Information Development	Missing Procedures
Build Test	R3	Command Execution		
Build Test	R3	Capability Invocation	None/Unknown	Nothing Fixed
I & T	R4	Capability Invocation	Ground Software	Function/Algorithm
Build Test	R3	Command Execution		
Build Test	R4	Start/Restart	Flight Software	Function/Algorithm
ATLO	R3	HW-SW Interaction	Ground Software	Assignment/Initialization
I & T	R4	Capability Invocation	None/Unknown	Nothing Fixed
I & T	R4	Unknown	Information Development	Documentation
I & T	R4	Command Execution	None/Unknown	Nothing Fixed

Activity	Release	Trigger	Target	Type
I & T	R4	Command Execution	Information Development	Documentation
I & T	R4	Command Execution	Flight Software	Function/Algorithm
Build Test	R4	Command Execution	Flight Software	Interfaces
I & T	R4	Capability Invocation	Information Development	Procedures not followed
I & T	R4	Capability Invocation	Information Development	Procedures not followed
I & T	R4	Capability Invocation	Information Development	Missing Procedures
I & T	R4	Capability Invocation	Information Development	Missing Procedures
I & T	R4	Command Execution	None/Unknown	Nothing Fixed
I & T	R4	Capability Invocation	Information Development	Procedures not followed
I & T	R4	Start/Restart	Ground Software	Timing
I & T	R4	Command Execution	Information Development	Missing Procedures
I & T	R4	Capability Invocation	Information Development	Missing Procedures
Build Test	R3	Workload & stress	Ground Software	Timing
Build Test	R3	Workload & stress		
Build Test	R3	Capability Invocation		
I & T	R4	HW-SW Interaction	Hardware	Hardware
I & T	R4	Capability Invocation	Information Development	Missing Procedures
Build Test	R3	HW Configuration	Hardware	Hardware
Build Test	R4	Command Execution	Ground Software	Interfaces
Build Test	R4	Capability Invocation	Ground Software	Function/Algorithm
I & T	R4	Command Execution	Ground Software	Assignment/Initialization
I & T	R4	HW-SW Interaction	Ground Software	Assignment/Initialization
I & T	R4	Command Execution	Hardware	Hardware
ATLO	R3	Unknown	Ground Software	Assignment/Initialization
I & T	R4	Recovery	Flight Software	Function/Algorithm
I & T	R4	Command Execution	Ground Software	Function/Algorithm
ATLO	R3	Command Execution	None/Unknown	Nothing fixed
I & T	R4	Command Execution	Information Development	Procedures not followed
I & T	R4	Capability Invocation	None/Unknown	Nothing Fixed
Build Test	R5	Inspection/Review	Ground Software	Function/Algorithm
I & T	R4	Command Execution	Ground Software	Function/Algorithm
I & T	R4	SW Configuration	Ground Software	Testbed Environment
Build Test	R4	Command Execution	Ground Software	Assignment/Initialization
ATLO	R4	Start/Restart	Ground Software	Assignment/Initialization
Build Test	R4	Capability Invocation	Ground Software	Function/Algorithm
Build Test	R4	Command Execution	Flight Software	Interfaces
I & T	R4	Start/Restart	None/Unknown	Nothing Fixed
Build Test	R4	HW-SW Interaction	Flight Software	Function/Algorithm
I & T	R4	Command Execution	Flight Software	Assignment/Initialization
I & T	R4	Special Procedure	Ground Software	Function/Algorithm
I & T	R4	Special Procedure	Ground Software	Function/Algorithm
ATLO	R4	HW Configuration	Hardware	Hardware
I & T	R4	Command Execution	Flight Software	Assignment/Initialization
ATLO	R4	HW-SW Interaction	Flight Software	Assignment/Initialization
I & T	R4	Capability Invocation	Flight Software	Assignment/Initialization
I & T	R4	Command Execution	Flight Software	Function/Algorithm
I & T	R4	Capability Invocation	Flight Software	Function/Algorithm
ATLO	R4	HW-SW Interaction	Ground Software	Function/Algorithm
I & T	R4	Capability Invocation	Flight Software	Timing
I & T	R4	Capability Invocation	Flight Software	Assignment/Initialization
Build Test	R4	Capability Invocation	None/Unknown	Nothing Fixed
I & T	R4	Capability Invocation	Flight Software	Timing
I & T	R4	Command Execution	None/Unknown	Nothing Fixed
I & T	R4	HW Configuration	None/Unknown	Nothing Fixed
I & T	R4	Recovery	Flight Software	Assignment/Initialization

Activity	Release	Trigger	Target	Type
I & T	R4	Recovery	Flight Software	Assignment/Initialization
ATLO	R4	Start/Restart	Information Development	Missing Procedures
I & T	R4	Start/Restart	Ground Software	Assignment/Initialization
I & T	R4	Command Execution	Information Development	Documentation
I & T	R4	Command Execution	None/Unknown	Nothing Fixed
ATLO	R4	Recovery	None/Unknown	Nothing Fixed
I & T	R4	Command Execution	Flight Software	Timing
ATLO	R4	Capability Invocation	Flight Software	Function/Algorithm
I & T	R4	Command Execution	None/Unknown	Nothing Fixed
I & T	R4	Start/Restart	None/Unknown	Nothing Fixed
I & T	R4	Capability Invocation	None/Unknown	Nothing Fixed
I & T	R4	Capability Invocation	Ground Software	Testbed Environment
I & T	R4	HW-SW Interaction	Information Development	Documentation
I & T	R4	Command Execution	None/Unknown	Nothing Fixed
ATLO	R4	HW-SW Interaction	Flight Software	Function/Algorithm
I & T	R4	Command Execution	Flight Software	Function/Algorithm
I & T	R4	Command Execution	None/Unknown	Nothing Fixed
I & T	R4	Command Execution	Flight Software	Function/Algorithm
I & T	R4	Start/Restart	None/Unknown	Nothing Fixed
ATLO	R4	Command Execution	Flight Software	Assignment/Initialization
I & T	R4	Command Execution	Flight Software	Function/Algorithm
I & T	R4	Command Execution	Flight Software	Function/Algorithm
I & T	R4	Recovery	None/Unknown	Nothing Fixed
I & T	R4	HW-SW Interaction	Flight Software	Function/Algorithm
I & T	R4	Capability Invocation	Information Development	Documentation
I & T	R4	Command Execution	Information Development	Procedures not followed
I & T	R4	Capability Invocation	Information Development	Documentation
I & T	R4	Capability Invocation	None/Unknown	Nothing Fixed
I & T	R4	HW-SW Interaction	Flight Software	Interfaces
I & T	R4	Capability Invocation	Information Development	Documentation
I & T	R4	HW-SW Interaction	Flight Software	Interfaces
I & T	R4	Command Execution	Flight Software	Assignment/Initialization
I & T	R4	Command Execution	None/Unknown	Nothing Fixed
I & T	R4	Command Execution	Ground Software	Timing
ATLO	R4	Command Execution	Flight Software	Function/Algorithm
ATLO	R4	Command Execution	Flight Software	Function/Algorithm
I & T	R4	Command Execution	Information Development	Documentation
I & T	R4	Command Execution	Flight Software	Function/Algorithm
I & T	R4	Inspection/Review	Information Development	Documentation
I & T	R4	Start/Restart	Information Development	Missing Procedures
I & T	R4	Command Execution	Flight Software	Assignment/Initialization
I & T	R4	Capability Invocation	Flight Software	Assignment/Initialization
I & T	R4	Capability Invocation	Flight Software	Timing
ATLO	R4	Start/Restart	Flight Software	Function/Algorithm
ATLO	R4	Command Execution	Flight Software	Function/Algorithm
I & T	R4	Start/Restart		
I & T	R4	Capability Invocation		
I & T	R4.1	Recovery	Flight Software	Assignment/Initialization
I & T	R4.1	Start/Restart	Flight Software	Assignment/Initialization
ATLO	R4	Command Execution	Flight Software	Assignment/Initialization
I & T	R4.1	Capability Invocation		
I & T	R4.1	Capability Invocation	Flight Software	Assignment/Initialization
I & T	R4.1	Command Execution	Ground Software	Function/Algorithm
ATLO	R4	Capability Invocation	Information Development	Nothing Fixed
ATLO	R4	Command Execution	Flight Software	Timing
I & T	R4.1	Command Execution	Flight Software	Function/Algorithm
I & T	R4.1	Command Execution	Information Development	Missing Procedures

Activity	Release	Trigger	Target	Type
I & T	R4.1	Capability Invocation	Flight Software	Assignment/Initialization
I & T	R4.1	Start/Restart	Flight Software	
I & T	R4.1	Recovery	Flight Software	Assignment/Initialization
Other	R4	Command Execution		
I & T	R4.1	Command Execution		
I & T	R4.1	Inspection/Review	Flight Software	Assignment/Initialization
I & T	R4.1	Workload & stress	Flight Software	Timing
ATLO	R4	Inspection/Review	None/Unknown	Unknown
I & T	R4.1	Capability Invocation		
I & T	R4	Command Execution		
I & T	R4.1	Command Execution	Flight Software	Function/Algorithm
I & T	R4.1	SW Configuration	Flight Software	Assignment/Initialization
ATLO	R4	Command Execution	Information Development	Missing Procedures
I & T	R4	Command Execution	None/Unknown	Nothing Fixed
I & T	R4	Start/Restart	Flight Software	Timing
I & T	R4	Capability Invocation	Flight Software	Timing
I & T	R4	Capability Invocation	Flight Software	Function/Algorithm
I & T	R4	Command Execution	Flight Software	Function/Algorithm
I & T	R4	Inspection/Review	Information Development	Documentation
I & T	R4	Start/Restart	None/Unknown	Nothing Fixed
I & T	R4	Command Execution	None/Unknown	Nothing Fixed
I & T	R4	Command Execution	None/Unknown	Nothing Fixed
I & T	R4	Command Execution	Flight Software	Function/Algorithm
I & T	R4	Recovery	Flight Software	Function/Algorithm
I & T	R4	Capability Invocation	Flight Software	Assignment/Initialization
I & T	R4	Command Execution	Ground Software	Interfaces
ATLO	R4	Command Execution	Flight Software	Interfaces
I & T	R4	Command Execution	Flight Software	Function/Algorithm
I & T	R4.1	Start/Restart	Flight Software	Function/Algorithm
ATLO	R4	Capability Invocation	Information Development	Documentation
ATLO	R4.1	Command Execution	Flight Software	Assignment/Initialization
I & T	R4	Recovery	Flight Software	Interfaces
I & T	R4.1	Unknown	Ground Software	Interfaces
I & T	R4.1	SW Configuration	Flight Software	Function/Algorithm
I & T	R4.1	Capability Invocation	Ground Software	Assignment/Initialization
I & T	R4.1	Recovery	Flight Software	Function/Algorithm
I & T	R4.1	Command Execution	Flight Software	Interfaces
ATLO	R4	Start/Restart	Flight Software	Interfaces
I & T	R4.1	Inspection/Review	Flight Software	Function/Algorithm
I & T	R4.1	Start/Restart	Information Development	Missing Procedures
I & T	R4.1	Capability Invocation	Hardware	Hardware
I & T	R4.1	Capability Invocation	Flight Software	Function/Algorithm
I & T	R4.1	Start/Restart	None/Unknown	Nothing Fixed
ATLO	R4	Capability Invocation	Flight Software	Function/Algorithm
ATLO	R4	SW Configuration		
I & T	R5	Capability Invocation	Flight Software	Interfaces
I & T	R5	Start/Restart	Flight Software	Assignment/Initialization
I & T	R5	Special Procedure	Flight Software	Assignment/Initialization
ATLO	R4	SW Configuration	Flight Software	Assignment/Initialization
I & T	R5	Capability Invocation	Flight Software	Function/Algorithm
I & T	R5	Command Execution	None/Unknown	Nothing Fixed
I & T	R5	Command Execution	Flight Software	Assignment/Initialization
I & T	R5	Command Execution	Flight Software	Function/Algorithm
I & T	R5	Command Execution	Flight Software	Function/Algorithm
I & T	R5	Capability Invocation	Flight Software	Interfaces
I & T	R5	Capability Invocation	Flight Software	Function/Algorithm
I & T	R5	Capability Invocation	Flight Software	Interfaces
I & T	R5	Command Execution	Flight Software	Function/Algorithm

Activity	Release	Trigger	Target	Type
I & T	R5	Inspection/Review	Ground Software	Interfaces
ATLO	R4	Special Procedure	Flight Software	
I & T	R5	Special Procedure	Flight Software	Function/Algorithm
I & T	R5	Recovery	Flight Software	Function/Algorithm
ATLO	R4	Capability Invocation	Ground Software	
I & T	R5	Start/Restart	Flight Software	Interfaces
ATLO	R4	Command Execution		
I & T	R5	Capability Invocation	Flight Software	Function/Algorithm
I & T	R5	Command Execution	None/Unknown	Nothing Fixed
I & T	R5	Command Execution	Flight Software	
ATLO	R4.1	Capability Invocation	Flight Software	Function/Algorithm
ATLO	R4	Special Procedure	Flight Software	
I & T	R5	Command Execution	Flight Software	Function/Algorithm
ATLO	R4	Capability Invocation	Flight Software	
ATLO	R4.1	Special Procedure	Flight Software	
I & T	R5	Command Execution	Flight Software	
I & T	R5	Command Execution		
I & T	R5	Command Execution		
I & T	R5	Command Execution	None/Unknown	Nothing Fixed
I & T	R5	Command Execution	Flight Software	Function/Algorithm
I & T	R5	Command Execution	Flight Software	Assignment/Initialization
ATLO	R4.1	Capability Invocation	Flight Software	
I & T	R5	Recovery	None/Unknown	Nothing Fixed
I & T	R5	Start/Restart	None/Unknown	Nothing Fixed
I & T	R6	Capability Invocation	None/Unknown	Nothing Fixed
I & T	R6	Special Procedure	None/Unknown	Nothing Fixed
I & T	R6	Capability Invocation	Flight Software	Function/Algorithm
I & T	R6	HW-SW Interaction	Flight Software	Interfaces
ATLO	R5	Capability Invocation	Information Development	Missing Procedures
I & T	R6	Recovery	Flight Software	Function/Algorithm
I & T	R4.1	Capability Invocation	Flight Software	
I & T	R4.1	Capability Invocation	None/Unknown	Nothing Fixed
I & T	R4.1	HW Configuration	None/Unknown	Nothing Fixed
I & T	R4.1	Command Execution		

Section 5. Pivot Table Summary of Results from Analysis of Post-Launch Anomalies

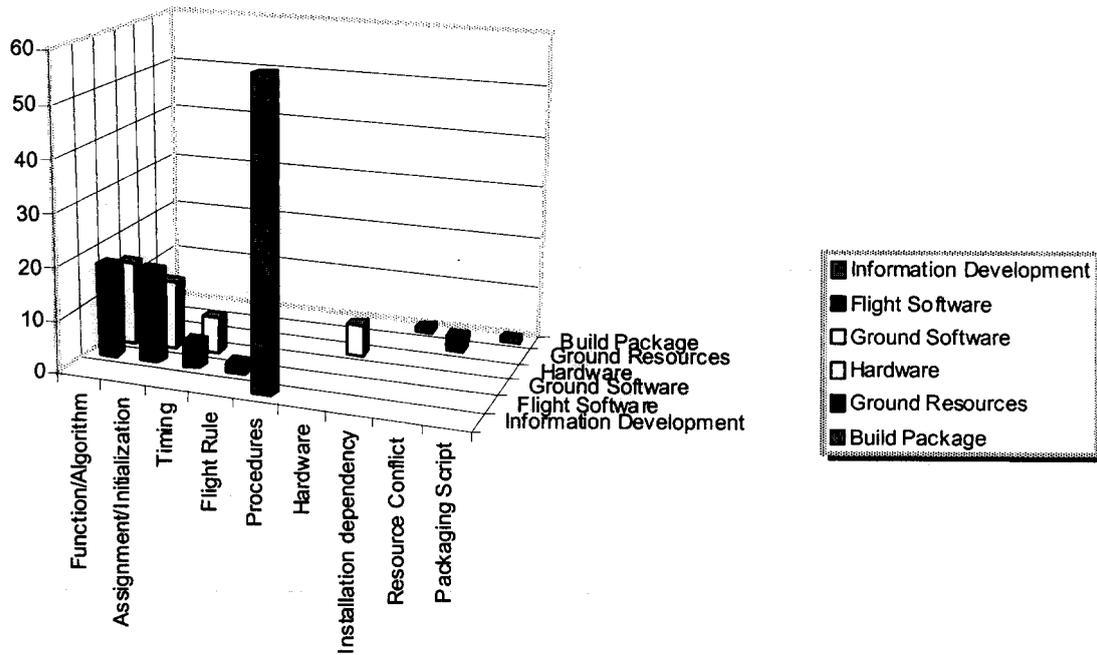


Figure 5-1 Distribution of Type within Target

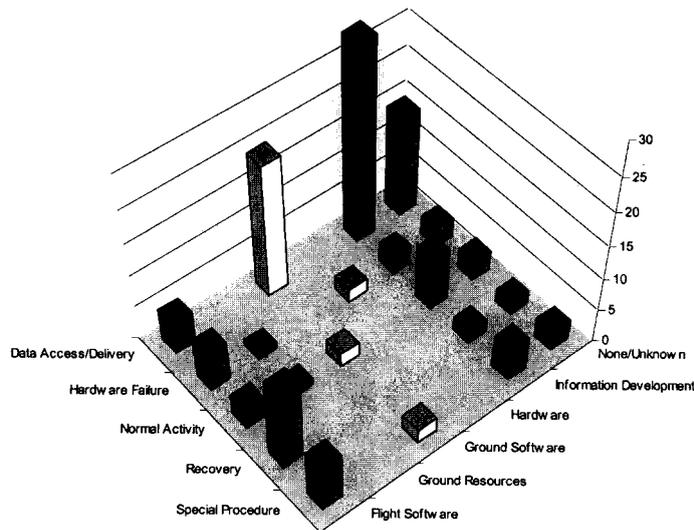


Figure 5-2 Distribution of Target within Trigger

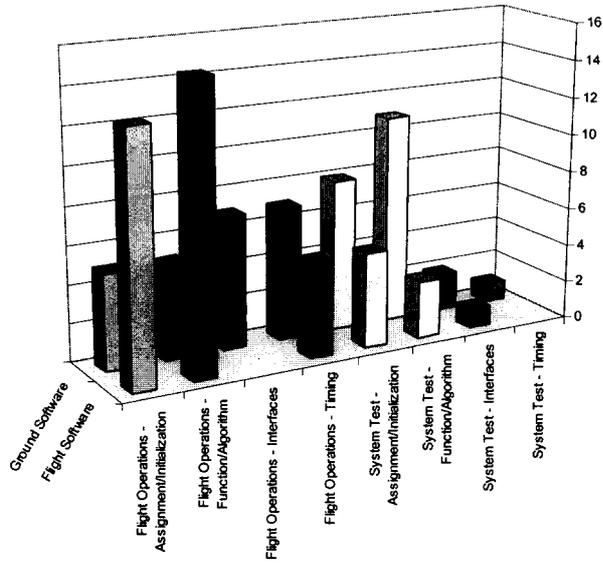
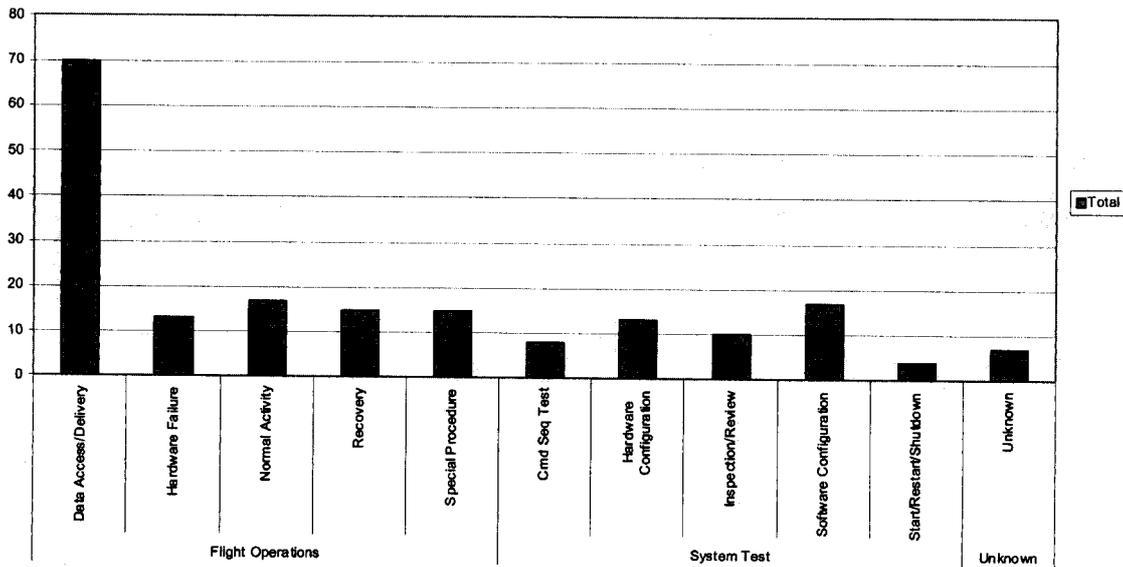


Figure 5-3 S/W Flight & Ground Targets vs. Type by Activity

Distribution of Triggers within Activity



Section 6. Pivot Table Summary of Results from Analysis of Some Testing Problem Reports

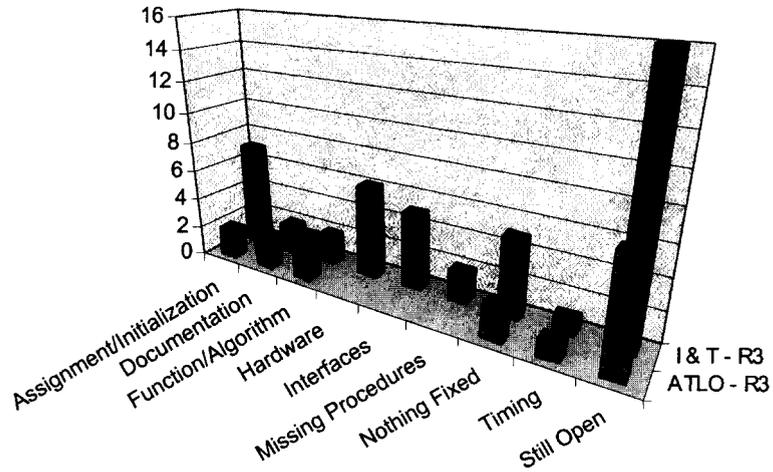


Figure 6-1 Distribution of Type by Activity for Release 3

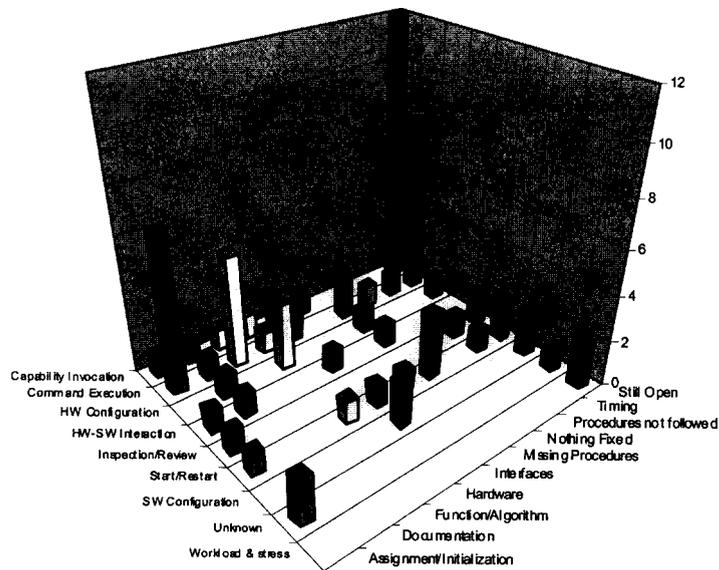


Figure 6-2 Type Distribution within Trigger

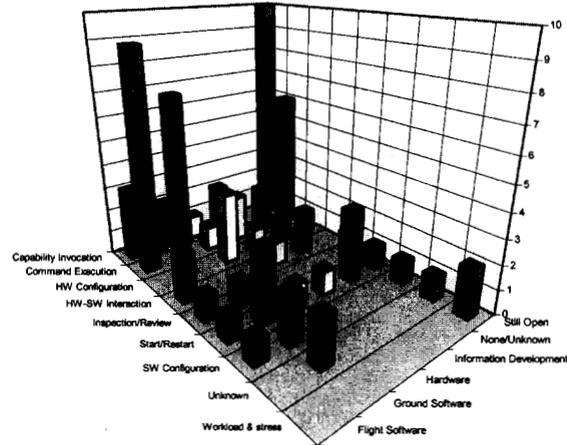


Figure 6-3 Target Distribution within Trigger

Section 7. Paper Describing Results from Preliminary Analysis of Some Testing Problem Reports (accepted to IEEE International Conference on Software Engineering, 2003)

Requirements Discovery During the Testing of Safety-Critical Software

Robyn R. Lutz
*Jet Propulsion Laboratory
 and Iowa State University*
rlutz@cs.iastate.edu

Inés Carmen Mikulski
*Jet Propulsion Laboratory
 Pasadena, CA 91109-8099*
ines.c.mikulski@jpl.nasa.gov

Abstract

This paper describes the role of requirements discovery during the testing of a safety-critical software system. Analysis of problem reports generated by the integration and system testing of an embedded, safety-critical software system identified four common mechanisms for requirements discovery and resolution during testing: (1) Incomplete requirements, resolved by changes to the software, (2) Unexpected requirements interactions, resolved by changes to the operational procedures, (3) Requirements confusion by the testers, resolved by changes to the documentation, and (4) Requirements confusion by the testers, resolved by a determination that no change was needed. The experience reported here confirms that requirements discovery during testing is frequently due to communication difficulties and subtle interface issues. The results also suggest that "false positive" problem reports from testing (in which the software behaves correctly but unexpectedly) provide a rich source of requirements information that can be used to reduce operational anomalies in critical systems.

1. Introduction

This paper describes the role of requirements discovery during the testing of a safety-critical software system. Difficulties with requirements have been repeatedly implicated as a source of both testing defects [2, 7] and accidents in deployed systems [3, 10]. In an effort to improve our understanding of how requirements discovery occurs during testing, and how such discoveries are resolved (or are not resolved) prior to deployment, we investigated the requirements-related problems reported during testing of a safety-critical system currently under development. Analysis of the problem reports generated during integration and system testing of the software distinguished four common mechanisms for requirements discovery and resolution:

The research described in this paper 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 Code Q Software Program Center Initiative, UPN 323-08. The first author's research is supported in part by National Science Foundation Grants CCR-0204139 and CCR-0205588.

(1) *Incomplete requirements, resolved by changes to the software.* As often occurs, testing caused several previously unidentified requirements to surface. These new requirements usually involved complicated interface issues between software components or between hardware and software. Several of the incomplete requirements involved fault protection, of special concern in safety-critical systems.

(2) *Unexpected requirements interactions, resolved by changes to the operational procedures.* A closely related mechanism for requirements discovery was the identification during testing of unexpected interactions among the existing requirements. Typically, these interactions resulted in new required sequencing of activities when the interleaved processes unexpectedly caused incorrect behavior or did not achieve the required precondition for correct execution of the software.

(3) *Requirements confusion by the testers, resolved by changes to the documentation.* Testing revealed some significant misunderstandings on the part of the testers regarding what the requirements actually were. In these cases the software worked as required and the requirements were correct, but the software's behavior was unexpected. The corrective action was not to fix

the software, but to enhance the documentation in order to better communicate the required software behavior or requirements rationale.

(4) *Requirements confusion by the testers, resolved by a determination that no change was needed.* In this mechanism testing also revealed a gap in requirements understanding. However, the problem report was judged to be a “false positive,” i.e., indicating failure where the software in fact behaved correctly. We found that in some cases where the software behaved correctly but unexpectedly, an opportunity was missed to prevent similar, subsequent requirements confusion by the operators of the deployed system. We propose some guidelines for distinguishing and responding to such situations.

The experience reported here suggests that problem reports generated during testing are an underused source of information about potential requirement-related anomalies that may occur after the software is deployed. Test defect reports provide a unique source of insights into future users' gaps in domain knowledge, misidentification of requirement rationales, and erroneous assumptions regarding required sequences of activities. In this limited sense, testing problem reports may provide a preview of some possible operational problems. The main contributions of the paper are (1) to identify the common mechanisms by which requirements discovery and resolution occurred during testing, and (2) to report the lessons learned regarding how such discoveries can be better used to reduce future requirements anomalies in the deployed system.

The rest of the paper is divided into sections as follows. Section 2 describes the approach used to investigate requirements discovery during testing. Section 3 discusses and evaluates the results in the context of some illustrative examples. Section 4 briefly compares the experience described here to others' findings. Section 5 summarizes the lessons learned.

2. Approach

The data for this analysis consisted of the 171 completed problem reports (PRs) written by project test teams during integration and system testing of the Mars Exploration Rovers (MER). MER, to be launched in 2003, will explore Mars with two robotic rovers equipped to search for evidence of previous water. The size of MER's flight software is roughly 300K Lines of Code, implementing approximately 400 software requirements of varying degrees of granularity. Although the software was delivered in a series of builds, we do not distinguish here among the builds due to the relatively small number of PRs.

The on-line problem reports (PRs) filled out by the project consist of three parts. The first part describes

the problem and is filled out by the tester when the problem occurs. The second part is filled out by the analyst assigned to investigate the problem. The third part is filled in later with a description of the corrective action that was taken to close out the problem.

The approach we selected for the analysis of the PRs was an adaptation of Orthogonal Defect Classification (ODC) [1]. ODC provides a way to “extract signatures from defects” and to correlate the defects to attributes of the development process (Fig. 1). Our ODC-based approach uses four attributes to characterize each PR: Activity, Trigger, Target, and Type. The Activity describes where the defect surfaced, e.g., Integration Test or System Test. The Trigger describes the environment or condition that had to exist

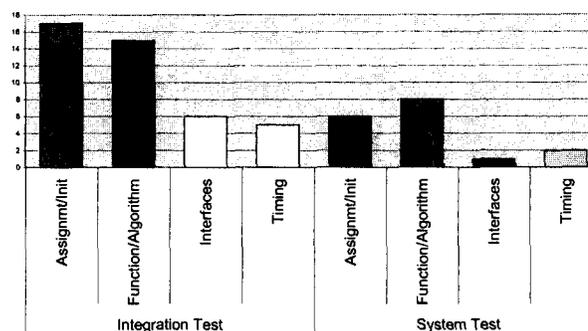


Figure 1 Types of Corrections for Testing Reports

for the defect to appear. In the testing environment, the trigger was usually the testing of a single command or of a capability sequence (i.e., a software requirement scenario). The Target describes the high-level entity that was fixed in response to the problem report, e.g., Flight software, Ground software, etc. The Type describes the actual correction that was made, i.e., “the meaning of the fix” [1].

The two authors classified the PRs using the adapted ODC. Both of us have experience on flight projects at JPL but neither are directly involved with the testing of the MER software. MER engineers generously assisted us with answers to our process and domain questions.

Following the ODC approach, we defined each classification attribute and the possible values it could take in a document that was reviewed by MER project personnel. Adaptation of the standard ODC categories to the spacecraft domain was driven by the need to capture core properties of the anomalies seen during testing. In order to improve repeatability and reduce bias, the process of classification involved three steps in which (1) each analyst separately classified the set of anomalies, (2) inconsistent classifications were highlighted and each analyst had an opportunity to correct any clear errors in her own classifications (e.g., missing fields), and (3) they analysts jointly reviewed

the remaining inconsistencies and resolved them through discussion. A detailed description of the classification process and of efforts to remove bias is provided in [9].

The work reported here is part of a multi-year pilot study to reduce the number of safety-critical software anomalies that occur post-launch. This paper reports the first experience using the adapted ODC technique on a spacecraft currently under development. The motivation was to mine the testing problem reports for insights into how requirements discovery during testing can be used to forestall or mitigate some critical software anomalies during operations.

3. Results and analysis

We here describe each of the four mechanisms for requirements discovery and resolution identified during analysis of the problem reports (PRs) generated in integration and system testing of the spacecraft software. A subsection describes each mechanism in terms of the ODC classification values that characterize it, provides a more in-depth causal analysis of some typical examples, and evaluates the adequacy of the corrective action taken to resolve the requirements discovery.

3.1 Incomplete requirements, resolved by changes to the software

Sixty-five of the completed 171 integration and system testing PRs were resolved by a change to the flight software (Fig. 2). In ODC terms, the Target for these sixty-five PRs was "Flight Software." Twenty-three of the Flight Software PRs had an ODC Type of "Assignment/Initialization." These PRs were resolved by changes to parameters in the light of new system knowledge. They entailed discovery of new requirements knowledge, but not of new functional requirements. Two typical examples of these PRs are, in one case, a change to the value of the variable "max" to avoid unintended triggering of fault protection and, in another case, a change to require that a component come up disabled rather than enabled after a reboot.

Another twenty-three of the sixty-five Flight Software testing PRs had an ODC Type of "Function/Algorithm." Some of these changes involved design or implementation issues such as testing of functions not yet delivered in the current build. However, the PRs of interest to us from a requirements perspective are the ten that entailed more substantial changes to the flight software as the result of knowledge gained during testing.

Each of these ten PRs was resolved by requiring a new software function. Many of the corrective actions

taken to close these PRs involved additional reasonableness checks on preconditions and post-conditions. Several involved startup/restart scenarios, or the correct triggering of recovery software. New requirements included an additional health check, a parameter validation check, an inhibit to checks of disabled software, distinguishing unavailability from non-response of a unit, turning off encoding in some cases, ignoring false out-of-order messages, providing a new capability to copy a rate to a register, an additional check so a warning does not occur in a shutdown mode, and a new capability to command a hardware unit.

An additional seven of the Flight Software PRs had an ODC Type of "Timing," and seven more had an ODC Type of "Interfaces." In these types, as well, the role of testing in the discovery of new requirements was

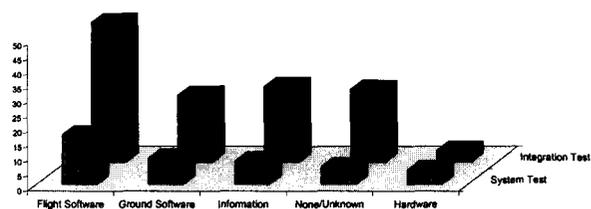


Figure 1 Fixing Testing Problem Reports

evident. Due to space constraints, we only mention briefly that several resulted in new requirements to insert delays in the software to compensate for interface delays. It is worth noting that no PRs documented extra requirements (where the flight software did more than it should).

3.2 Unexpected requirements interactions, resolved by changes to the operational procedures

The previous subsection described new requirements that were discovered during testing and resolved by changes to the flight software. In this subsection we describe unexpected requirement interactions that were discovered during testing and fixed, not by changes to the software, but instead by changes to the procedures that will constrain operational activities.

This mechanism for requirements discovery tended to involve emerging requirements, not discovered until testing, on the sequencing or timing of activities in interfacing software components or software/hardware interfaces. The ODC Target for these thirteen PRs was "Information Development" and the ODC Type for these PRs was "Missing or Incomplete Procedures." This second mechanism is a special case of the incomplete requirements described above, involving

new knowledge and requirements that must be enforced on interactions. However, this mechanism differs from the first mechanism described above in that achievement of the new requirement is here allocated to procedures rather than to software.

Most of these PRs dealt only with testing procedures and were not relevant to operations or maintenance. However, three of them involved discovery during testing of unexpected requirements interactions. In each of these three cases, responsibility for the requirement was allocated to operations. For example, in one PR testing revealed that unless a spacecraft component was re-calibrated before use, it triggered fault-protection software. The discovery of this requirement for sequential activities (first calibrate, then use) was allocated to an operational procedure.

In another, a tester observed that, contrary to expectations, an off command was issued redundantly by a software fault monitor. Analysis showed that this behavior was correct, but idiosyncratic. The corrective action was to avoid these redundant commands during operations by carefully selecting the high and low limits to preclude the state observed in testing. It is easy to see how, even with a documented procedure in place, this situation might recur in operations.

This third mechanism for requirements discovery is of interest in preventing operational anomalies because corrections made to procedures still depend on the correct implementation of the procedure by the operator of the deployed system each time the relevant scenario arises. We were thus interested in whether some of the new requirements for constraining interactions, levied on the operational procedures, might be better handled in software. Given the small number of s in the study, no conclusion was appropriate. However, the examples suggest that in long-lived systems, the tradeoff between easy but operator-dependent procedural fixes and more costly but operator-independent software fixes should be considered.

3.3 Requirements confusion by the testers, resolved by changes to the documentation

The previous two subsections both described requirements discovery mechanisms in which the testers' expectations were consistent with the required software behavior. Testing revealed missing requirements that had to be added in order to achieve the correct, and expected, behavior. The requirements discovery mechanism described in this section is different in that the testers' expectations regarding the required software behavior were incorrect. The resolution was to try to remove the source of the testers' confusion by improving the documentation of the existing requirements and their rationale.

Fourteen of the 171 testing PRs were resolved by

changes to the documentation. The ODC Target for these PRs was "Information Development" and the ODC Type was "Documentation." (Only PRs that changed just documentation but not software or procedures are labeled this way).

Four of the PRs of type "Documentation" revealed erroneous requirements assumptions by the testers. For example, in one case, the tester incorrectly assumed that certain heaters remain on during the transition from one mode to another, as the spacecraft transitions from the pre-separation mode of the Mars lander to the entry/descent mode (as the lander enters the Martian atmosphere). The tester's assumption was reasonable but incorrect. In fact, there is a software requirement on another component to turn the heaters off when this transition occurs. Documentation of this fact was added to the Functional Design Document and the procedure writers were notified of the update in order to correct the misunderstanding prior to launch.

In these PRs it was requirements confusion, rather than new requirements that were discovered during testing. The perceived inconsistency between the test results and the required behavior was inaccurate. The corrective action was not to fix the software but the source of confusion. This resulted in improved communication of the rationale for the existing behavior in the existing project documentation

3.4 Requirements confusion by the testers, resolved by a determination that no change was needed.

The final mechanism for requirements discovery is similar to the previous one except that no fix is made, even to documentation. Thirty of the 171 testing PRs have an ODC Target of "None/Unknown" and an ODC Type of "Nothing Fixed." The reason that nothing was fixed is that these PRs were "false positives," raising an alarm when nothing was broken. Our interest in investigating this mechanism was to see if any of these PRs described requirements confusion or requirements interactions that could potentially recur in flight operations. If so, it might be that some change to documentation or procedure was indicated.

As expected, for most of the PRs there was, in fact, nothing to fix. For example, thirteen of the thirty PRs referred to problems that were no longer relevant (e.g., the current build removed the issue); two were clearly one-time operator errors (e.g., misreading the test results); and three were relevant only to the test environment but not to flight. However, eight of the thirty raised possible flight concerns, although in each case the software worked as required. We describe several of these more fully here, since they support our claim that false positives encountered during testing often provide a useful window into latent requirements

misunderstandings during operations.

For example, in one case the PR stated as an error that commands issued when a remote unit was off were not rejected as expected, but instead were completed when the unit rebooted. Although the software operated correctly, the PR revealed a gap in understanding of the rationale for the software's required behavior (a gap, by the way, that was shared by the analysts). Since this requirements confusion could apparently reappear in a post-launch operational scenario, it may merit additional documentation to preclude a similar mistake by an operator.

Another PR of Type "Nothing Fixed" describes a situation in which one component, attempting to communicate with another component, received warning messages indicating that an invalid response had occurred. In fact, the communication attempt happened to occur during a few-millisecond timeout that takes place in some particular scenarios. This behavior is, in fact, correct and required, and subsequent communication attempts will be normal. However, the effect of the timeout is rather subtle.

In a third example, the tester incorrectly assumed that a telemetry (data download) channel output the value of a counter when the channel instead provided the value of the counter's high-water mark (the highest value yet recorded for the counter). Thus, even when the counter was reset, the telemetry value remained constant. The requirements rationale is sound -- that the fault-protection software needs information regarding the worst case over a time interval, not just the current snapshot of a frequently reset counter. However, the requirements misunderstanding by the tester is reasonable and suggests that a similar erroneous assumption might be possible later.

Testing PRs often provide detailed descriptions of sequences of input, states, error messages, and even partial dumps in order that the test scenario can later be duplicated. This level of detail is extraordinarily useful in allowing an analyst to pinpoint not only whether an error has occurred but also the source of any confusion regarding the required behavior. Incorrect assumptions (e.g., about the effect of specific commands on the state of the system) and gaps in domain knowledge (e.g., of hardware idiosyncrasies or transients) can often be identified from the details in the problem reports.

3.5 Implications for testing

Given limited project resources (in terms of schedule and budget), should these "false-positive" testing reports be documented further? Based on the problem reports seen here and on past experience with operational anomalies [8, 9], we suggest the following guideline: *if the situation described in the problem report could recur in operations, and if the*

requirements confusion or misunderstanding of required interactions could also recur in operations, then the problem report may merit additional attention. Using this guideline, each of the three examples above would have involved additional corrective actions.

For example, one such false-positive PR recorded a perceived discrepancy between two time tags that should be identical. In fact, the software worked as required. The two time tags were two different representations of the same time (cumulative number of seconds since a standard base time and the translation of that value to the current UTC, the Universal Time). This misunderstanding by the tester is one that could be repeated by an operator or maintenance programmer with conceivably hazardous effect, so may merit additional documentation.

Experience with the MER testing PRs also suggests that PRs related to certain critical activities always merit additional attention even if the PR merely records requirements confusion. Thus, if the testing PR involves fault protection software, critical control software, critical maneuvers or activities (e.g., engine burns), or critical mission phases (e.g., insertion of the spacecraft into a planetary orbit), then the problem report should take into account measures to prevent the required behavior that surprised the testers from later surprising the operators.

3.6 Implications for operations

False-positive problem reports from testing (when the software behavior was correct but unexpected, so nothing was fixed) have significant value in a development organization if the requirements confusion or emerging domain knowledge that led to them can be identified and remedied. Especially in a long-lived spacecraft system where turnover of operational personnel is to be expected, loss of knowledge regarding requirement rationale can be substantial. It appears that testers' requirements confusion may provide some small degree of "crystal ball" insight into possible future post-release misunderstandings and, thus, the opportunity to mitigate those gaps, whether by documentation, training, or changes to software or procedures. Techniques to trace the requirements misunderstandings encountered during testing into operations are at this time an open problem.

Some results from a recent study by the authors confirm that the requirements discovery mechanisms found in testing can affect safety-critical operations. This ODC-based study profiled 199 safety-critical software anomalies recorded post-launch on seven spacecraft [9]. One of the surprises to emerge from that study was that some procedures needed for post-launch operations were not in place, and that these omissions contributed to 21% of the safety-critical anomalies.

Another finding related to requirements discovery was that in most of the anomalies of Type “Nothing Fixed” (14% of the total), what was originally reported as a safety-critical anomaly was in fact the required behavior of the spacecraft, i.e., requirements confusion. Better understanding of the various requirements-discovery mechanisms in testing has as its primary goal to prevent slippage of requirements-related testing problems into operations.

4. Related Work

Most work on the analysis of testing defects has focused on measuring the quality or readiness of the software for release (see, e.g., [2]). In our study, the focus was instead on how to use the requirements discoveries made during testing (either of incomplete software or of incorrect human assumptions) to reduce critical defects during operations.

The results reported here tend to confirm the central role that Hanks, Knight, and Strunk have found for problems communicating domain knowledge [3]. Weiss, Leveson, Lundqvist, Farid, and Stringfellow specifically implicate requirements misunderstanding in several recent disasters, stating, “software-related accidents almost always are due to misunderstanding about what the software should do” [10]. In this regard, the instances of requirements confusion found here are somewhat similar to the examples of mode confusion by pilots and other operators that Leveson and others have described.

Previous work by one of the authors found that safety-related testing defects on two earlier spacecraft arose most commonly from (1) misunderstanding of the software’s interfaces with the rest of the system and (2) discrepancies between the documented requirements and the requirements needed for correct functioning of the system [7]. A recent study by Lauesen and Vinter found similar results for non-critical systems, with slightly more than half the defect reports being requirements defects and the major source being missing requirements [5].

Several defect classification methods (see, e.g., [6, 1]) include communication failures as root causes or as defect triggers. However, these approaches tend not to distinguish requirements confusion in which the reported software behavior is actually correct from other kinds of communication failures, as we found helpful here. These studies also focus on ways to prevent requirements defects from reaching testing, whereas we were more interested in how to use testing problem reports to prevent defects from reaching operations.

Harold recently suggested the use of “test artifacts” for software engineering tasks in describing future

directions for work, but added that “this research is in its infancy” [4]. The experience described here suggests that testing problem reports may be useful test artifacts that can be more effectively mined for requirements insights to reduce post-deployment anomalies.

5. Conclusion

The results reported here distinguish four common mechanisms for requirements discovery and resolution during the integration and system testing of a safety-critical software system. One of the lessons learned was that requirements discovery during testing is frequently due to communication difficulties and subtle interface issues. Requirements discovery in testing thus drove changes not only to the software but also to the operational procedures and to the documentation of requirements rationale. Another lesson learned was that false-positive problem reports from testing (where the software behaves correctly but unexpectedly) provide a rich source of insights into potential requirements-related anomalies during operations. This information may be able to be used to reduce operational anomalies in critical systems.

Acknowledgments. The authors thank Daniel Erickson and the Mars Exploration Rover engineers and test teams for their assistance and feedback.

References

- [1] R. Chillarege, I. S. Bhandari, J. K. Chaar, M. J. Halliday, D. S. Moebus, B. K. Ray, and M.-Y. Wong, “Orthogonal Defect Classification—A Concept for In-Process Measurements, *IEEE Trans on SW Eng*, Nov. 1992, pp. 943-956.
- [2] S. Gardiner, ed. *Testing Safety-Critical Software*, Springer-Verlag, London, 1999.
- [3] K. S. Hanks, J. C. Knight, and E. A. Strunk, “Erroneous Requirements: A Linguistic Basis for Their Occurrence and an Approach to Their Reduction,” *Proc. 26th NASA Goddard SW Eng Workshop*, IEEE, Greenbelt, MD, Nov., 2001.
- [4] M. J. Harold, “Testing: A Roadmap” in *The Future of Software Engineering*, A. Finkelstein, ed., ACM Press, New York, 2000.
- [5] S. Lauesen and O. Vinter, “Preventing Requirements Defects: An Experiment in Process Improvement,” *Requirements Engineering Journal*, 2001, pp. 37-50.
- [6] M. Leszak, D. E. Perry, and D. Stoll, “A Case Study in Root Cause Defect Analysis,” *Proc 22nd Intl Conf SW Eng (ICSE’00)*, IEEE CS Press, Los Alamitos, CA, 2002, pp. 428-437.

[7] R. Lutz, "Analyzing Software Requirements Errors in Safety-Critical, Embedded Systems," *Proc IEEE Intl Symp Req Eng*, IEEE CS Press, 1993, pp. 126-133.

[8] R. Lutz and I. C. Mikulski, "Operational Anomalies as a Cause of Safety-Critical Requirements Evolution," *The Journal of Systems and Software*, to appear.

[9] R. Lutz and I. C. Mikulski, "Empirical Analysis of Safety-Critical Anomalies During Operations," submitted to *IEEE Trans on SW Eng*.

[10] K. A. Weiss, N. Leveson, K. Lundqvist, N. Farid, and M. Stringfellow, "An Analysis of Causation in Aerospace Accidents," *Space*, 2001, Aug., 2001.

Section 8. Process Recommendations Resulting from ODC Analysis of Safety-Critical Post-Launch Software Anomalies from Seven Spacecraft

Finding	Original Hypothesis	True/False	Process Recommendation
1. Some needed procedures were not in place during operations	Most critical anomalies caused by procedures not being followed (58 of 199)	False (for 29%; 17 of 58)	Reuse Lessons Learned regarding missing procedures needed in previous missions: (1) Assemble generic “Checklist of Common Procedures” (2) Inspect new missions against this checklist prior to launch (3) Increase operational mission testing pre-launch
2. Operations sometimes not informed of expected behavior of spacecraft so no fix required	For critical anomalies where no fix ever occurred, this was result of loss of the spacecraft (25 of 199)	False (for 36%; 9 of 25)	Improve communication with Mission Operations: (1) Formalize ISA update procedure (2) Remove high-criticality rating from ISAs that turn out to be non-problems (3) Update documentation used by operations as software or procedures change
3. Technical difficulties with downlink cause critical anomalies	Ground-software downlink difficulties cause many critical anomalies (20 of 199)	True (for 100%; 20 of 20)	Downlink is technically challenging: (1) Monitor Ground Systems anomaly-reporting metrics during operations to track trends and apply suggested improvements to software development (2) Shorten time-to-close for open ground system anomaly reports (ISAs)
4. Most changes to flight software aren’t just fixes to code, but involve requirements and design	Incorrect code causes many flight-software anomalies (44 of 199)	False (for 34%; 15 of 44)	Better requirements engineering for maintenance is needed: (1) Use anomaly-reporting database for early identification of potential new software requirements to compensate for hardware or environmental problems (2) Take advantage of hardware trend analysis to maximize

Finding	Original Hypothesis	True/False	Process Recommendation
			lead time for planning software changes
5. Software requirements changes often involve (1) new requirements to handle rare but high-consequence scenarios; (2) new requirements to compensate for hardware limitation or failure	Incorrect flight-software requirements cause many anomalies	False (for 29%; 13 of 44)	Attention to fault scenarios during requirements phase pays off: (1) Use early Contingency Planning to support requirements evolution (2) Pursue requirements completeness with regard to failure scenarios via software FMECA and software FTA
6. Most critical software anomalies involve ground software or operations	Most critical software ISAs involve flight software	False (for 22%; 44 of 199)	Ground software causes many critical anomalies: (1) Identify historically high-risk ground software via problem reports (2) Increase software assurance of ISA-associated ground software
7. "Rare" events trigger many but not most (~1/3) anomalies	Most critical software ISAs rest from atypical situations (recovery, hardware failure or special procedures)	False (for 34%; 68 of 199)	"Rare events" do occur fairly often and do cause anomalies: (1) Continue to test software for correct recovery from hardware and software failure scenarios
8. Most anomalies don't occur in critical mission phases	Most anomalies occur during critical mission phases	False (for 18 %; 36 of 199)	Don't let guard down during cruise: (1) Update analyses as hardware, environment change (2) System test special procedures (e.g., calibration) prior to use (3) Since much testing occurs during cruise, explicitly incorporate cruise-phase testing into project test plans

Summary of Unexpected Patterns

<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.