Research Infusion Collaboration: Finding Defect Patterns in Reused Code

Robyn Lutz and Scott Morgan

Contributors: Tuan Do, Carmen Mikulski, Martha Berg Strain, Steve Rockwell, and Belinda Wilkinson

Final Report
November18, 2004
Jet Propulsion Laboratory/Caltech

This work was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. It was funded by NASA's Office of Safety and Mission Assurance Software Assurance Research Program.

Overview

Point of Contact:

Dr. Robyn R. Lutz
robyn.r.lutz@jpl.nasa.gov
515-294-3654

Introduction

The "Finding Defect Patterns in Reused Code" Research Infusion Collaboration was performed by Jet Propulsion Laboratory/Caltech under Contract 104-07-02.679 102197 08.14.4. This final report describes the collaboration and documents the findings, including lessons learned.

Problem Statement

The research infusion collaboration characterized, using Orthogonal Defect Classification, defect reports for code that will be reused in mission-critical software on Deep Space Network Antenna controllers. Code reuse is estimated to be 90%, so it is important to identify systemic defects, or patterns, prior to reuse of this code. The work also identified ways to avoid certain types of defects and to test more efficiently.
The primary objectives of the project were:

- To analyze defect patterns of the code to be reused based on the defects' Orthogonal Defect Classification (ODC)
- To achieve a successful infusion of ODC to a project

Application of the technology to the target project

ODC is an established method for analyzing defect reports, originally developed at IBM [Chillarege] and later adapted to the spacecraft domain at JPL [Lutz]. ODC provides a way to "extract signatures from defects" and to correlate the defects to attributes of the development process.

The ODC-based approach uses four attributes to characterize each anomaly report. The first is the Activity, which describes when the anomaly occurred (e.g., Build Test). The Trigger describes the environment or condition that had to exist for the anomaly to surface (e.g., Hardware/Software Interaction). The Target characterizes the high-level entity that was fixed in response to the anomaly's occurrence (e.g., the Antenna Control Assembly). Finally, the Type describes at a lower-level the actual fix that was made (e.g., Documentation, Missing Procedure, or Function/Algorithm).

Data Collection and Analysis

The research infusion task classified 167 test reports from the Project's Harvest defect database. They were generated during build tests on the reuse code for the Deep Space Network antenna controllers. The research infusion task used a customized version of Orthogonal Defect Classification to analyze the anomalies. Section 3 of the final report analyzes the results of the ODC classification of these test reports in terms of the ODC

Activity, Triggers, Targets, and Types. It also describes nine defect patterns of interest found by the ODC analysis. The ODC analysis of the test reports produced five recommendations for possible reductions in the undesirable defect patterns as the DSN antenna project continues, described below.

Summary and Lessons Learned

The research infusion project was a successful collaboration between the research team with experience in ODC and the DSN team with experience in the domain of antenna controllers. The Project Manager got useful data regarding the reuse of the software for the next project, which was the BWG Antenna Controller Replacement Retrofit project. The graphical visualization capabilities (pivot tables, bar charts, etc.) once the test reports were classified were useful in that they allowed browsing the results for defect patterns of concern.

In Section 4 we discuss three key lessons learned that apply to future users of ODC:

Match the technology to the project. A prerequisite for using ODC on a project is that the project already have a defect tracking system in place and that they are using it to collect defect data. This prerequisite was met in this project. A second prerequisite is that the ODC analysts will have access to domain experts to answer questions where the defect report is incomplete or unclear. This prerequisite was also met in this project. A third prerequisite for using ODC on a project is that the project adequately scope the application in terms of which defect reports will be analyzed. This was determined collaboratively in this project.

Provide training and continued support. For small projects such as the Antenna Controller Replacement Task (3.2 FTE), informal one-on-one training seemed to work well. From our previous SARP work, we could also provide quite a few explanatory documents and sets of examples from eight spacecraft using ODC (see References). It is highly recommended that individuals with experience in the new technology subsequently be available for continued, low-level support as questions arise. Weekly telecons worked well during this period.

Run a small experiment initially. We had learned from our previous applications of ODC to projects that doing an initial classification on a small subset of defects saved time. If any tweaks to classification categories are needed (as they often are), or if any differences of opinion as to the process (e.g., how to scope the defects to be classified) exist, they are thus discovered early. At that point, it is easy and fast to adjust the classification categories without significant re-work. Subsequently scaling up the number of defects considered is then straightforward.

1.0 Introduction

The "Finding Defect Patterns in Reused Code" Research Infusion Collaboration was performed by Jet Propulsion Laboratory/Caltech under Contract 102197-08.14.4. This final report describes the collaboration and documents the findings, including lessons learned.

1.1 Problem Statement

The research infusion collaboration characterized, using Orthogonal Defect Classification, defect reports for code that will be reused in mission-critical software on Deep Space Network Antenna controllers. Code reuse is estimated to be 90%, so it is important to identify systemic defects, or patterns, prior to reuse of this code. The work also identified ways to avoid certain types of defects and to test more efficiently.
The primary objectives of the project were:

- To analyze defect patterns of the code to be reused based on the defects' Orthogonal Defect Classification (ODC)
- To achieve a successful infusion of ODC to a project

1.2 Target Project

The collaboration was between the research team at Jet Propulsion Lab/CIT that had extended Orthogonal Defect Classification (ODC) to the spacecraft domain and the project team at Jet Propulsion Lab/CIT that was developing and testing the software to be used to upgrade the Beam Waveguide (BWG) antenna controllers as part of the Antenna Controller Replacement Task. The Antenna Beamwave Guide is an integral part of NASA's Deep Space Network used to support JPL and other spacecraft in their missions by providing telemetry, science data, and commanding.

1.3 Collaboration Scope

The research team helped adopt the ODC classification to the Antenna task domain and trained the users on how to classify the defects. The project collected the defect classification data in their existing defect reporting system (CCC-Harvest) as they were discovered. A subset of the classified defects was analyzed by the project with support from the research team to find patterns and formulate recommendations.

The expected benefits include:

- Identify defect patterns prior to reuse of the code
- Recommendations for ways to reduce certain types of defects
- More efficient testing of reuse code

1.4 Application of the technology to the target project

ODC is an established method for analyzing defect reports, originally developed at IBM [Chillarege] and later adapted to the spacecraft domain at JPL [Lutz]. ODC provides a

way to "extract signatures from defects" and to correlate the defects to attributes of the development process.

The ODC-based approach uses four attributes to characterize each anomaly report. The first is the Activity, which describes when the anomaly occurred (e.g., Build Test). The Trigger describes the environment or condition that had to exist for the anomaly to surface (e.g., Hardware/Software Interaction). The Target characterizes the high-level entity that was fixed in response to the anomaly's occurrence (e.g., the Antenna Control Assembly). Finally, the Type describes at a lower-level the actual fix that was made (e.g., Documentation, Missing Procedure, or Function/Algorithm).

The steps in the application are described below in section 2.

a) The initial schedule (finish by Q4) was not met, primarily because funding did not arrive until Q3.

b) The time required to introduce the technology was an initial set of training meetings on ODC (approx. 8 hours total) followed by on-going availability for questions at a low level of support (a weekly telecom and some email). The time required to apply the technology had been previously calculated as approximately 1-3 minutes/defect [Lutz, 3/04].

c) The primary risk was that Carmen Mikulski, i.e., half of the research team, retired July 1. However, she did a great job of training people before she left and we were able to tie up any loose ends in a joint presentation to the project manager before she left, so the transition was smooth.

2.0 Methodology

The steps in the application of the technology to the project were:

1. Initial planning meetings between the research team and the project team to agree on the steps and the initial set of objectives. The Project identified a contact person to help with domain questions.
2. Project provided access and guidance to their defect classification database (CCC/Harvest) and their goals and schedules.
3. Research team customized the defect categories in ODC to the project's needs and got feedback from the project manager.
4. Selected (with project help) and classified a subset (27) of the Internal Anomaly Reports (IARS) generated by the project. These all had priority field = 1 (the highest priority) and were from software to be reused in the 70-meter antenna.

   Each anomaly was classified twice, once by Robyn and once by Carmen. If there were discrepancies between these two classifications, they were reconciled in joint discussions. Both authors have experience on flight projects at JPL, but neither is involved with the testing or operations of the system under study. A fuller description of the classification process appears in [Lutz, 3/04].

5. Research team gave a presentation in June, 2004 to project personnel on the customized defect classifications and initial results on the subset of IARs.
6. Project personnel provided additional feedback (e.g., added a subsystem to the study).
7. Trained two people (Belinda Wilkinson and Tuan Do) to use ODC.
8. Tuan Do (in software assurance and already working with the project in another capacity) performed ODC classification of the remaining 140 test reports and produced pivot tables of the results.
9. Steve Rockwell, from the project, met regularly with Tuan Do to answer domain questions.
10. Research team identified defect patterns and produced recommendations for final report.

The primary metric was the number of defects analyzed in the reused software. This number was 167. A secondary metric was the number of defect patterns identified in the reused software. Nine defect patterns are described below. Another secondary metric was the number of recommendations made. We made two kinds of recommendations: to the project, based on the ODC analysis, and to future users, based on our experience with the research infusion. Five recommendations to the project are described in section 3. Three key recommendations for future users of ODC are discussed in section 4

Everyone knew that the funding was minimal and delayed, so worked hard to keep costs of the research infusion down and to get started soon enough to keep momentum. The project generously contributed the time that they spent reviewing the customized ODC categories, answering domain questions, and providing feedback on the ODC classification of the test reports. The Software Quality Initiative at JPL generously contributed Carmen Mikulski's and Belinda Wilkinson's time on this effort as part of their support for ODC (and defect metrics in general) at JPL.

3.0 Data Collection and Analysis

The task analyzed 147 Internal Anomaly Reports and 20 New Requirement Reports from the Project's Harvest defect database. Because almost all of the reports were IARs, we use the shorthand "IAR" to refer to both the IARs and the NRs in the discussion that follows. All the IARs were generated during testing of the software that will be reused. This section analyzes the results of the ODC classification of these IARs and describes the defect patterns found.

3.1 ODC Defect Patterns

The ODC analysis identified nine defect patterns of interest. They are shown in italics in the discussion below. Each of the four ODC attributes—Activity, Trigger, Target, and Type are described.

ODC Activity. Note that since all the IARs were generated during the same activity, namely the build testing of the software that was to be reused, that the ODC Activity category for all of them is the same. If the task were expanded to apply ODC during other testing activities, or during operations, then there would be a diversity of activity categories.

ODC Triggers. The distribution of the ODC triggers of the IARs shows that *more of them (76 of 167 or 46%) have "Capability/Invocation" as the trigger than any other trigger.* Capability/Invocation anomalies occur when the anomaly was detected while testing the response to a particular antenna function.

The high number of IARs with Capability/Invocation as the trigger is appropriate for the testing phase, since it shows that anomalies are being generated primarily through the exercising of the functional capabilities of the system.

The second most common ODC trigger was "Inspection/Review" (45 of 167, or 27%). These anomalies occur when a problem has been found by manual inspection of the code or test results. Some of these anomalies might be able to be found earlier in the development process by earlier inspection of the code. These anomalies also reflect the familiarity of the testers with the domain, in that they frequently noticed small inconsistencies that might not have been visible until later in testing to someone less familiar with the system.

The third most common ODC trigger was "User Interface" (29 of 167, or 17%). This is perhaps a somewhat high number at this point in testing. It appears to reflect the tweaking of the code in the reuse setting to get it to provide the expected user interfaces. We speculate that this ODC trigger (User Interface) is more common in testing of software for reuse than in the original testing. The reason is that in the original testing the discrepancies between the code and the original, expected interfaces would appear in unit testing, whereas with reuse software, the code may be consistent with the original interface requirements but not with the reuse interface requirements in the updated, integrated system. The scope of the work done here did not allow us to evaluate this hypothesis.

The very low number of IARs with a trigger of "Recovery" indicates either little fault-protection code, little testing of off-nominal (fault-protection) scenarios, or fault protection in good shape. This defect pattern can thus be either positive or negative, and requires further investigation by the project (i.e., root cause analysis on the subset of IARs with a defect type of "Recovery") to fully understand.

ODC Targets. The ODC targets (what was fixed) were the eleven software subsystems. As shown in Figure 1, five of the eleven subsystems account for 80% of the IARs (135 of 167). These five subsystems are ACA (Antenna Control Assembly). AMC (Antenna Monitor and Control), APS (Antenna Pointing Subsystem), CTL (Controller), and DSP (Display). *One of these subsystems, CTL, accounts for 23% of the IARs.* This pattern appears to be due to the complexity of the functional requirements that are implemented in CTL. CTL might thus be a high-priority candidate for additional inspection or review prior to beginning the testing phase.

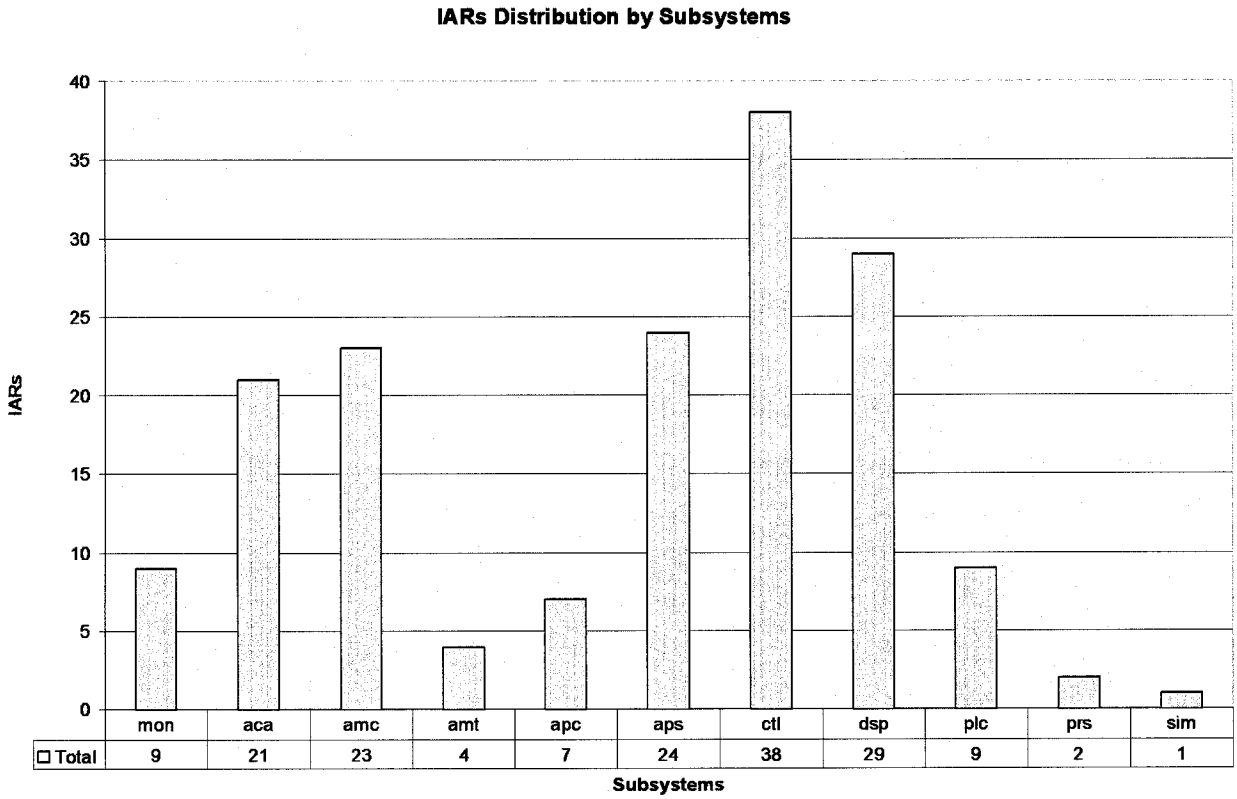


Figure 1. Distribution of Anomalies among ODC Targets

ODC Types. The distribution of the ODC types of IARs in Figure 2 shows that the *most frequent anomaly type is "Function/Algorithm" (115 of 167, or 69%).* The type is Function/Algorithm when the anomaly is the result of the omission or incorrect implementation of significant capability, requirements, end-user interfaces, or global data structures that was fixed by re-implementing an algorithm.

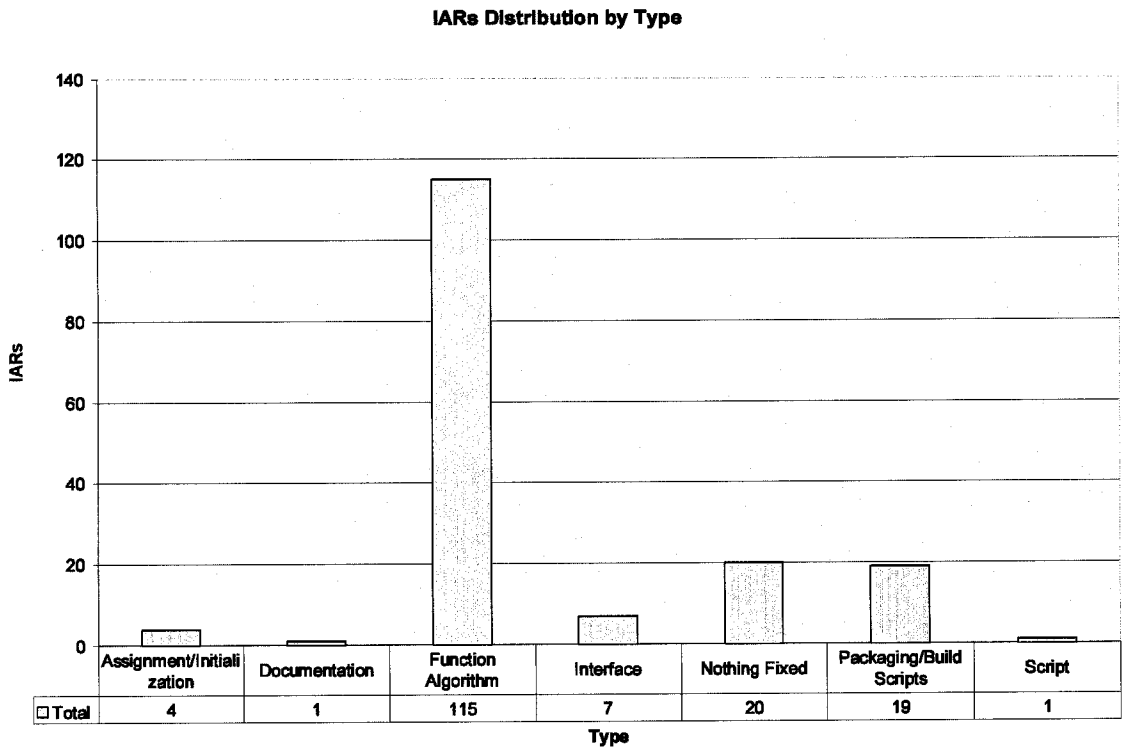


Figure 2. Distribution of ODC Anomaly Types

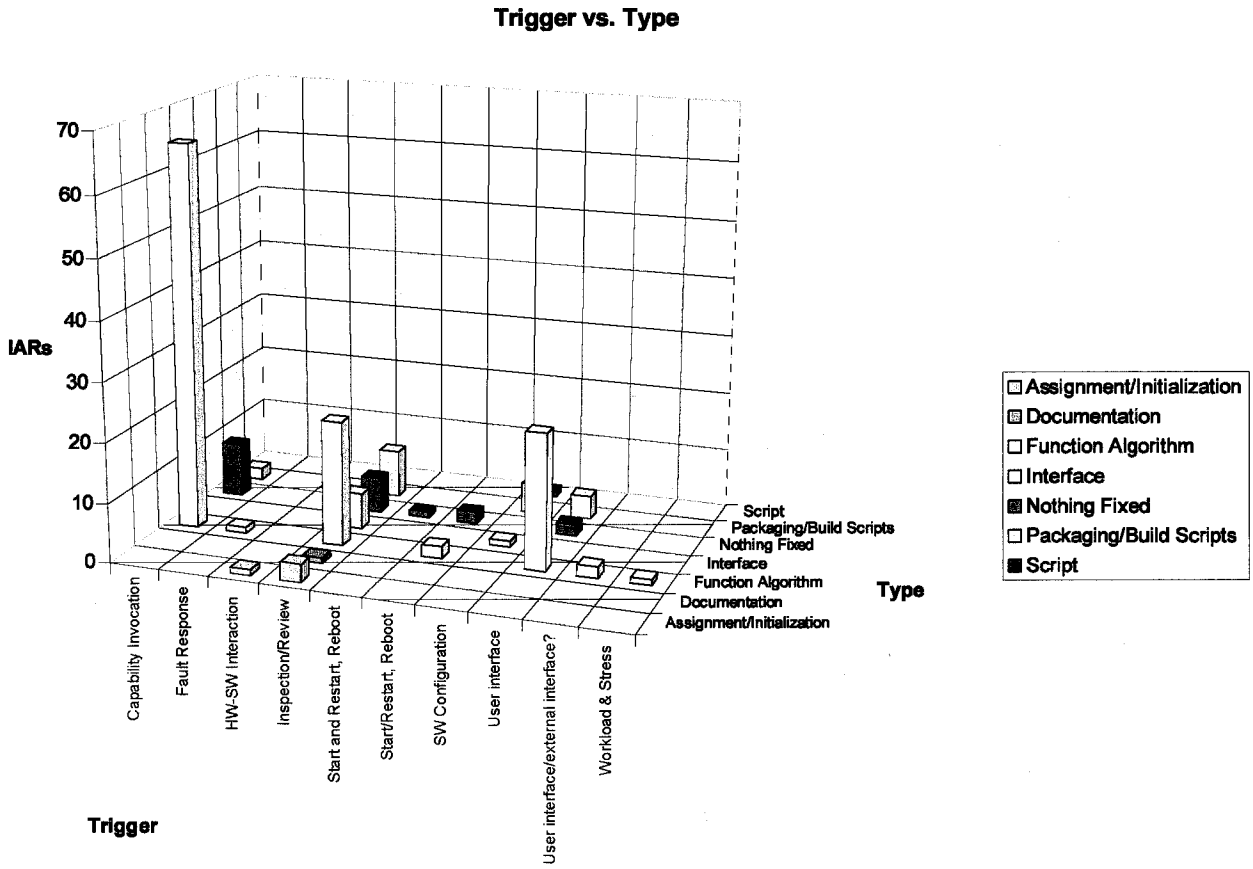
Two other ODC types are somewhat frequent. The first type is *"Packaging/Build Scripts" (19/167 or 11%).* This ODC type is selected when the defect is encountered during the system build process and was the result of the library system, faulty change management or version control, or when a routine build of the software version does not proceed as expected.

The other ODC type that appears with some frequency is "Nothing Fixed" (20/167 or 12%). As noted in our previous ODC work on MER integration and system testing, anomalies of this type merit special attention. On other projects we found that some anomalies with a "no fix" ODC type described cases where the software behavior was correct, but the testers thought that the software behavior was wrong. In these situations "no fix" sometimes left the door open for the same misunderstanding to recur in testing or operations [Lutz, 4/04]. Such anomalies are false-positives in that the tester mistakenly reported an anomaly where none existed. These anomalies can function as a "crystal ball" into future, operational misunderstandings of the software behavior. We thus suggested that in some cases, rather than "no fix" that the anomaly prompt additional training or documentation that could preclude future misunderstandings. Given the potential usefulness of "nothing fixed" anomalies, we went back and looked at the description of each such IAR. Four of the anomalies in this set met the criteria and were identified to the project.

ODC Trigger/Type. The distribution of the ODC Triggers and Types in Figure 3 shows that *39% (65 of 167) of them have an ODC Trigger of "Capability Invocation" and an*

ODC Type of "Function/Algorithm". This is appropriate for testing, reflecting that much of the testing was exercising individual functional requirements and fixing the bugs found in the logic that implemented these requirements.

Figure 3. Distribution of ODC Triggers/Types



ODC Target/Type. The distribution of the ODC Targets and Types in Figure 4 shows that the ODC type of "Function/Algorithm" is most frequent for all targets except AMC (Antenna Monitor and Control). The fact that this defect pattern holds across all-but-one subsystem confirms that anomalies of this type drive the number of anomalies on the project. As noted before, this is an indication of a healthy testing process.

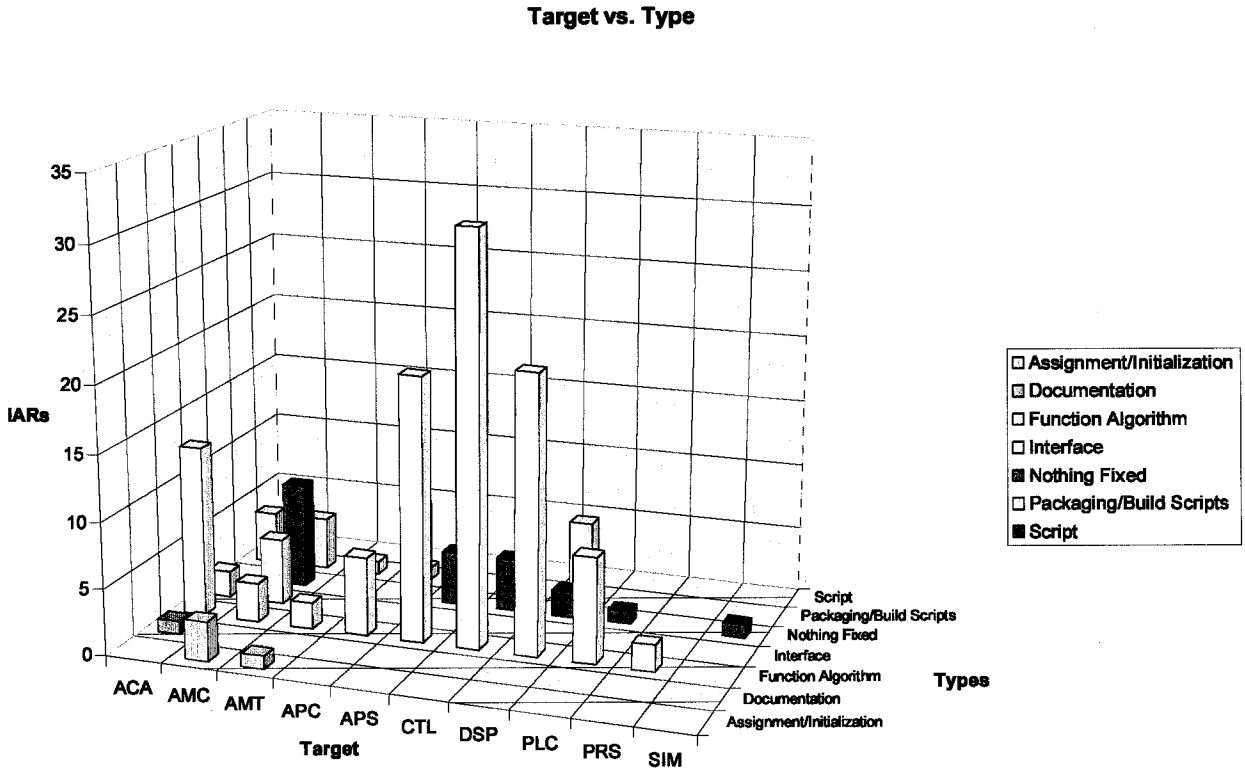


Figure 4. Distribution of ODC Targets/Types

3.2 Recommendations for avoiding undesirable defect patterns in the integrated software

The ODC analysis of the test reports produced the following five recommendations for possible reductions in the undesirable defect patterns as the DSN antenna project continues:

1. The 20 test reports whose ODC trigger is Inspection/Review might be able to be identified earlier in the development process by enhancing the code review process prior to build tests.
2. The use of the IARs to record "to-do" lists for future development or testing (much like testing log notes) muddies the picture of defect metrics somewhat. This complicates the collection of defect metrics, especially across projects, in that some reports do not describe anomalies. The additional information is invaluable and needs to be collected, but perhaps not in Harvest.
3. The research team sometimes found it difficult to classify the defects based on the limited information found in the IARs. In addition, the capabilities (such as search facilities) of the defect-management toolset, Harvest is sometimes awkward or inflexible.

4. ODC could be readily incorporated into Harvest so that the ODC classification is partially automated and done at the same time that the IAR is processed in Harvest.
5. Refine the ODC types into additional subtypes at the request of the project. In the future, it may be useful to have subcategories added (for example, to split up the function/algorithm trigger into several triggers).

3.3 Summary of ODC Analysis for the Reuse Software.

The ODC analysis of the IARs provides one indicator of the software's readiness for reuse. In summary, the defect patterns that the ODC analysis reveals tend to show a healthy testing program with most of the anomalies occurring as the software is exercised to demonstrate required functional capabilities. Most notably, the ODC analysis does not show "red flags" in the key areas of problems with fault protection, with boot-up and reboot issues, or with unanticipated hardware/software interactions. These are all areas of special concern if the software is to be reused and integrated into an updated system because it indicates problems with controlling the complexity of the software and its interactions with the system. The fact that the reuse software here does not have many IARs with those triggers is good.

4.0 Summary and Lessons Learned

The research infusion project was a successful collaboration between a research team with experience in ODC and the DSN team with experience in the domain of antenna controllers. 167 Internal Anomaly Reports generated during build tests on the reuse code for the Deep Space Network antenna controllers were classified using a customized version of Orthogonal Defect Classification. Nine defect patterns of interest were identified. The lessons learned are captured below in three recommendations for future users of ODC.

The Project Manager got useful data regarding the reuse of the software for the next project, which was the BWG Antenna Controller Replacement Retrofit project. The graphical visualization capabilities (pivot tables, bar charts, etc.) once the IARs were classified was useful in that it allowed browsing the results for anomaly patterns of concern.

We identified three lessons learned that apply to future users of ODC:

Match the technology to the project. A prerequisite for using ODC on a project is that the project already have a defect tracking system in place and that they are using it to collect defect data. This prerequisite was met in this project. A second prerequisite is that the ODC analysts will have access to domain experts to answer questions where the defect report is incomplete or unclear. This prerequisite was also met in this project. A third prerequisite for using ODC on a project is that the project adequately scope the application in terms of which defect reports will be analyzed. The scope may be all defects from a certain phase (e.g., of testing) or all defects from a certain priority level

(e.g., the most-critical defects). This prerequisite was met collaboratively in this infusion task by discussion between the project team and the research team.

Provide training and continued support. For small projects such as the Antenna Controller Replacement Task (3.2 FTE), informal one-on-one training seemed to work well. From our previous SARP work, we could also provide quite a few explanatory documents and sets of examples from eight spacecraft using ODC. It is highly recommended that individuals with experience in the technology subsequently be available for continued, low-level support as questions arise. Weekly telecons worked well during this period.

Run a small experiment initially. We had learned from our previous applications of ODC to projects that doing an initial classification on a small subset of defects saved time. If any tweaks to classification categories are needed (as they often are), or if any differences of opinion as to the process (e.g., how to scope the defects to be classified) exist, they are thus discovered early. At that point, it is easy and fast to adjust the classification categories without significant re-work. Subsequently scaling up the number of defects considered is then straightforward.

5. 0 Acronyms and Definitions

ACA Antenna Control Assembly

ACR Antenna Controller Replacement task

AMC Antenna Monitor and Control

APS Antenna Pointing Subsystem

BWG Beam Waveguide

CTL Controller

DSN Deep Space Network

DSP Display

ODC Orthogonal Defect Classification

6.0 References and Resources

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 Transactions on Software Engineering*, Nov. 1992, pp. 943-956.

R. Lutz and C. Mikulski, "Empirical Analysis of Safety-Critical Anomalies during Operations," *IEEE Transactions on Software Engineering*," vol. 30, no., 3, March, 2004, pp. 172-180.

R. Lutz and C. Mikulski, "On-Going Requirements Discovery in High-Integrity Systems," *IEEE Software*, Vol. 21, 2, March-April, 2004, pp. 19-25.

R. Lutz and C. Mikulski, "Resolving Requirements Discovery in Testing and Operations," with C. Mikulski, *Proc. 11th IEEE Requirements Engineering Conference (RE'03)*, Sept. 8-12, 2003, Monterey Bay, CA, pp. 33-41 ("Best Experience Paper of RE'03").

T.Menzies, R. Lutz and C. Mikulski "Better Analysis of Defect Data at NASA," *Proc. 15th International Conference on Software Engineering and Knowledge Engineering (SEKE'03)*, July 1-3, 2003, San Francisco, CA

R. Lutz and C. Mikulski, "Patterns of Software Defect Data on Spacecraft," *International Conference on Space Mission Challenges for Information Technology (SMC-IT'03),* Pasadena, CA, July 13-16, 2003.

R. Lutz and C. Mikulski, "Requirements Discovery during the Testing of Safety-Critical Software," *Proc. 25th International Conference on Software Engineering (ICSE'03),* May 3-10, 2003, Portland, OR, pp. 578-583.