Defect Measurement and Analysis of JPL Ground Software: 
A Case Study

John D. Powell & John N. Spagnuolo Jr.

Jet Propulsion Laboratory, California Institute of Technology 
4800 Oak Grove Rive 
M/S 125-233 
Pasadena, CA 91109 

Phone: 818-393-4355 

Focus Issue: 
Software and mission assurance, system integration, and testing
Abstract

Ground software systems at JPL must meet high assurance standards while remaining on schedule due to relatively immovable launch dates for spacecraft that will be controlled by such systems. Toward this end, the Software Quality Improvement (SQI) project’s Measurement and Benchmarking (M&B) team is collecting and analyzing defect data of JPL ground system software projects to build software defect prediction models. The aim of these models is to improve predictability with regard to software quality activities. Predictive models will quantitatively define typical trends for JPL ground systems as well as Critical Discriminators (CDs) to provide explanations for atypical deviations from the norm at JPL. CDs are software characteristics that can be estimated or foreseen early in a software project’s planning. Thus, these CDs will assist in planning for the predicted degree to which software quality activities for a project are likely to deviation from the normal JPL ground system based on pasted experience across the lab.

The case study discussed in this abstract will illustrate analysis of a relatively rich source of JPL defect data for ground system software. This data allows conclusions to be drawn regarding:

- Effort to repair defects leading to a generalized cost metrics for both the test and operations phases
- The use of CDs such as personnel turnover and experience resulting in higher defect densities for portions of the software system
- The use of the CDs describing subsystem interconnectivity and complexity resulting, on average, in unusually high levels of effort to fix defects based on the size of a subsystem
- The variations in the ability or inability of defect trends to remain consistent during both software testing and operations.

Preliminary predictive metrics regarding defects of varying types and the amount of effort associated with repairing those defect in software code of a given size.

Once access to a project’s defect data has been granted, the methodology used to collect and analyze defect data for projects at JPL can be best described as a four-part process. First, to the extent that the data will allow, the defects are

Figure 1: Defect Classification Hierarchy
characterized according to a classification hierarchy. (See Figure 1) The Hardware defects are eliminated from consideration. Then the software, as a whole is classified as either flight software, ground software or instrument software. Second, the software is decomposed into logical units referred to as Software Development Sets (SDS). An SDS is similar, but not necessarily identical to a Computer Software Configuration Item (CSCI). It is at the SDS level where the defect characteristics are identified for each software defect. Third, the defect characteristics are then combined with other characteristics such as software size or test phase of discovery to for relationship that are normalized across SDSs. Forth and finally, the defect characteristics and relationships are grouped by software versions, releases and/or builds to examine trends over time.

The desired set of data with respect to each defect include:
- Test phase of discovery for a defect
- Development phase of introduction (Requirements, Design, Code & Test) for a defect
- Criticality of the defect
- Effort to fully repair the defect.

Desired data regarding the overall SDS in which the defect resides includes 1) software size (KSLOC), 2) Complexity (CPLX from COCOMO), 3) Interconnectivity (Coupling). The combination of defect data and SDS data allows relationships for the software as a whole to be examined for trends. The temporal groupings of interest involve grouping the date by version, build, release etc... to examine relationship trends over time.

The case study project data is taken from a JPL ground system and consists of 580 defects
- 453 defects recorded during the test phase of development
- 127 defects recorded during system operations.

The overall defect density was 1.2 defects per LKSLOC. It was discovered that the effort to repair defects during operation of the ground software system was approximately 50% greater than that of defects found during software testing. (See Figure 2) This is largely due to a high percentage of defects occurring in software test that took less than 8 hours of effort to repair. Roughly 70% of the defects found in test fell into the less-than-8-hours-to-repair category while less than 40% of the operations defects

![Figure 2: Effort to Repair defects in Test versus Operations](image)
could be group in the 8 hours or less category. In order to comparably encompass approximately 70% operations defects, all categories up to and including the 16-hours-or-less category must be included. It can be observed that a general linear defect density trend exists when the number of problem reports is plotted against LKSLOC. There exists one exception to this trend in the case study project. (See Figure 3) The outlier is explained by CDs having to do with the fact that wholesale turnover of the part of the team working on that portion of the system occurred and the new team had far less experience that the original development team. The data in the case study goes on to show that while this outlying portion of the system had an inordinate number of problem reports it did not require more than the average number of work hours to repair them. Therefore the allocation of personnel to repair defects based on the number of defects in existence is not always a proper use of resources. Various CDs and quantifiable data must be consulted in order to make informed decisions about resource allocations to quality activities throughout the software system. Fortunately, many of the CDs lend them self to early estimation before the software system enters the later test phases of the development lifecycle.

Acknowledgement

The work described in this abstract was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.