Empirical Analysis of Safety-Critical Anomalies During Operations

Robyn R. Lutz, *Member*, *IEEE*, and Inés Carmen Mikulski, *Member*, *IEEE Computer Society*

Abstract—Analysis of anomalies that occur during operations is an important means of improving the quality of current and future software. Although the benefits of anomaly analysis of operational software are widely recognized, there has been relatively little research on anomaly analysis of safety-critical systems. In particular, patterns of software anomaly data for operational, safety-critical systems are not well understood. This paper presents the results of a pilot study using Orthogonal Defect Classification (ODC) to analyze nearly two hundred such anomalies on seven spacecraft systems. These data show several unexpected classification patterns such as the causal role of difficulties accessing or delivering data, of hardware degradation, and of rare events. The anomalies often revealed latent software requirements that were essential for robust, correct operation of the system. The anomalies also caused changes to documentation and to operational procedures to prevent the same anomalous situations from recurring. Feedback from operational anomaly reports helped measure the accuracy of assumptions about operational profiles, identified unexpected dependencies among embedded software and their systems and environment, and indicated needed improvements to the software, the development process, and the operational procedures. The results indicate that, for long-lived, critical systems, analysis of the most severe anomalies can be a useful mechanism both for maintaining safer, deployed systems and for building safer, similar systems in the future.

Index Terms—Software and system safety, diagnostics, maintenance process, product metrics.

1 Introduction

The effort to build safe systems benefits from knowledge of the past. A key part of this knowledge is an understanding of past safety-critical anomalies. As Leveson has noted, "feedback of operational experience is one of the most important sources of information in designing, maintaining, and improving safety" [21]. The data needed to pursue this understanding is often recorded, occasionally analyzed, and too rarely used [4].

The work described here investigates one piece of this puzzle, namely, safety-critical software anomalies that occurred during operations. We focus on those situations that have in the past posed serious threats to the embedded software systems in an effort to gain insight into how to build safer systems in the future. Safety-critical operational anomalies are a source of information both for trend analysis of the current system and for improved safety of other similar systems. In this paper, we focus on the latter role. Feedback from safety-critical, operational anomaly reports can measure the accuracy of our assumptions about operational profiles, identify unexpected dependencies among embedded software and their systems and environment, and indicate needed improvements to the software, the development process, and the operational procedures.

The rest of the paper is organized as follows: Section 2 describes the approach and gives some background information regarding the anomaly data. Section 3 presents related work in the area. Section 4 describes the analysis process. Section 5 presents the results. Section 6 evaluates the results and considers their implications for safety-critical systems. Section 7 provides concluding remarks.

2 Approach

The data for the analysis of safety-critical operational anomalies were drawn from an institutional database of anomaly reports for multiple missions at Jet Propulsion Laboratory. The reporting mechanism for the anomalies is an online form called an "Incident/Surprise/Anomaly" (ISA) report. An ISA consists of three parts. The first part describes the problem as experienced by the operator. The second part presents the results of the subsequent analysis of the occurrence. The third part describes the final corrective action taken to close out the incident. Until all three parts are completed and signed, the ISA is said to be "open." Additional information regarding criticality, priority, time and date, subsystem, etc., may be entered into available fields.

It is worth noting that an ISA is not a defect report. An ISA is written whenever the behavior of the system differs from the expected behavior in the eyes of the operator. The ISA thus provides valuable information regarding gaps between the requirements as specified and implemented and the, perhaps different, users' expectations. The ISA also

- *R.R. Lutz is with the Department of Computer Science, Iowa State University, 226 Atanasoff Hall, Ames, IA 50011-1041 and the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109-8099. E-mail: rlutz@cs.iastate.edu.*
- *I.C. Mikulski is with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109-8099. E-mail: ines.c.mikulski@jpl.nasa.gov.*

Manuscript received 12 Mar. 2002; revised 26 Sept. 2003; accepted 19 Jan. 2004.
Recommended for acceptance by J. Knight.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 116071.

TABLE 1
Data Sources

Spacecraft	*Mission*	Launch Date
Galileo (GLL)	Jupiter	1989
Mars Global Surveyor (MGS)	Mars mapping	1996
Cassini/Huygens (Cass)	Saturn/Titan	1997
Deep Space 1 (DS1)	Ion propulsion, remote agent, etc.	1998
Mars Climate Orbiter (MCO)	Mars orbiter	1998
Mars Polar Lander (MPL)	Mars lander	1999
Stardust (STAR)	Comet sample return	1999

provides a means of documenting near-misses, i.e., high-consequence failures that almost occurred but were prevented by some fortuitous circumstance (e.g., fault monitoring, contingency commands, a change of mode, etc.). In some cases, analysis of the near-miss prompted a change to the flight software requirements to preclude such an anomaly in the future [23].

The data set analyzed consisted of the 199 critical ISAs from seven spacecraft that occurred between the launch date of each spacecraft and 21 August 2001. Table 1 lists each spacecraft, its primary mission, and its launch data. The systems were selected for this study to represent a cross-section of deep-space missions and were all launched between 1989 and 1999. The selection was made in accordance with recommendations from the database administrators regarding their confidence in the quality of the raw data (i.e., which projects populated the database most consistently).

The safety-critical anomalies per spacecraft numbered 8, 8, 15, 19, 29, 59, and 61, for a total of 199. The only data points that were removed from the original data set were four duplicates.

The criticality level for each ISA was assigned by the project based on standard classifications [17]. Since there were slight differences in the processes of the seven projects regarding which fields of the anomaly reports were used, we studied all anomaly reports that met one of the following four, project-assigned criteria in order to assure coverage of all critical ISAs:

1. high mission risk with significant or catastrophic risk and uncertain fix,
2. highest level criticality, i.e., unacceptable risk with no workaround,
3. catastrophic failure effect, and
4. highest priority ("must-fix") with significant or catastrophic failure effect.

The term "critical ISAs" refers to anomalies meeting one or more of these criteria.

The scope of the investigation reported here was to characterize postlaunch, safety-critical, software anomalies. The reason for focusing on this small set of anomalies was that, in order to improve the safety of future missions, we first examine those anomalies classified as safety-critical in previous missions. It is expected that additional insights into operational anomalies could be obtained from studying noncritical anomalies during operations, as well as from a comparative analysis of critical and noncritical anomalies. However, those studies are beyond the scope of the effort reported here.

The approach selected for the study of the anomalies was Orthogonal Defect Classification (ODC), described below. ODC provides a way to "extract signatures from defects" [5] and to correlate the defects to attributes of the development process. The problem definition was to analyze safety-critical, postlaunch spacecraft anomaly data in order to determine the effects of using the Orthogonal Defect Classification (ODC) method on the understanding of flight software anomalies, from the point of view of the software developers [26]. The pilot-study framework described by Glass [14] was used to structure the investigation. This framework consists of 35 steps grouped into the five activities of planning, designing, conducting, evaluating, and using the results.

3 Related Work

Our investigation built on an extensive body of work in defect analysis. The defect analysis technique that we used as the basis for our investigation of anomalies is Orthogonal Defect Classification (ODC), developed at IBM by Chillarege et al. [5] in the 1980's. ODC has been applied for defect analysis throughout the lifecycle. For example, Chillarege and Bassin have used ODC to classify software defects found during field operations [6], Dalal et al. have used ODC in operational systems but with the purpose of guiding prerelease process improvement [10], and Chillarege and Prasad have used ODC's trigger in a retrospective defect analysis of a Web application's development aimed at process improvement [8].

Statistical defect modeling, on the other hand, predicts the reliability of a software product, typically by measuring the short-term defect detection rate and estimating the number of defects remaining or the failure rate of the software [16], [12]. Bishop and Bloomfield, for example, used estimates of residual defects to calculate worst-case probability of survival [3]. Whereas their interest was in predicting reliability on a project, ours was on characterizing critical anomalies across projects.

Causal analysis, another widely used defect-analysis technique, has as its goal to identify the root cause of the defect and initiate an action so that the source of the defect is eliminated. In an earlier such study by one of the authors on spacecraft software, it was found that most of the critical anomalies during the testing phase involved requirements

or interfaces [22]. Similarly, when Lauesen and Vinter looked at 200 of the 800 defect reports available a few months after a product's release, they found that missing requirements were the most frequent cause [19]. However, when Leszak et al. performed a large, retrospective causal-analysis study of defects through the load-building phase (i.e., not during operations), they found that implementation defects consumed 75 percent of all effort. The defects were primarily of type algorithm or functionality [20]. Ostrand and Weyuker recently compared pre and post-release faults to investigate module fault density and fault proneness in an inventory-tracking system [28]. Unlike the results reported here, they found very few high-severity faults postrelease and did not investigate fault causes.

The work described here is unlike these previous studies in that it combines:

1. investigation of anomalies (including near-misses and operator confusion) rather than just defects,
2. focus on critical anomalies,
3. consideration of operational (postdeployment) anomalies, and
4. analysis of anomalies across a set of similar systems.

4 Analysis

Using a nominal-scale classification scheme, each defect was placed in a particular classification category based on the value of an attribute [13]. Each classification attribute and the possible values it could take were defined in a document. Of the eight original ODC attributes, four were selected as relevant to the data: Activity, Trigger, Target, and Type. Very roughly speaking, the Activity during which the anomaly occurred is the "When" (when the anomaly surfaced), the Trigger is the "What" (the environment or condition that had to exist for the defect to appear), the Target is the "Where" (the entity being fixed), and the Type is the "How" (the nature of the fix). The "orthogonal" nature of the attributes comes from their nonredundancy and the multiple perspectives they provide. A fifth ODC attribute, Impact, is implicit in our selection of the data (high-criticality anomalies). The other three attributes (defect qualifier, source, and age) were not available in the anomaly reports or were judged to be not of much relevance (e.g., constant values) in these systems. The ODC values within classification attributes required some adaptation to the postlaunch spacecraft domain. For example, almost all postlaunch ISAs involve what in ODC terms are system-testing defects, so the Activity classification had to be made more fine-grained. Fig. 1 presents the classification attributes and their possible values.

A preexperiment was performed on a small set of ISAs, in accordance with Kitchenham's suggestion [18], to gauge the experimental power of the approach. Results from the preexperiment included: 1) more precisely defining the attribute values so as to improve repeatability among analysts, 2) supplementing the definitions with examples from real anomaly data to improve training materials, and 3) adding a comment field to provide an in-process way to record issues as they arose. The comment field allowed capture of process observations and explanatory insights similar to Seaman's use of field memos to find patterns [29].

We used a three-step process in which 1) two analysts separately classified the same set of anomalies and entered the results in a spreadsheet, 2) after highlighting inconsistent classifications, each analyst performed a prereview check, looking at the other analyst's classifications and correcting any clear errors in one's own classifications, and 3) the analysts jointly reviewed the remaining discrepancies and resolved them through discussion.

Step 1 encouraged accuracy in that the analysts caught each other's misunderstandings (e.g., one of us had greater expertise in ground software and operations, the other one in flight software and fault protection) and gave us some measurements of repeatability. Step 2 encouraged rapid correction of simple errors, e.g., omitted fields due to interruptions or inattention. Step 3 supported process improvement by identifying unclear definitions, mismatches between the available classification values and the anomalies' descriptions, and multiple-valued attributes (e.g., two triggers, discussed in Section 5.2). Each of the first two steps took one to three minutes per ISA for each analyst, and the third step (the review) took about five minutes jointly for the small set of remaining ISAs for which the classification values still differed.

5 Results

In this section, we describe the results of classifying the safety-critical, postlaunch software anomalies according to the ODC-based technique described above.

After Pareto analysis of the results identified patterns of interest in the data (unexpected distributions of classification values or associations among attribute values), causal analysis explored possible explanations for why the patterns occurred. The use of causal analysis to investigate subsets of defects belonging to patterns of interest is typical of both ODC [5], [10] and of defect analysis [20], [25]. Often the diagnosis is performed by experts gathered to investigate the reasons for the patterns and how such patterns of defects can be prevented or controlled in the future.

Figs. 1 and 2 profile the distribution for Activity, Trigger, Target, and Type. Computation of the Chi-square statistic with $\alpha = 0.05$ for the significance level for each of Activity, Trigger, Target, and Type, leads us to reject for each the null hypothesis that the distribution is uniform among possible values. The following description first combines the results for all seven projects and then considers the normalized data for each project.

5.1 Activity

The most frequent Activity was "Flight Operations" (68 percent) (see Fig. 1). This distribution was expected for a postlaunch anomaly profile. On the other hand, it might seem surprising that 29 percent of the anomalies instead involved system testing. This is, in fact, characteristic of robotic space missions, which typically require many updates to the software during their missions. The lifetime

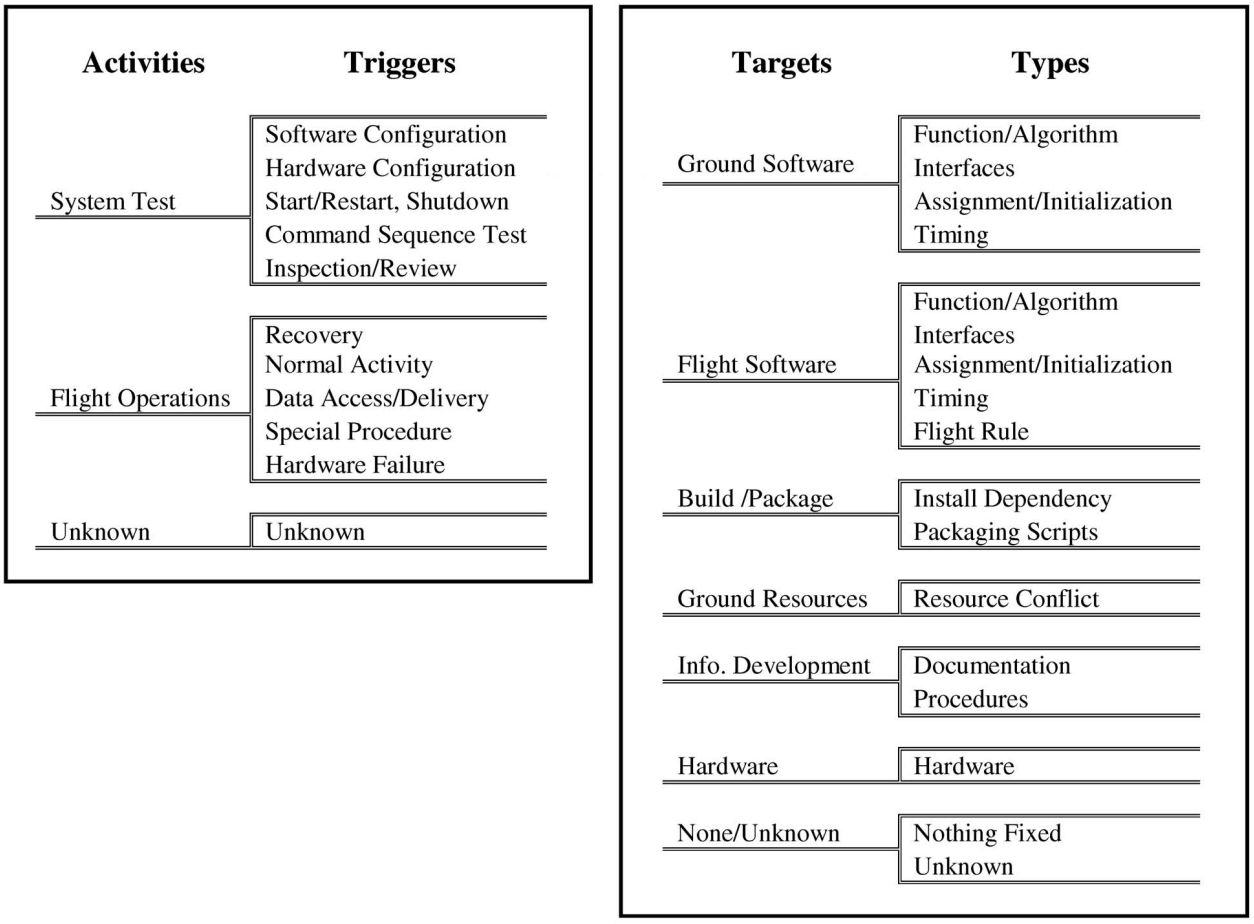


Fig. 1. Classification and activity distribution.

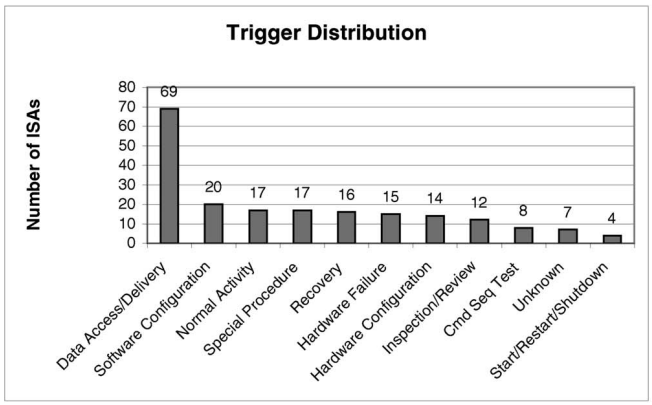
of a spacecraft is usually measured in years, so, corrective, adaptive, and perfective maintenance is performed on the software as the spacecraft proceeds through the phases of its mission [2]. For example, new software tailored to the next phase will often be uplinked to a spacecraft's computers prior to a navigational special maneuver, orbital insertion around a planet, sequence of scientific data-gathering, etc. Each of the planned updates, both to flight and to ground-support software, undergoes thorough system testing, usually at least in part subsequent to launch. Similar high levels of postdeployment maintenance are typical of other long-lived, high-integrity systems such as implantable medical devices or power-plant control systems.

5.2 Trigger

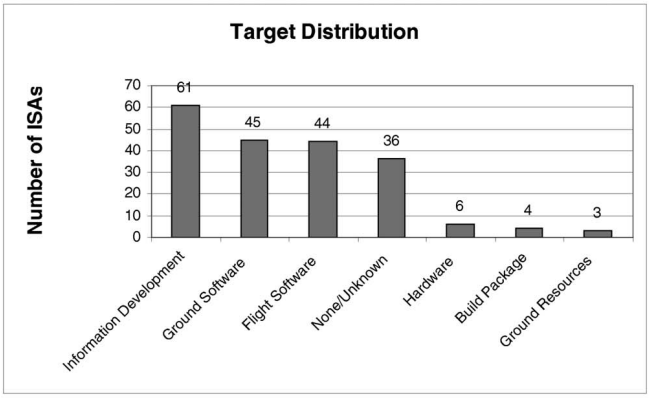
The most frequent Trigger (Fig. 2a) was "Data Access/Delivery" (35 percent).

While it was known that many *noncritical* anomalies involved problems with access to science or engineering data for operations or end-users, it was not expected that problems with data access were a catalyst for critical anomalies. The fact that data access and delivery was a trigger for almost a third of the safety-critical, postlaunch software anomalies prompted additional analysis, described below.

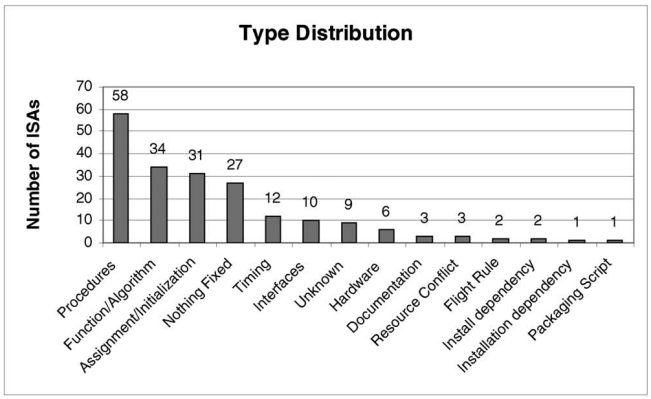
When the Activity was "System Test," the most frequent Trigger was "Software Configuration" (20 or 10 percent).



(a)



(b)



(c)

Fig. 2. Trigger, target, and type distribution.

These anomalies were caused by configuration management problems: files missing from directories, incorrect versions, problems with builds or deliveries, etc. Again, the surprise was that these triggered critical anomalies, rather than being inconveniences without significant consequences. These results suggest that acceptance of configuration problems as routine is inappropriate for critical systems.

We also investigated how often the Trigger for the safety-critical anomalies is an atypical or unusual situation. The basis for this query was a 1993 finding by Hecht that "Rare events were clearly the leading cause of failures among the most severe failure categories" [15]. Examination of the data (Fig. 2a) shows that such situations were well represented in the ISAs, with "Recovery" (an anomaly occuring during recovery from a fault condition), "Special Procedures" (e.g., calibration of an instrument), or "Hardware Failure" as the Trigger in one-third of the ISAs. Rare events (such as an unusual code path, an unforeseen usage scenario, a software request for data just as it became unavailable, or the failure of both redundant units) were thus a significant factor in the safety-critical anomalies [23].

On the other hand, most safety-critical anomalies did not occur at critical phases of the mission (e.g., during launch or as a spacecraft was inserted into orbit around a planet). Only 36 of the 199 anomalies (18 percent) occurred during a critical phase, defined for this study as launch, aerobraking of a spacecraft, orbital insertion around a planet, the separation of a probe from a spacecraft, entry into an atmosphere, descent and landing, landed operations, and planetary encounters (e.g., swing-by without landing). It was instead largely the complexity of the embedded software and its interfaces with the system, the dynamic nature of the operating environment, and the novelty of some activities that characterized the occurrence of anomalies.

We found that, in some cases, the triggers were not unique, i.e., the trigger for the anomaly was not one but instead more than one coincident failure, e.g., a double-point failure, or a chain of events with the most proximate trigger benign in isolation. It appears from the anomalies we studied that one consequence of a complex system highly coupled with its environment is that some anomalies have multiple triggers with the same degree of importance and with similar temporal proximities to the anomaly. This finding is consistent with previous experience of similar systems [15], [21] and with comments on ODC by a 1996 panel on statistical software engineering. They described the nonuniqueness as "multiple spawning," in which "although the defect types are mutually exclusive, it is possible that a fault may result in many defects and vice versa" [9].

Similarly, in some anomalies there were multiple targets. For example, in one anomaly, the temporary fix was to update a procedure to prevent recurrence of the scenario that had caused the anomaly by restricting operations. The permanent fix was to update the software by adding functionality to detect and handle such a scenario in the future. In such cases, there was more than one target with both (or all) of them essential for preventing recurrence of the anomaly.

The consequence for anomaly analysis is that sometimes the values of an attribute may not be orthogonal. In the case of multiple targets, guidelines as to which fix should be selected as the target were practical (e.g., the first fix). In the case of multiple triggers, guidelines as to which event to select as the trigger (e.g., most proximate in time to the anomaly) risked masking an understanding of dependencies that such anomalies can reveal. In those cases, an understanding of all the triggers involved in the rogue scenario might be necessary to preclude hazardous situations in the future. This suggests that causal analysis may play a larger role in the use of ODC for operational anomalies in complex, safety-critical systems than in

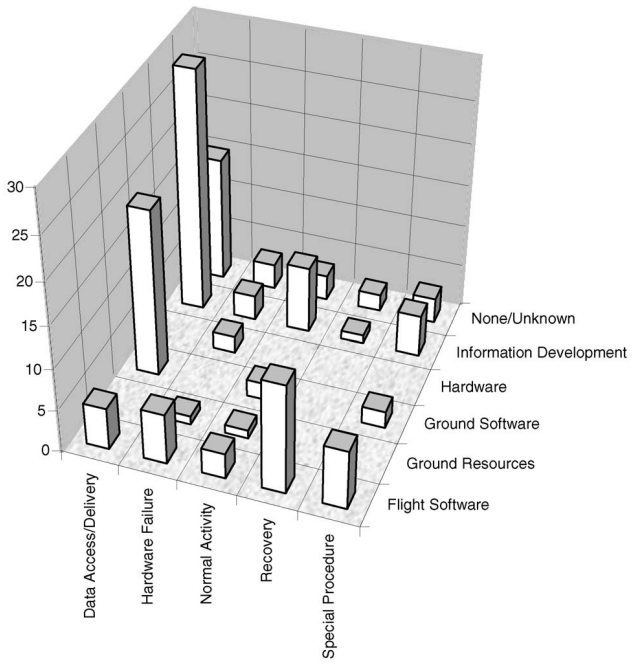


Fig. 3. Trigger/target distribution.

smaller, simpler systems. Failure scenarios in the former systems can involve subtle dependencies among multiple triggers, as well as multiple targets to prevent the recurrence of the anomaly.

5.3 Target and Trigger/Target

Fig. 2b shows the distribution of Targets in the systems. The most frequent Target (what was fixed) was "Information Development" (61 or 31 percent). Information Development is a change to a procedure or to documentation. The fact that the most frequent corrective action in response to critical operational anomalies was not to software but to documentation or procedures indicates the importance of adequately communicating domain knowledge to prevent critical operational anomalies.

Fig. 2b shows that there were also many anomalies with a Target of "Flight Software" (44 or 22 percent) or "Ground Software" (45 or 23 percent). In roughly a third of the cases where the Target was "Flight Software", there was either a new requirement for the software to handle rare but high-consequence events, or a new requirement for the software to compensate for hardware failures or limitations [23]. For example, looking at Fig. 3, which shows associations between some frequent triggers and targets, we see that anomalies caused by hardware failures more often result in corrective actions to flight software than to hardware. This is typical of systems in which the hardware simply cannot be accessed (such as spacecraft), as well as systems in which it is difficult to access the hardware (such as implanted medical devices), and systems in which the operating environment is dangerous (such as those in high-radiation, high-temperature, hostile, or poisonous settings). Such systems challenge the common assumption in defect analysis that what breaks is what gets fixed since, when hardware fails or degrades, software is often modified to compensate for the failure.

The remaining cases in which the Target was "Flight Software" were evenly divided between design fixes and code fixes. The lack of coding defects is not typical of defect analyses in non-safety-critical domains but is consistent with an earlier study that found few coding defects during integration and system testing of two spacecraft [22].

Fig. 3 also shows that anomalies triggered by problems with the access or delivery of data are most often fixed by changes to "Information Development", i.e., to documentation or procedures, or to ground software. We discuss the prevalence of procedural fixes (Type = "Procedure") more fully below.

5.4 Type and Trigger/Type

Fig. 2c shows that the most frequent Type of correction that was made in the anomalies was "Procedures." Table 2 is a presentation of associations between Triggers and Types, with a few of the least-frequent attribute values merged for

TABLE 2
Trigger/Type

Types:	*Triggers:*											
	Cmd Seq	Data Acc	HW Conf	HW Fail	Inspec Rev	Normal Activ	Reco-very	SW Conf	Spec	Start Shut	Unkn	Total
Ass/Init	3	4	2	3	2	3	3	3	5	3		31
Doc		1	1		1		2					5
Func/Alg	5	9	2	3	4		7	2	1	1		34
Hardware			4	2								6
Install								4				4
Interface		6	1			1		2				10
No Fix		9	2	3	2	2	2	1	3		3	27
Procedure		28	2	3	3	8		6	5		3	58
Resources				1		1		1				3
Timing		6				1	2		3			12
Unknown		6				1		1			1	9
Total	8	69	14	15	12	17	16	20	17	4	7	199

space reasons. The abbreviations used in the table are spelled out in Fig. 1. Looking at Table 2, there were 58 anomalies in which the Target was "Information Development" and the Type was "Procedures."

In 41 of the 58, a new or updated procedure was the correction. This suggests that some procedures needed for postlaunch operations were not in place at the time, and that these omissions contributed to safety-critical anomalies. In the other 17 of these 58, an existing procedure was not used when it should have been (44 or 23 percent). This usually reflected a breakdown in the dissemination of the procedure to operational personnel.

It appears to be the case that, since procedures were added or updated to fix (i.e., prevent the recurrence) of 41 ISAs, that incomplete procedures were a frequent contributing factor to critical anomalies. This suggests that a checklist of anomalous situations that required new procedures in previous operational systems might be useful to gauge the completeness of future missions' operational procedures.

When the Target was "Flight Software" or "Ground Software", the most frequent Types were "Function/Algorithm" (34) and "Assignment/Initialization" (31). Defect studies on other large software systems have shown similar results as far as the occurrence of these defects in software [5], [20], [22].

Twenty ISAs had the following three-way association: Activity = "Flight Operations," Trigger = "Data Access/Delivery," and Target = "Ground Software." Examination showed that all 20 were due to end-to-end downlink problems in receiving the data sent from the spacecraft. The existence of this pattern reflects the technical difficulties associated with remote communication with the spacecraft, e.g., problems with biases, residuals, jumps, and doppler banding.

For 27 (14 percent) of the anomalies, the Type was "Nothing Fixed." We conjectured that perhaps these anomalies occurred on spacecraft that were lost before a fix could be made. However, subsequent causal analysis showed that only in five of the 27 anomalies was loss of mission the reason that no fix occurred. In another eight, the anomalies reflected failures of communication between teams, i.e., the reported anomaly was, in fact, a false positive. The observed software behavior was actually correct but unexpected by the operator. The remaining were not fixed for a variety of reasons.

Results from normalized data for each project was similar but not identical. The most frequent Activity was "Flight Operations" both for all spacecraft considered together, and also for each individual spacecraft, considered separately. The most frequent Target was "Information/Development" for all spacecraft considered together, and also for three individual spacecraft considered separately (with these three comprising 149 of the 199 ISAs). The most common Trigger was "Data Access/Delivery" for all spacecraft, and also for four individual spacecraft (comprising 154 of the 199 ISAs). The most common Type was "Procedures" for all spacecraft, and also for four individual spacecraft (comprising 157 of the 199 ISAs). Since the study was limited to safety-critical, postlaunch, software ISAs, the number of anomalies on some of the individual spacecraft was small (e.g., eight on one system). Hence, assembling detailed profiles of individual spacecraft was not found to be useful.

TABLE 3
Signature of a Postlaunch Critical Anomaly

Attribute	*Value*
Activity:	Flight Operations
Trigger:	Data Access/Delivery
Target:	Information Development
Type:	Procedures

The three projects with the most ISAs in this study—one with 61, one with 59, and one with 29—were also checked individually. The individual project data supported the same findings, except that the third of these showed slightly more anomalies due to rare events than the other systems. Table 3 provides an anomaly signature in terms of the most frequent combination of attribute values.

6 Discussion and Evaluation

This section first discusses the internal and external validity of the results. It then considers how the anomaly results can be used to build and operate systems that may experience fewer critical, postdeployment anomalies.

6.1 Validity

The internal validity of the measurements, i.e., whether the ODC classification accurately measured the attributes of activity, trigger, target, and type for the anomalies, faced several challenges [13]. We describe here the major threats to validity and how each was addressed in the pilot study.

One set of threats involved the analysts, principally the consistency with which they classified anomalies. To address inconsistent classifications, we used independent classification by two analysts, described in Section 4. A previous study by El Emam and Wierczorek of the repeatability of defect classifications in ODC for code-inspection data found that the classification scheme was in general repeatable (not dependent on the individual) for two people in that context [11].

Another set of threats involved the quality of the raw data (the anomaly reports). The primary threats to validity here were:

1. Duplicates. These were rare (only four) and were deleted.
2. Incomplete or missing descriptions. Since the data were missing at random, we did not try to impute values [27] but recorded the value of missing Targets or Types as "Unknown."
3. Unclear or ambiguous descriptions. To avoid misclassifying anomalies due to misunderstanding, both analysts classified each anomaly and sought help from project personnel when needed.

4. Different projects. The process of anomaly reporting was not standardized across projects and project phases, so there were minor differences in which fields in the anomaly reporting forms recorded the information.

Within the spacecraft domain, the ODC measures gave indications both of being internally valid and of having predictive value, although the number of critical anomalies was too small to draw conclusions. For example, as we added anomaly data from additional spacecraft to the original set of three systems, the results did not change appreciably.

External validity (the extent to which the conclusions from the study can be generalized) is here limited by the domain of interest (operational spacecraft) and, perhaps, by the specific organization [1], [13]. Since many complex, embedded, high-criticality systems share with this domain a long operational phase and/or a high degree of maintenance postdeployment, we conjecture that future studies will support the generalization of these conclusions to similar systems. The pilot study was designed, documented, and run such that others can both verify and replicate the results. Replication might involve what Basili and Lanubile call "a family of experiments" [1], for example, on other NASA projects, other critical legacy systems, noncritical (rather than critical) spacecraft anomalies, or earlier development phases of the same systems.

6.2 Using Anomaly Analysis to Enhance Safety

The spacecraft domain shares with many other high-integrity domains operation in environments that are only partially understood, use of novel technologies, highly interactive subsystems, timing constraints for correct operations, and a high degree of autonomy. This study identified several directions for future improvement that may help reduce operational anomalies in these and other safety-critical systems:

- Integrating requirements engineering into maintenance activities [23], [25]. The finding that new software requirements continue to emerge post-launch in response to anomalies (to accommodate rare events or to compensate for hardware failures) indicates that the process of requirements specification, analysis, and verification continues postdeployment.
- Capturing requirements confusions to prevent future anomalies. Several anomalies turned out to require no corrective action because the software behaved as it should. The investigation of anomalies (including surprises) rather than just defects is valuable in that anomalies document sources of confusion on the part of the operational team or users. Using data from anomaly reports to supplement training and documentation can limit recurrence of these anomalies, especially in systems with long lifetimes and high turnover of personnel.
- Maintaining the documentation of system requirements allocated to operational procecures. For high-criticality systems, traceability of procedures to requirements (e.g., required sequencing of activities, required preconditions for activities, etc.) needs to be kept up-to-date and disseminated to operators.
- Mining anomaly reports to reuse knowledge regarding one system on other, similar systems, especially within a product line. Anomaly reports in this study often explicitly warned of similar vulnerabilities on future systems. Such feed-forward references need to be captured for inclusion in the inspections, reviews, and testcases of subsequent, similar systems, especially as organizations move toward a product-line approach. Anomaly analysis can be a valuable product-line asset.
- Challenging the operational assumptions. Anomaly analysis revealed patterns of data that were not consistent with some existing rationales. Identifying and investigating patterns of anomalies provided additional insight into operational risks beyond what was currently available through other methods.

7 Conclusion

The analysis of nearly 200 anomaly reports from seven spacecraft assembled a profile of operational, safety-critical software anomalies. Among the findings were that:

- The most frequent Trigger (what enabled the anomaly to occur) was difficulty accessing or delivering data. Most operational anomalies did not occur during critical mission phases.
- The most frequent Target (what was fixed) was information development rather than either flight or ground software. When there were changes to flight software, these often resulted from new software requirements to compensate for hardware degradation or to provide more robust fault protection against rare events.
- The most frequent Type of fix to was operating procedures. Anomalies did not result from failure to follow procedures but in the appearance or evolution of conditions requiring new or updated procedures to prevent recurrence.
- Anomaly reports that described inaccurate expectations by the operators (rather than actual defects) provided insights into how to improve documentation or training to prevent similar anomalies in the future.

System behavior, requirements, and procedures continued to evolve during operations in these systems, due largely to the dynamic nature of the environment, the degradation of hardware, and the complexity of the operational scenarios. An accurate understanding of the critical, operational anomalies that occurred on the spacecraft systems provided some insights into how to reduce recurring anomalies for these and other, similar systems.

The results presented here show that empirical analysis of operational anomalies can provide useful information not currently available to projects. The anomaly analysis contributes to a better understanding of what triggers and prevents safety-critical, operational anomalies.

ACKNOWLEDGMENTS

The initial research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the US National Aeronautics and Space Administration. It was partially funded by NASA's Code Q Software Program Center Initiative, UPN 323-08. The first author's research is supported in part by US National Science Foundation grants CCR-0204139 and CCR-0205588. The authors thank Tonja Harris and Don Potter for technical advice, Peter Neubauer for tool development, and Martin Feather, Tim Menzies, and the anonymous reviewers for feedback on an earlier draft.

REFERENCES

[1] V.R. Basili and F. Lanubile, "Building Knowledge through Families of Experiments," *IEEE Trans. Software Eng.,* vol. 25, no. 4, pp. 456-473, July/Aug. 1999.
[2] K.H. Bennett and V.T. Rajlich, "Software Maintenance and Evolution: A Roadmap," *The Future of Software Eng.,* A. Finkelstein, ed., 2000.
[3] P.G. Bishop and R.E. Bloomfield, "Worst Case Reliability Prediction Based on a Prior Estimate of Residual Defects," *Proc. 13th Int'l Symp. Software Reliability Eng.,* pp. 295-303, 2002.
[4] D.N. Card, "Learning from Our Mistakes with Defect Causal Analysis," *IEEE Software,* pp. 56-63, Jan./Feb. 1998.
[5] 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. Software Eng.,* vol. 18, no. 11, pp. 943-956, Nov. 1992.
[6] R. Chillarege and K.A. Bassin, "Software Triggers as a Function of Time—ODC on Field Faults," *Proc. Fifth IFIP Working Conf. Dependable Computing for Critical Applications (DCCA-5),* Sept. 1995.
[7] R. Chillarege, "Orthogonal Defect Classification," *Handbook of Software Reliability Eng.,* M. Lyu, ed., pp. 359-400 1995.
[8] R. Chillarege and K.R. Prasad, "Test and Development Process Retrospective—A Case Study Using ODC Triggers," *Proc. Int'l Conf. Dependable Systems and Networks,* pp. 669-678, 2002.
[9] Committee on Applied and Theoretical Statistics, *Statistical Software Engineering,* Nat'l Academy of Science, 1996.
[10] S. Dalal, M. Hamada, P. Matthews, and G. Patton, "Using Defect Patterns to Uncover Opportunities for Improvement," *Proc. Int'l Conf. Applications of Software Measurement (ASM),* 1999.
[11] K. El Emam and I. Wierczorek, "The Repeatability of Code Defect Classifications," *Proc. Int'l Symp. Software Reliablity Eng. (ISSRE),* 1998.
[12] N.E. Fenton and M. Neil, "A Critique of Software Defect Prediction Models," *IEEE Trans. Software Eng.,* vol. 25, no. 5, pp. 675-689, Sept./Oct. 1999.
[13] N.E. Fenton and S.L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach.* second ed. PWS Publishing, 1997.
[14] R.L. Glass, "Pilot Studies: What, Why, and How?" *J. Systems and Software,* vol. 36, pp. 85-97, 1997.
[15] H. Hecht, "Rare Conditions—An Important Cause of Failures," *Proc. Eighth Ann. Conf. Computer Assurance,* pp. 81-85, 1993.
[16] R.K. Iyer and I. Lee, "Measurement-Based Analysis of Software Reliability," *Handbook of Software Reliability Eng.,* M.R. Lyu, ed., pp. 303-358, 1995.
[17] Jet Propulsion Laboratory, "ICAP Anomaly Process, Glossary," Safety and Mission Assurance Information Systems, JPL, Pasadena, Calif., 1997.
[18] B. Kitchenham, "Guidelines for Conducting and Evaluating Empirical Studies," http://www.cs.utexas.edu/users/software/1999/Kitchenham/sld001.htm, 1999.
[19] S. Lauesen and O. Vinter, "Preventing Requirements Defects: An Experiment in Process Improvement," *Requirements Eng. J.,* vol. 6, no. 1, pp. 37-50, Feb. 2001.
[20] M. Leszak, D.E. Perry, and D. Stoll, "Classification and Evaluation of Defects in a Project Retrospective," *J. Systems and Software,* vol. 61, pp. 173-187, Apr. 2002.
[21] N. Leveson, *Safeware.* Addison-Wesley 1995.
[22] R. Lutz, "Analyzing Software Requirements Errors in Safety-Critical, Embedded Systems," *Proc. IEEE Int'l Symp. Requirements Eng.,* pp. 126-133, 1993.
[23] R. Lutz and I.C. Mikulski, "Operational Anomalies as a Cause of Safety-Critical Requirements Evolution," *The J. Systems and Software,* vol. 65, pp. 155-161, 2003.
[24] R. Lutz and I.C. Mikulski, "Requirements Discovery During the Testing of Safety-Critical Software" *Proc. Int'l. Conf. Software Eng.,* pp. 578-583, 2003.
[25] R. Lutz and I.C. Mikulski, "Resolving Requirements Discovery in Testing and Operations," *Proc. Int'l. Requirements Eng. Conf.,* pp. 33-41, 2003.
[26] M.G. Mendonça and V.R. Basili, "Validation of an Approach for Improving Existing Measurement Frameworks," *IEEE Trans. Software Eng.,* vol. 26, no. 6, pp. 484-499, June 2000.
[27] I. Myrtveit, E. Stensrud, and U.H. Olsson, "Analyzing Data Sets with Missing Data: An Empirical Evaluation of Imputaton Methods and Likelihood-Based Methods," *IEEE Trans. Software Eng.,* vol. 27, no. 11, pp. 999-1013, Nov. 2001.
[28] T.J. Ostrand and E.J. Weyuker, "The Distribution of Faults in a Large Industrial Software System," *Proc. Int'l. Symp. Software Testing and Analysis,* pp. 55-64, 2002.
[29] C. Seaman, "Qualitative Methods in Empirical Studies of Software Engineering," *IEEE Trans. Software Eng.,* vol. 25, no. 4, pp. 557-572, July/Aug. 1999.

Robyn R. Lutz received the PhD degree from the University of Kansas. She is a senior engineer at the Jet Propulsion Laboratory, California Institute of Technology, and an associate professor in the Department of Computer Science at Iowa State University. Her research interests include software safety, safety-critical product lines, defect analysis, and the specification and verification of requirements, especially for fault monitoring and recovery. Her research is supported by NASA and by the US National Science Foundation. She is a member of IEEE and of the IEEE Computer Society.

Inés Carmen Mikulski received the MS degree in mathematics from the State University of New York at Buffalo. She is a senior engineer at the Jet Propulsion Laboratory (JPL), California Institute of Technology. Prior to joining JPL, she worked as a software developer, database administrator, a new-technology evaluator/implementer, and a software development manager in both commercial and educational institutions. Her current research interests center on development of a project-level and institution-level metrics program as part of JPL and NASA's software quality improvement initiative. She is a member of the IEEE Computer Society.