Predictive Software Design Measures

by

Randall J. Love

Thesis submitted to the faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

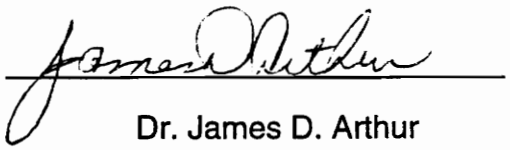
Master of Science

in

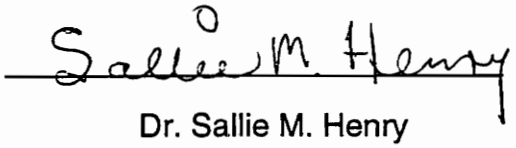
Computer Science

APPROVED:

Dr. Richard E. Nance, Chair



Dr. James D. Arthur



Dr. Sallie M. Henry

October, 1994

Blacksburg, Virginia

c.2

LD
S655
V855
1994
L684
c.2

Predictive Software Design Measures

by
Randall J. Love
Richard E. Nance, Chairman
Computer Science
(ABSTRACT)

This research develops a set of predictive measures enabling software testers and designers to identify and target potential problem areas for additional and/or enhanced testing. Predictions are available as early in the design process as requirements allocation and as late as code walk-throughs. These predictions are based on characteristics of the design artifacts prior to coding.

Prediction equations are formed at established points in the software development process called milestones. Four areas of predictive measurement are examined at each design milestone for candidate predictive metrics. These areas are: internal complexity, information flow, defect categorization, and the change in design. Prediction equations are created from the set of candidate predictive metrics at each milestone. The most promising of the prediction equations are selected and evaluated. The single "best" prediction equation is selected at each design milestone.

The resulting predictions are promising in terms of ranking areas of the software design by the number of predicted defects. Predictions of the actual number of defects are less accurate.

Acknowledgments

My thanks go out to the friends and family that contributed to this thesis research. Most importantly, thanks to Dr. Richard Nance whose guidance and support made this all possible. Thanks also to the other members of my committee, Dr. James Arthur and Dr. Sallie Henry for taking the time to review this thesis and provide valuable comments and suggestions. You have all contributed much more to my personal and professional development than I ever thought possible and I thank you again.

Thanks to Lisa Cox, Sandra Griffith, and Patricia Hubble of the Systems Research Center for all their help. Thanks also to the entire Computer Science Department at Virginia Tech; Special thanks to Jessie Eaves for her help with VTVM1. Thanks also to Tammi Johnston and Dr. Dennis Kafura for there help with conference room scheduling. Thanks and appologies for my poor writing to my editor Sylvia DeSantis.

Thanks are much overdue due to my friends and associates in both Virginia and Connecticut. Thanks to Dr. Arthur, Jamie Evans, Ben Keller, Colin Klipsch, Dr. Nance, and Dr. Ernie Page for their friendship and for giving me breaks from my research. I must also point out that I could not have completed this thesis without the help of Ernie Page and I cannot thank him enough.

I am especially grateful to my parents James and Arlee Love, my big sister Lisa, brother-in-law Mark, niece Maxine, little brother Mike, sister-in-law Paula,. and my favorite grandmothers Helen Love and Maxine Rosenberg. I cannot imagine completing my studies without all your support.

Table of Contents

1 Introduction 1
1.1 Problem Statement and Related Work 1
1.2 Motivation and Goals 3
1.2.1. Major Goals 3
1.2.1.1 Creation 4
1.2.1.2 Deployment 6
1.2.2 Minor Goals 7
1.3 Organization of Thesis 8

2 Background 10
2.1 Verification and Validation 10
2.1.1 Inspections 10
2.1.2 Testing 11
2.2 The Benefits of Early and Successful Prediction of Defect Location 12
2.3 Predictive Measurement 14
2.3.1 Defect Analysis 14
2.3.2 Complexity Analysis 16
2.3.2.1 McCabe's Cyclomatic Complexity 17
2.3.2.2 Information Flow Metrics 18
2.3.2.2.1 Fan-In 18
2.3.2.2.2 Fan-Out 19
2.3.2.2.3 Information Flow Metric 20
2.3.2.2.4 Evaluation of the Information Flow Measures 21
2.3.2.2.5 Additional Information Flow Measures Based Upon Fan-Out 21

3 Development of the Experiment 24
3.1 Literature Research 24
3.2 Four Areas of Concern in Software Quality Prediction 25
3.2.1 Internal Complexity 27
3.2.2 Information Flow 27
3.2.3 Changes in Design 27
3.2.4 Defect Categorization 28
3.3 Selection of a Project for Study 28
3.4 Tailoring the Research to the Project 29
3.4.1 The Software Development Methodology 30
3.4.1.1 Analyzing the Methodology 30
3.4.1.2 Software Design Milestones 30
3.4.2 Analysis of Artifacts from Developing Software Product 32
3.4.2.1 Requirements Allocation 33
3.4.2.2 Functional Capabilities 33
3.4.2.3 Preliminary Design 33
3.4.2.4 Detailed Design 34

3.4.2.5 As-Built Design 34
3.4.2.6 Unit Test Plans 35
3.4.2.7 Minutes from Reviews and Walk-throughs 35
3.4.2.8 System Trouble Reports 36

4 Application and Evaluation of Predictive Software Design Measures 37
4.1 The Predictive Model 37
4.1.1 Selection of Candidate Predictive Software Design Measures 38
4.1.2 Milestones and Candidate Predictive Measures 40
4.2 An Overview of the Experiment 43
4.2.1 Preparation 43
4.2.2 Two Step Analysis of Candidate Predictive Design Measures 45

5 Regression Analysis 46
5.1 Analysis Procedure 48
5.2 Selection of Prediction Equations By Design Milestone 50
5.2.1 Requirements Allocation (Milestone 1) 51
5.2.2 Preliminary Design (Milestone 2) 52
5.2.3 Preliminary Design Review (Milestone 3) 53
5.2.4 Detailed Design (Milestone 4) 54
5.2.5 Detailed Design Review (Milestone 5) 56
5.2.6 As-Built Design (Milestone 6) 57
5.2.6.1 As-Built Design Characteristics 57
5.2.6.2 As-Built Design Changes from Detailed Design 59
5.2.7 Unit Test Plans (Milestone 7) 61
5.2.8 Code Walk-Throughs (Milestone 8) 62
5.3 Minimizing Data Collection Effort 63
5.4 Evaluation of Regression Analysis 66

6 Evaluation of Selected Prediction Equations 68
6.1 Means of Evaluation 68
6.2 Evaluation Procedure and Analysis 71
6.2.1 Requirements Allocation (Milestone 1) 71
6.2.2 Preliminary Design (Milestone 2) 73
6.2.3 Preliminary Design Review (Milestone 3) 75
6.2.4 Detailed Design (Milestone 4) 76
6.2.5 Detailed Design Review (Milestone 5) 78
6.2.6 As-Built Design (Milestone 6) 80
6.2.6.1 Results of As-Built Design Characteristics 80
6.2.6.2 Results of As-Built Changes in Design 82
6.2.7 Unit Test Plans (Milestone 7) 84
6.2.8 Code Walk-Throughs (Milestone 8) 85
6.3 Evaluation Results 87
6.4 Minimal Distinct PSDM Prediction Model Results 94

7 Conclusions 100
7.1 Brief Overview of Research and Experimentation 100

7.1.1 Revisiting the Major Goals 101
7.1.2 Revisiting the Minor Goals 103
7.2 Future Research 108

References 110

Appendix A Discussion of Excluded Measures 113
A.1 Function Point Analysis 113
A.2 Halstead's Software Science 115

Appendix B Data Collection for Predictive Software Design Measures 118
B.1 Required Data Collection 118
B.2 Requirements Allocation 120
B.3 Functional Capabilities Section 120
B.4 Preliminary Design 120
B.5 Information Flow Diagrams 121
B.6 Preliminary Design Review Minutes/Mark-Ups 121
B.7 Post-Review Preliminary Design 121
B.8 Detailed Design Documents 122
B.9 Detailed Design Review Minutes/Mark-Ups 123
B.10 As-Built Design Documents 123
B.11 Unit Test Plans 123
B.12 Code Walk-Through Minutes/Mark-Ups 124

Appendix C Candidate Predictive Software Design Measures 125
C.1 Candidate Predictive Software Design Measures 125
C.2 Requirements Allocation Milestone 126
C.3 Preliminary Design 127
C.4 Preliminary Design Review 128
C.5 Detailed Design 129
C.6 Detailed Design Review 131
C.7 As-Built Design 132
C.8 Unit Test Plans 134
C.9 Code Walk-Throughs 135

Appendix D List of Acronyms 136

Appendix E PSDM Values 139
E.1. Requirements Allocation 139
E.2 Preliminary Design 140
E.3. Preliminary Design Review 141
E.4. Detailed Design 143
E.5. Detailed Design Review 146
E.6. As-Built Design 147
E.7. Unit Test Plans 153
E.8. Code Walk-Throughs 155

Appendix F Selected Prediction Equations 157
F.1 Requirements Allocation 157
F.2 Preliminary Design 158
F.3 Preliminary Design Review 158
F.4 Detailed Design 159
F.5 Detailed Design Review 160
F.6 As-Built Design 161
F.7 Unit Test Plans 162
F.8 Code Walk-Throughs 163

Vita 164

List Of Tables

Table 4.1 Summary of Milestones Versus Areas of Concern 41

Table 5.2.1 Candidate Prediction Equations at Requirements Allocation 51
Table 5.2.2 Candidate Prediction Equations at the Preliminary Design Milestone 52
Table 5.2.3 Candidate Prediction Equations at the Preliminary Design Review Milestone 53
Table 5.2.4 Prediction Equations at the Detailed Design Milestone 55
Table 5.2.5 Candidate Prediction Equations at the Detailed Design Review Milestone 56
Table 5.2.6.1 Candidate Prediction Equations for the Design Characteristics at the As-Built Design Milestone 58
Table 5.2.6.2 Prediction Equations for Changes in the Design from the Detailed Design to the As-Built Design Milestone 60
Table 5.2.7 Candidate Prediction Equations at the Unit Test Plan Design Milestone 61
Table 5.2.8 Candidate Prediction Equations at the Code Walk-Through Design Milestone 62
Table 5.3 Minimal Distinct PSDM Prediction Equation Results 65

Table 6.2.1.1 Results of Ranking By Defect Potential for the Requirements Allocation Milestone 72
Table 6.2.1.2 Results of Predicted Defect Counts for the Requirements Allocation Milestone 72
Table 6.2.2.1 Results of Ranking By Defect Potential for the Preliminary Design Milestone 73
Table 6.2.2.2 Results of Predicted Defect Counts for the Preliminary Design Milestone 74
Table 6.2.2.3 DR Prediction Equation Results 74
Table 6.2.3.2 Results of Predicted Defect Counts for the Preliminary Design Review Milestone 75
Table 6.2.3.3 AIARI Prediction Equation Results 76
Table 6.2.4.1 Results of Ranking By Defect Potential for the Detailed Design Milestone 77
Table 6.2.4.2 Results of Predicted Defect Counts for the Detailed Design Milestone 77
Table 6.2.4.3 FO Prediction Equation Results 78
Table 6.2.5.1 Results of Ranking By Defect Potential for the Detailed Design Review Milestone 78
Table 6.2.5.2 Results of Predicted Defect Counts for the Detailed Design Review Milestone 79
Table 6.2.5.2 AIT Prediction Equation Results 80
Table 6.2.6.1.1 Results of Ranking By Defect Potential for As-Built Design Milestone 80

Table 6.2.6.1.2 Results of Predicted Defect Counts for the As-Built Design Milestone 81
Table 6.2.6.1.3 FO and CKP Prediction Equation Results 82
Table 6.2.6.2.1 Results of Ranking By Defect Potential for the As-Built Design Milestone (Changes) 82
Table 6.2.6.2.2 Results of Predicted Defect Counts for the As-Built Design Milestone (Changes) 83
Table 6.2.6.2.3 ADMO Prediction Equation Results 83
Table 6.2.7.1 Results of Ranking By Defect Potential for the Unit Test Plan Milestone 84
Table 6.2.7.2 Results of Predicted Defect Counts for the Unit Test Plan Milestone 84
Table 6.2.7.3 TSTE Prediction Equation Results 85
Table 6.2.8.1 Results of Ranking By Defect Potential for the Code Walk-Through Milestone 86
Table 6.2.8.2 Results of Predicted Defect Counts for the Code Walk-Through Milestone 86
Table 6.2.8.3 AIPC Prediction Equation Results 87
Table 6.4.1 Summary Table of Best Prediction Equations 87
Table 6.4.1 Summary of Minimal Distinct PSDM Prediction Model Results 94
Table 6.4.2 Percent Changes in RSV and MSE in Selecting Minimal Distinct PSDM Prediction Equations Over Best Prediction Equations 95
Table 6.4.3 Percent Changes in USS and CSS in Selecting Minimal Distinct PSDM Prediction Equations Over Best Prediction Equations 98

Table 7.1.2.1 Results of Sizing PSDM Predictions by Design Milestone 105
Table 7.1.2.2 USS and CSS Values for "Best" and "Sizing Only" Prediction Equations 107

Table E.1 PSDM Data at Requirements Allocation 139
Table E.2.1 Information Flow PSDM Values at Preliminary Design 140
Table E.2.2 Internal Complexity and Sizing PSDMs at Preliminary Design 140
Table E.2.3 Change in Design PSDMs at Preliminary Design 141
Table E.3.1 Defect Data PSDMs at the Preliminary Design Review 142
Table E.3.2 PSDMs for Defect Data Normalized by Requirement Count at the Preliminary Design Review 142
Table E.3.3 Internal Complexity and Sizing PSDMs at the Preliminary Design Review 142
Table E.3.4 Change in Design PSDMs at the Preliminary Design Review 143
Table E.4.1 Information Flow PSDMs at the Detailed Design Milestone 144
Table E.4.2 Cyclomatic Complexity PSDMs at the Detailed Design Milestone 144
Table E.4.3 Internal Complexity and Sizing PSDMs at the Detailed Design Milestone 144
Table E.4.4 Change in Design PSDMs at the Detailed Design Milestone 145

Table E.5.1 Defect Data PSDMs at the Detailed Design Review 146
Table E.5.2 Categorized Defect Data PSDMs at the Detailed Design Review 146
Table E.6.1 Information Flow PSDMs at the As-Built Design Milestone 147
Table E.6.2 Cyclomatic Complexity PSDMs at the As-Built Design Milestone 148
Table E.6.3 Internal Complexity PSDMs at the As-Built Design Milestone 148
Table E.6.4 Information Flow Absolute Difference PSDMs 149
Table E.6.5 Cyclomatic Complexity Absolute Difference PSDMs 150
Table E.6.6 Internal Complexity Absolute Difference PSDMs 150
Table E.6.7 Information Flow Percent Change PSDMs 151
Table E.6.8 Cyclomatic Complexity Percent Change PSDMs 151
Table E.6.9 Internal Complexity Percent Change PSDMs 152
Table E.7.1 Unit Test Sizes and Per CI PSDMs 153
Table E.7.2 Per PDL and Per Module Unit Test Data PSDMs 154
Table E.7.3 Per Cyclomatic Complexity Unit Test Data PSDMs 154
Table E.8.1 Action Item Data PSDMs at the Code Walk-Through Milestone 155
Table E.8.2 Normalized Action Item Data PSDMs at the Code Walk-Through Milestone 155

List of Figures

Figure 6.3.1 Prediction USS and CSS Between Design Milestones 89
Figure 6.3.2 Accuracy of Rankings by Milestone 90
Figure 6.4 RUSS Values of Minimal Distinct PSDM and Best Prediction Equations for Each Software Design Milestone 96

Figure 7.1.2 Summary of "Best" and "Sizing Only" RUSS Values 106

1 Introduction

Certain characteristics of software designs have been shown to be predictive of quality and defect potential[1]. The development and verification of metrics measuring these design characteristics is primarily based in the context of accessing the cost and quality of the final product[2]. If this context is changed to improving verification and validation efforts by predicting which areas of the design are prone to defects, the end product's quality improves. This research develops a set of predictive measures which enable testers to identify and target potential problem areas for additional and/or enhanced testing.

1.1 Problem Statement and Related Work

Problem Statement

This research addresses the problem of predicting which areas of a developing software design have the greatest potential for defects (defect proneness). These predictions are formulated through analysis of the artifacts (design and other documentation) produced in the software design process. The software design process includes all phases of development from requirements allocation through the start of coding. The defects detected and documented as a result of formal testing[3] are used in both developing and evaluating predictive software design measures (PSDMs).

---

[1] Please refer to section 2.3 for a brief presentation of predictive metrics.
[2] Cost includes expected effort and scheduling. Specific areas addressed as part of quality are
Reliability and Maintainability.
[3] Please refer to section 2.1.2 for more information regarding formal testing.

Related Work

Various methods and metrics have been developed for predicting the quality of a product from characteristics of its code. Candidate metrics and a theoretical basis for others are readily available in the literature. The problem with most existing metrics is that they target the coded design. This code is not available until the software design process is complete. Artifacts produced in the software design process have received limited examination.

Henry and Selig (1990) present an automated tool for predicting source code quality from a formal Preliminary (or Pseudo) Design Language (PDL). The limiting factor of this tool is that the PDL must be formal. Most designs, including the software design used in this research, use informal PDL. This is especially true in early phases of the design process when less is known about the implementation of the design. This research evaluates other artifacts from the software design process in addition to PDL for use as candidate predictive software design metrics.

Research and experimentation by Shen, Yu, Thebaut, and Paulson (1985) also aims at improving the testing process by identifying defect-prone software. Their study differs from this proposed research in that:

1. Various separate software products are measured and compared.(This research addresses a single software product.)

2. The comparisons are partitioned to address new, modified and translated modules and the programming language used. (The software product in this research is all new, and programmed in a single language.)

3. The majority of testing metric application begins with initial code and extends through the phases of formal testing. (Emphasis of metric

development and application for this research is on software design prior to formal testing.)

1.2 Motivation and Goals

While testing and debugging a software product is typically allocated between a quarter to one half of the development schedule and effort (Sommerville, 1992, pp. 5-10), this initial allocation does not always reflect the actual amounts of time and manpower required. Testing efforts are often caught between the completion of the software design and the contracted release date. As a developing software project nears completion, the time allocated to testing and defect resolution becomes so critical that shortcuts and extended work schedules may become necessary (Grady, 1987, p. 36). Therefore, test plans must be well designed and complete prior to the initiation of actual testing so that no time is wasted.

One way of improving the testing process is to identify, prior to running any tests, areas of a design that contain defects. Obviously this is impossible, but the identification of areas of the design with a high potential for mistakes and defects is possible. Research has shown that certain design and code characteristics can be used both to predict problem areas and to evaluate the likelihood that a software component contains defects. Predictive metrics formed from these characteristics can identify areas for increased testing and/or redesign.

1.2.1. Major Goals

The major goals of this research are reflected in the creation of the set of predictive metrics and their consequent use.

[Creation] The metrics should:

* Enable accurate and early predictions of defect proneness;

* Address all phases of the software design process;

* Provide simple collection and interpretation.

[Deployment] Further, the metrics should constitute a model for:

* Test designers to target defect-prone areas of a software product;

* Designers to identify areas of the design to revisit and possibly redesign.

1.2.1.1 Creation

Enable Accurate and Early Predictions of Defect Proneness

In creating a set of predictive software design metrics, accurate predictions of defect proneness are important. Accurate predictions made early in the design process provide valuable information to both testers and designers. Testers can use the predictions in the development of their test plans. Specifically, identified areas can be targeted for additional testing and inspection. Designers can use the information to revisit and/or redesign areas of the software.

Address All Phases of the Software Design Process

In order to provide early and accurate predictions of defect proneness, the predictive software design measures must address all phases of the development process. There are two reasons for this.

First, the formation of software quality predictions should consider all the materials or artifacts produced. This necessitates addressing each phase of the software design in an orderly manner and evaluating the available data.

Second, while it is important to have early and accurate predictions for the defect proneness of areas in the developing software design, the software is still developing. Changes in the relative rankings of areas of the design by defect proneness are expected. These changes can result from the phases of the software design, requirements changes, and/or revision of the design based upon prior predictions of defect proneness.[4]

Provide Simple Collection and Interpretation

Chillarege et al. (1992) state that "...the area of software quality measurements and quantification is beset with undue complexity and has, in some ways, advanced away from the developer. ... (T)he need to define tractable measurements that are reasonable to undertake and intuitively plausible cannot be understated"[5] (p. 943).

The effort required for collection and interpretation of any metric is a major factor in that metric's acceptance and deployment (Seddio, 1993, p. 231). The quality of collected data can be compromised if the collection process is time consuming or difficult to understand. If the collection and interpretation of metric data is difficult or cumbersome, one can expect the metric collection to be incomplete or abandoned as deadlines approach. Time spent collecting and interpreting metrics is time taken from the project.

---

[4] Preliminary design reviews and detailed design reviews are examples of phases of the software design that can cause changes in the relative rankings.
[5] The author intended "overstated" rather than "understated".

1.2.1.2 Deployment

This research creates a model for metrics collection and evaluation. Model in this case is the definition of a set of predictive measures. This definition includes: specification of the data to be collected and its interpretation, phase(s) of the software design process where the predictive metric is applicable, and the expected value of the metric as a predictor of defect proneness. The model is intended to aid both the test and design efforts by providing accurate predictions of defect-prone areas while addressing all phases of the software design process.

Use by Testers

The research is primarily intended as an aid to the testing process. The benefits are in two areas. The first is to enable test designers to target the areas identified as being prone to defects with additional and/or enhanced testing. The second benefit is that the predictions provide project and test managers an insight into the extent and cost of testing.

Use by Software Designers

The predictive metric model is of interest to software designers because it is designed to provide feedback at each phase of the software design process. Areas of the design identified as being prone to defects can be revisited and/or redesigned.

1.2.2 Minor Goals

There are two minor goals of this research effort:

* Examining the relationship of size to defect proneness.

* Collecting data describing a developing software design.

Examining the Relationship of Size to Defect Proneness

While researching predictive metrics, it became apparent that the value of size, in terms of lines of PDL or code, is debatable. Researchers in software engineering are split on this issue. The count of lines is a commonly used measure and term for normalization because of its simplicity and intuitive appeal. Proponents of Function Point Analysis, such as Albrecht and Gaffney (1983), argue that the quantification of functionality is a better measure of a software design. As a compromise of these conflicting views, both the count of PDL lines and the number of requirements/functions implemented are used in predicting defect-prone areas.[6]

Because sizing measures are commonly used in software development, their performance as predictors of defect counts is of interest. How well do predictive measurements drawn solely from sizing measures (i.e. counts of requirements, lines of PDL, and the number of separate unit tests designed) perform against the "best" prediction equations from this research?

---

[6] Please refer to section 2.3.3.1 for a discussion of Function Point Analysis and its omission from this research.

Collecting Data Describing a Developing Software Design

Another minor goal of this research is to gather data describing a developing software design. This is accomplished through the combination of extensive data collection and analysis of the artifacts produced. The resulting design data and analysis may prove valuable for future research.

1.3 Organization of Thesis

This chapter (Chapter One) introduces the research by presenting the problem and other related research. The goals of this research are also defined.

Chapter Two provides background into software quality, quality prediction, and predictive indicators.

Chapter Three provides a discussion and analysis of the software project used in this research. A skeletal model for the application of predictive design measures is also developed.

Chapter Four defines and discusses the initial candidate predictive design measures and links them to the measurement skeleton developed in Chapter Three.

Chapter Five presents the first phase of the experiment through an overview of the methods, a discussion of the data collection efforts, and presentation of the creation of predictive equations using predictive software design measures calculated from a test group of software functional areas.

Chapter Six contains the validation of the predictive equations selected in Chapter Five. This validation is based upon the performance of the

selected predictive equations when calculated with data from an experimental group of software functional areas.

Chapter Seven is the conclusion. A summary, evaluation of the major and minor goals, and areas of future research are presented.

2 Background

This chapter presents relevant background information and research in the areas of software verification, validation, and predictive measurement.

2.1 Verification and Validation

Verification and validation (V&V) is the process of ensuring that a software product meets all requirements (Plant, 1992, p. 142). V&V also addresses the quality of the design, the performance, and the correctness of all documents produced for the software product.

In the commonly used waterfall model of software development[1], V&V is partitioned into two distinct phases: inspection, and testing. Inspection includes all reviews, walk-throughs, and materials created in the design process. Testing begins with the unit test efforts and ends with acceptance testing.

2.1.1 Inspections

The value of inspecting requirements, design, and code in detecting errors is shown for a variety of software projects and software design methodologies.[2] Inspection takes the form of peer reviews, review meetings, and self inspection. All artifacts (materials and documents) produced in the development (including V&V efforts) are subject to inspection. Defects detected and corrected in the inspection phase of the V&V effort have a lower impact on the product, schedule, and budget than defects detected and corrected in later phases.[3]

---

[1] The waterfall model is used in the software project analyzed in this research. Please refer to Sommerville (1992), pp. 5 -10 for more information about the waterfall model, and section 3.3 of this document for more information about the software project.

[2] Please refer to Kelly, Sherif, and Hops (1992), Beizer (1984), Chillarege et al. (1992), Seddio (1993), and/or Statland (1986) for more information on the benefits of inspections.

[3] Empirical support provided by Kelly et al. (1992) and Seddio (1993).

2.1.2 Testing

Testing is the process of providing inputs to a system and analyzing the resulting outputs against the expected outputs. Testing as a phase of the software life cycle is the verification that the software product satisfies the design requirements prior to release to the user. The goal of testing is to prevent and detect defects (Beizer, 1990, pp. 3-4). Inspections and test case design prevent defects from entering the software product. Those defects in the software product are detected by executing test cases. If the testing efforts are complete and well designed, a better software product is produced.

The testing phase of V&V includes the verification of the units and components of the software product and the validation of the final product. The separate modules of the design are verified via unit testing. Modules are then grouped together into functional components and verified during integration and integration testing. System testing verifies and validates the software product formed from the combination of the separate functional components. Final validation and verification that the product satisfies the customer's requirements is performed by (or in concert with) the customer in acceptance testing (sometimes referred to as Formal Qualification testing).

Formal testing begins after unit testing is completed and the modules have entered configuration control. This is when the code leaves the private domain and enters the public domain of the design team (Beizer, 1984, p. 95-93). Test plans, results, and defect resolution are documented and monitored as part of formal test activities.

2.2 The Benefits of Early and Successful Prediction of Defect Location

The ability to predict defect-prone areas of a software design from artifacts produced in that phase provides the following benefits:

* An improved verification and validation process;
* An aid for software design and development;
* Improvement in end product quality, cost, and schedule.

An Improved Verification and Validation Process

The two phases of verification and validation, inspection and testing, are more effective if accurate predictions of defect-prone areas are available. The benefits result from the identification of defect-prone areas, the time of the predictions, and the traceability of predictions to artifacts produced in the software design process.

Plant (1992, pp.142-143) and Voas and Miller (1993, pp.207-208) state that a goal of inspection and testing efforts is to minimize the effort while maximizing the effectiveness. Identifying defect-prone areas of a developing design allows these areas to be targeted with additional or enhanced review and testing. This increases the likelihood that defects in the targeted areas are exposed.

The earlier defect-prone areas are identified, the earlier inspections and testing can be modified to target them. Test case design begins as soon as the requirements are known (Beizer, 1990, pp. 437-438). Test cases should be designed and tested in a process as rigorous as that used in software design (Beizer, 1990, p. 7). The identification of defect-prone areas early in the design process provides more time for the test case designers to develop and verify test plans.

Predictive measures utilize data from artifacts produced in the software design process to identify defect-prone areas of the design. These identifications are traceable to an artifact or artifacts which can be probed for weaknesses and used in constructing test cases.

An Aid for Software Design and Development

Testers are not the only ones to benefit from the identification of defect-prone areas. Software designers may use predictive metrics to provide in-process feedback identifying areas needing review and/or redesign (Chillarege et al., 1992, pp. 954-955, & Seddio, 1993, p.228).

Improvement in End Product Quality, Cost, and Schedule

The use of predictive metrics to identify defect-prone areas of a developing software design improves the quality of the end product and the management of the project's cost and schedule.

The early identification of defect-prone areas of a software design enables earlier detection and correction of errors. Analysis of these areas for the creation of test cases and/or possible redesign can expose undetected defects. Because defects tend to hide other defects, earlier correction of a given defect means an earlier exposure of additional hidden errors.

The cost[4] of a defect is related to its detection and correction. This cost goes up significantly as a software product matures (Beizer, 1990, p. 27). Late in the software's development, correcting major defects can be catastrophic to the schedule, the design, and the reputations of the software developers.

[4] Cost is in terms of the effort and effect on the schedule.

2.3 Predictive Measurement

Predictive metrics are informative measures derived from software and/or the artifacts produced in the software's development. The intent of collecting these measurements is to provide predictions of the expected quality of a software product from its characteristics. The areas of defect analysis and complexity assessment are the most promising for providing predictive indicators of defect proneness.

2.3.1 Defect Analysis

Defect analysis as a predictor of defect proneness is based upon the assumption that problematic designs or areas of a design can be expected to remain problematic. Reasons for this assumption include the tendency of defects to hide other defects, and the potential introduction of additional defects by any corrections.

Defect Categorization

Defect Categorization is an accounting of defects by counts, locations, types, and/or severities. The resulting data can provide information on identification of problem areas and the status of a developing software project (Chillarege et al, 1992, p. 954).

Defect categorization can be applied on the unit/component level as a predictor of defect proneness:

* Problem areas tend to have large numbers of defects and/or defects of high severity;
* Defects tend to hide other defects[5];
* Correction of defects exposes the design to defect introduction.

At each phase of program development, certain types of defects are more likely to be exposed. For example, the preliminary design review exposes defects related to requirements allocation while defects related to the implementation (PDL) cannot be found until the detailed design phase begins. This characteristic of defect categorization aids in assessing the maturity of a developing software product. If a large number of severe defects related to the understanding and allocation of requirements are present at the detailed design review, the project is not as mature as it should be and the preliminary design may need to be revisited. Other actions to consider are revising the schedule and applying extra effort in the current and future software design and development phases.

Defect Density

Defect density is a metric computed from the data collected in defect categorization. This metric is defined as the number of defects detected in each module or component divided by the size of that module or component (Goodman, 1993, p. 30). The normalization by size addresses the difference in module or component size and its relation to the occurrence of defects[6]. The underlying assumption of defect density is that defects occur at a rate that is

[5] The absence of detected faults may also be used as a basis for investigation. If modules of similar size and/or complexity in a project have logic errors of some severity level, those that have not yet been found to have logic errors should be examined more closely.

[6] Please refer to Basili and Perricone (1984) or Inglls (1985) for a more detailed discussion of defect density.

independent of the individual sizes of the modules. Given the same number of detected defects, a small module has a greater defect density than a large module. Alternate normalizing variables include the number of modules in a component, the number of requirements, and the number of functions to be performed.

2.3.2 Complexity Analysis

Measuring the complexity of a piece of software is a means of quantifying the effort required to gain complete understanding of that software. Modules with high complexity measures tend to have more defects and are more difficult to repair correctly than modules with low complexity measures.[7] Curtis states complexity and sizing can provide guidance to both the testing and the software design efforts by identifying potential problem areas (Perlis & Shaw, 1981, p.204). Complexity measurement also reflects the effect of corrections and modifications. If a module's complexity changes beyond some pre-determined level, a re-review and/or redesign may be required (Seddio, 1993, p. 228).

Areas of Complexity Analysis

Complexity measures quantify the expected difficulty of understanding a software design. This quantification is based upon measurements and evaluations of one or more of the following software characteristics:

* Internal structure
* Information flow
* Size

Internal structure measures quantify the complexity of a piece or pieces of a software design in terms of statements and control structures used. These

[7]Empirical evidence is available in Khoshgoftaar, Munson, Bhattacharya, and Richardson (1992) and Seddio (1993).

measures include counts of GOTO statements, counts of exit points, and McCabe's Cyclomatic Complexity.

Information flow measures address the complexity of the connections between pieces of a software design. Control flow, parameters, and/or global data are used for information flow measures. These measures include fan-in, fan-out, Data complexity, and Structural complexity.

Sizing measures answer the question: "How big is the software?" Examples of sizing measures include the number of requirements, function points, number of modules, lines of pseudo-code, lines of description, and number of comments.

2.3.2.1 McCabe's Cyclomatic Complexity

Cyclomatic Complexity is one of the most widely accepted measures of internal complexity.[8] Curtis contends that Cyclomatic Complexity assesses the difficulty of testing a program because it is a representation of the control paths that must be exercised (Perlis & Shaw, 1981, p. 210). This metric is a count of the number of the number of possible control paths through a program, which can be simplified to:

The number of decision statements + 1

Glenford Myers provides a variant of McCabe's Cyclomatic Complexity (McClure, 1992, p. 45) which recognizes the difficulty of compound condition statements by including Boolean operators in the count.

Evaluation of Cyclomatic Complexity is a simple matter. Higher values mean greater complexity. A general rule in evaluating Cyclomatic Complexity is that the value should not exceed ten paths per module (Goodman, 1993, p. 63).

---

[8]For a complete discussion of Cyclomatic complexity, please refer to McClure (1992), Goodman (1993), Card and Glass (1990), or Grady and Caswell (1987).

Card and Glass (1990, pp.28-29) support using Cyclomatic complexity measures to evaluate the effectiveness of the testing efforts in terms of control paths exercised.

2.3.2.2 Information Flow Metrics

Information flow metrics address system connectivity by observing the flow of information and/or control among system components (Henry & Kafura, 1981, p. 511). They complement the use of structural complexity metrics such as McCabe's Cyclomatic Complexity by addressing the control and data flows within and between modules of a system.

Because the majority of software systems are constructed out of multiple components, the complexity of the system is influenced by the processing performed in each component and how the components interface with each other. This complexity is assessable through analyzing the system's coupling and cohesion relationships. Coupling is the degree of linkage between one component and others in a system. Cohesion is the degree to which a component performs a single function. Components that are highly coupled and lack cohesion tend to be less reliable than those that are loosely coupled and have high cohesion (Goodman, 1993, p. 68).

The complexity of a component is defined to be the sum of the complexities of the modules within the component (Henry & Kafura, 1981, p.514). This is used to calculate the complexity of components and/or the entire software system in terms of fan-in, fan-out, and/or the Information Flow Metric.

2.3.2.2.1 Fan-In (Henry & Kafura, 1981)

Fan-in addresses the potential information and control flow passed into a module. The fan-in for a module A is the number of local flows into A plus the number of data structures read. Local flows for a module A are defined as:

module B calls module A, module A calls module B and uses a value from module B, or module C calls module B and then module A passing a value from B to A (Henry & Kafura, 1981, p. 512). In other words, fan-in is an accounting of calls to, parameters received by, and parameters indirectly received by a given module in a software system.

An enhanced formula for fan-in which considers the count of parameters passed into a module is given by Goodman (1993, p. 70) as:

Enhanced fan-in(A): = Sum of:
- the number of components that call A,
- the number of parameters passed to A (from both higher and lower modules in the module hierarchy), and
- the number of data elements read from by module A.

Fan-in is used to assess the coupling and cohesion of a module. High fan-in values indicate high coupling, low cohesion, and the potential of multiple functions per component (Goodman, 1993, p.69).

2.3.2.2.2 Fan-Out (Henry & Kafura, 1981)

Fan-out addresses the potential information and control flow out of a module. The fan-out for a module A is the number of local flows out of A plus the number of data structures written to. Local flows for a module A are defined as: module A calls module B, module B calls module A and uses a value from module A, or module C calls module A and then module B, passing a value from A to B (Henry & Kafura, 1981, p.512). Fan-out is an accounting of calls from, parameters sent by, and parameters indirectly sent by a given module in a software system.

As in fan-in, an enhanced fan-out formula is given by Goodman (1993, p.70):

Enhanced fan-out(A) = Sum of:
the number of components called by module A,
the number of parameters passed from module A (to modules higher and lower in the module hierarchy), and
the number of data elements written to by module A.

High values of fan-out indicate high coupling, low cohesion, and the potential for missing levels of abstraction in the functional decomposition. Fan-out appears to be a better indicator of a module's complexity and proneness to defects than fan-in (Goodman, 1993, p. 69).

2.3.2.2.3 Information Flow Metric

Information flow measures attempt to address and quantify the strength of a module's communication relationships with other modules in the system (Henry & Kafura, 1981, p.513). The Information Flow Metric is a combination of fan-in and fan-out measures. It is defined by Henry and Kafura (1981, p. 513) as:

$$IF(A) = length(A) * (fan\text{-}in(A) * fan\text{-}out(A))^2$$

where length(A) is the size of module A in terms of lines of code, or lines of pseudo-code.

The Information Flow Metric is weighted by squaring the fan-in times fan-out value because of the belief that the complexity is more than linear in terms of the connections which a procedure has to its environment (Henry and Kafura, 1981, p. 513). Goodman's presentation of the Information Flow Metric drops the Length(A) term from the equation without compromising the metric's effectiveness (Goodman, 1993, p. 67).

2.3.2.2.4 Evaluation of the Information Flow Measures

Henry and Kafura (1981, p.514) present three potential areas of concern in the evaluation of a module or component's complexity with fan-in, fan-out, and/or the Information Flow Metric:

1) When fan-in and fan-out values are high, there are a large number of connections and interactions between the module or component and its environment. This could indicate that the module or component is performing more than one function.

2) High complexity identifies stress points and areas that are difficult to modify correctly due to the large number of potential effects upon the rest of the system.

3) High complexity may reveal inadequate refinement in terms of functional decomposition and module size. Missing levels of functional decomposition and large size modules due to the implementation may result in high complexity.

Goodman (1993, p.68) states that information flow measures are to be used as relative measures between modules and/or components of a system. He also suggests a general rule for evaluating information flow measures: The top twenty-five percent of modules or components in terms of fan-in, fan-out, and/or the Information Flow Metric tend to be problematic and should be examined.

2.3.2.2.5 Additional Information Flow Measures Based Upon Fan-Out

David Card and Robert Glass (1990, pp. 48-50) present two additional uses of the fan-out measurement values in assessing a module or component's structural and data complexity.

Structural Complexity

Because all descendants of a given module are connected to each other by their common parent, there are fan-out squared possible connections between the separate modules to concern the developer. Structural complexity for a component or system A is defined as:

Structural Complexity(A) = Sum(fan-out(i)^2, i= 1..n) / n
where n is the number of individual modules in A.

High values of structural complexity identify those areas of a system in which there exists a greater concentration of module invocations. Low structural complexity values identify those areas of a system in which the module invocations are more evenly distributed.

Data Complexity

Data complexity is a quantification of the expected number of data items to be processed based upon the module's input and output variables and its fan-out. As the functionality of a module is deferred to lower levels via functional decomposition, the complexity of the original module is reduced. This phenomenon is addressed with the data complexity measure. The data complexity of a module is directly dependent upon its own input/output complexity and inversely dependent upon the input/output complexity that it defers to a lower level. For a module A, data complexity is defined as:

Data Complexity(A) = (Count of input/output variables of A) / (fan-out(A) +1).

Modules with high values of data complexity defer less functionality to lower level modules than those modules with low values of data complexity.

2.3.3 Other Common Measures Considered and Not Selected

Albrecht's Function Point analysis and Halstead's Software Science metrics are examples of software size and complexity measures. Albrecht's Function Point Analysis is a method for sizing software projects based upon the number and difficulty of functions they perform. Halstead's Software Science is a collection of formulas for assessing a program or module's size and complexity through operator and operand counts. The methods and formulas defined in Albrecht's Function Point Analysis are not included in this research, but the theoretical basis of Function Point Analysis is addressed. Also, due primarily to the use of informal design PDL, Halstead's Software Science formulas are excluded from this research.[9]

[9]Appendix A presents brief descriptions of Albrecht's Function Point Analysis and Halstead's Software Science accompanied by discussions of their exclusion from this research.

3 Development of the Experiment

To determine which characteristics of a software design identify areas prone to defects, an orderly approach to the problem is required. The steps in this approach are: research of the literature, determining areas of concern, selection of a software project for study, tailoring measurements to the selected project, and defining milestones for applying and evaluating predictive measures.

3.1 Literature Research

Research into related literature highlights complexity and defect analysis as promising areas for defect prediction. Recall from Chapter Two of this document that many potential measures and methods are available for assessing internal complexity, information flow, and defect analysis. Across all types of software projects, no single area of predictive measurement and/or analysis provides definite identification of defect-prone areas. They must be used in combination so as to address the many germane characteristics of a developing software design. In addition to the combination of complexity measures and defect analysis, sizing measures and the magnitude of change provide valuable insight into the potential for defects.

Sizing

Most complexity measures have some dependency on the size of the module or component being measured. If a fixed rate of defects per statement or requirement is assumed, sizing measures are potential predictors of a functional area's defect proneness. An additional use is as a normalizing term for other measures. Sizing measures consist of counts of statements and/or quantification of functionality.

With regards to testing efforts, the size of an area of a software design is potentially related to the amount of testing needed. Areas containing many statements require more complete (and possibly more) tests to ensure the

desired level of statement coverage. Areas implementing many functions and/or requirements may require more test cases than those areas implementing fewer functions or requirements.

Change in Design

Any change in a software design can introduce defects. Seddio (1993, p.230) suggests that any module or component changing in complexity by more than 30% should be revisited and possibly redesigned. The implication is that large changes mean increased proneness to defects. With this research focusing on a developing software design, change is expected as the design matures. The concern is in the relative amount of change. The amount of change or development in a functional area between phases of design should be relatively equal for all functional components. Large changes in a functional area indicate possible problems in prior design phases and/or changing requirements and functionality. Both these factors can cause problems.

The prediction of defect-prone areas requires addressing all relevant design characteristics. This can be accomplished through the use of a combination of complexity measures, defect analysis, sizing, and change measures.

3.2 Four Areas of Concern in Software Quality Prediction

Recall from Section 3.1 that there are five promising areas in the prediction of software quality: internal complexity, information flow, defect analysis, sizing, and change measures. This is reduced to four areas of concern by recognizing that size is readily incorporated into the other measurement areas.

Incorporation of Sizing

Sizing measures are incorporated into the other measurement areas as a measure of change, and as a normalization factor.

As a measure of change, sizing can be used to determine the absolute value of change or the percentage of change in the number of requirements or count of lines of PDL between measurements. Due to the lack of design detail at early stages of development, sizing measures provide predictions of defect proneness through the quantification of functionality. This quantification of functionality is the count of the requirements that a functional area must satisfy. As the design develops, more artifacts are produced, more potential predictive characteristics are available, and sizing measures become attractive as normalization factors.

Normalization by size allows other characteristics of the design to be examined on a 'per LOPC' or 'per requirement' basis. Another reason for using sizing measures as normalization factors is to avoid the loss of predictive information. In developing a small set of predictive software design measures, many characteristics are discarded due to their low value as predictors. When the resulting normalized value is predictive of defect potential, both the original characteristic measure and the normalization factor contribute to the prediction as one variable.

Because sizing measures are incorporated into the other areas of predictive measurement, it is unnecessary to consider them separately. This incorporation results in four areas of concern to be considered in the selection of predictive software design measures. These areas are:

* Internal Complexity
* Information Flow
* Changes in Design
* Defect Categorization

3.2.1 Internal Complexity

Internal complexity is the complexity of the functional area with regards to its internal structure as well as that of the modules and components that comprise it. Internal complexity is the compliment to information flow. The structure and internal complexity of the developing functional area is evaluated through measures such as McCabe's Cyclomatic complexity, counts of GOTOs, counts of loop and extra procedure exits, and counts of looping structures.

3.2.2 Information Flow

Information flow metrics provide a quantification of complexity due to the flow of control and data between modules, components, and functional areas. Measures such as the Information Flow metric, Fan-In, Fan-Out, and the structural and data complexity values presented by Card and Glass (1990) are used to measure information flow complexity.

3.2.3 Changes in Design

The monitoring of the level or amount of change in areas of a developing design aids in identifying problem areas. The comparisons between software design characteristics measured at different points in the design process quantifies the amount of change in a functional area. The assumption is that functional areas experiencing relatively high amounts of change are prone to defects. Levels of change are determined through comparisons of characteristics at two measurement and evaluation points. These levels can be the absolute change and/or percentage of change in a measure.

3.2.4 Defect Categorization

Defect categorization allows testers and designers to determine which functional areas contain the most detected defects. Problematic software tends to remain problematic due to defects hiding other defects and the introduction of defects through correction efforts. Reviews and inspections generate defect reports called Action Items (AIs). Defect categorization uses information contained in these AIs to predict defect-proneness.

3.3 Selection of a Project for Study

For an empirical investigation of predictive software design measures, an actual software project must be studied. The Systems Research Center of Virginia Tech (SRC) is validating the Objective Principles Attributes (OPA) framework.[1] The OPA is validated by applying the framework to the code generated at a defense contractor. The design artifacts produced in the development of this code provide the empirical data for this thesis research.

Project Terminology

The software design is divided up into functional units called Computer System Components (CSCs). These CSCs are divided into Configuration Items (CIs) which perform the various functions and tasks assigned to the CSC. Each CI is made up of one or more modules. Predictions are made on the CSC level based upon data collected by analyzing CSCs, CIs, and modules.

---

[1] Please refer to Nance, Arthur, Love, and Dorsey (1993) for a detailed description of the OPA verification efforts.

Other Contributions by the OPA Validation Effort

In addition to the experimental data, the OPA validation efforts provide the structure for the experimentation of this thesis research. This structure includes both the selection of functional areas of the software design and the partitioning of these areas for the selection and evaluation of predictive software design measures (PSDMs).

The OPA validation efforts concentrate on a group of ten CSCs developed as part of the project. These ten CSCs are selected for study based upon the following criteria:

- Although developed at two geographically and managerially separated sites, the CSCs selected come from a single site.
- They provide a mix of size and functionality.
- The set can be easily split into experimental and test groups.

Partitioning the ten selected CSCs into the test and experimental groups is accomplished by satisfying the following criteria:

- Obtaining two groups with five CSCs each.
- Getting a mixture of large, medium, and small CSCs based upon initial size estimates.
- Avoiding contact with designers responsible for CSCs in the experimental group.

3.4 Tailoring the Research to the Project

Characteristics of the selected software project determine the methodology for the application of predictive measures. These characteristics are the software development methodology used and the design artifacts produced.

3.4.1 The Software Development Methodology

3.4.1.1 Analyzing the Methodology

The software development methodology used in developing the software analyzed in this research defines the common points for measurement and the artifacts produced. The project selected for study uses the common waterfall model of software development.[2] The waterfall model has the following phases of software development:

Requirements Allocation
Preliminary Design
Preliminary Design Review
Detailed Design
Detailed Design Review
Coding
Code Walk-throughs
Unit Testing
Integration and Integration Testing
Systems Integration and Systems Testing
Acceptance Testing (Formal Qualification Testing)

The software design process begins with requirements allocation and ends with the start of coding.

3.4.1.2 Software Design Milestones

The software design milestones of the software design process are those easily definable points where the design pieces are expected to be at the same level of maturity. A given milestone may not occur on the same day for all the functional areas. All the functional areas should be at the same level of maturity and should have produced the same artifacts when compared at the same milestone.

---

[2] Please refer to Sommerville (1992) for a more detailed presentation of the waterfall model.

Why Address Milestones of Software Development?

Software design milestones are used as application points of predictive software design measures for three reasons.

The first reason is that there is no benefit to a "continuous" analysis. Separate functional areas of the software design develop at different rates between design milestones. The milestones are contractually bound points of synchronization for the various functional areas. Any area that has reached a design milestone should be at the same level of maturity as any other areas at that same milestone. Comparisons can only be made between functional areas if they are at the same milestone of development.

The second reason is that design artifacts are produced in reaching or satisfying design milestones. For example, preliminary designs must be complete and documented at the preliminary design review. This is also true for detailed designs and the detailed design review.

The third reason relates to practical considerations. Calculating predictive software measures at design milestones addresses all major phases of the software design process. The data collection and calculation is infrequent enough so as to have a low impact on effort and schedule.

Defining the Milestones

Completion of a phase in the software design process is considered a milestone for the analysis of artifacts and the calculation of predictive software design measures. This means that the milestones for the application of predictive software design measures can now be defined from the phases of the software design process:

Milestone	Software Design Phase
Requirements allocation	Completion of requirements allocation and functional capabilities.
Preliminary design	Completion of preliminary design.
Preliminary design review	Completion of preliminary design review and any modifications to the preliminary design caused by the review.
Detailed design	Completed detailed design prior to detailed design review.
Detailed design review	Completion of detailed design review.
As-built design	Completion of any changes to design caused by detailed design review and/or implementation (coding).
Unit test plan completion	Completion of design of unit test plans.
Code walk-through	Completion of code walk-through.

3.4.2 Analysis of Artifacts from Developing Software Product

With the milestones for the application of predictive software design measures defined, the artifacts produced in the development of the software design can be addressed. At a given milestone, all available artifacts are subject to evaluation for use in predictive measures. The artifacts analyzed are all found in the designer-maintained software development folder (SDF). There is an SDF for each functional area or computer system component (CSC) in the software design. Each SDF contains sections documenting:

Requirements allocation.
Functional capabilities.
Preliminary design (pre and post review).
Detailed design (prior to reviews).
As-built design (post review detailed design).
Unit test plans.
Minutes from reviews and walk-throughs.
- Preliminary design
- Detailed design
- Code walk-throughs

System trouble reports.

Please refer to Appendix B for a detailed presentation of the data collected from each design artifact.

3.4.2.1 Requirements Allocation

This section of the SDF contains the initial allocation of requirements to the functional area or CSC. This allocation includes a preliminary listing of derived requirements and constraints. Derived requirements are requirements created as a result of the allocation of requirements and functional decomposition. Examples of derived requirements include controlling CSC access to global data and providing utilities for use by other CSCs. Constraints are limitations placed upon the CSC due to its environment and include the size of message and print queues, available memory, and the limitations of other software components.

3.4.2.2 Functional Capabilities

The functional capabilities section of the SDF is an overview or summary of the requirements, and the operation of the CSC. Each functional capability of the CSC is described in a sentence or two.

3.4.2.3 Preliminary Design (Pre- and Post-Review)

The preliminary design demonstrates the designer's understanding of the requirements and includes the requirements allocation (with modifications in some cases), an explanation of the required execution, and a high level diagram of the CSC and its interfaces to other CSCs.

The modifications to the requirements are primarily due to a better understanding of the design but may be due to mistakes in the allocation or

understanding of requirements. This is especially true for changes made in the post-review preliminary design.

The explanation of the execution is a brief statement of how the CSC addresses the requirements and how it interacts with the rest of the system. The brevity of this description reflects the immaturity of the design.

High level diagrams of the CSC and its interfaces show that the designers have addressed how the CSC functions as a part of the overall system. The diagrams define the CSCs interacting with a given CSC, the control flows between them, and any data exchanged.

3.4.2.4 Detailed Design (Prior to Reviews)

The software design undergoes a major increase in maturity as it passes from preliminary design to detailed design. Recall that preliminary design is an allocation of requirements, high level definition of interfaces, and a brief description of execution. At detailed design, a CSC is decomposed into its functional components or Configuration Items (CIs). CIs are decomposed into sub-functions or modules. Interfaces and control flow between CSCs, between a CSC's CIs, and within each CI are defined in detail. In addition, the processing for each module, CI, and CSC is defined, designed, and written in PDL. At the completion of detailed design, all intricacies of representing the design as PDL should be addressed. Further changes are expected due to reviews and implementation.

3.4.2.5 As-Built Design (Post-Review Detailed Design)

The as-built design represents how the software is coded. It includes any and all changes and corrections resulting from the detailed design review. As-built designs also reflect changes made to the design due to implementation factors. These factors include interfacing with the environment, language specific

constructs and processing, and changes caused by the further development of other CSCs. After the detailed design review, the design is revisited as a part of code walk-throughs. The large amount of change in the design as it passes from detailed design to as-built is a cause for concern.

3.4.2.6 Unit Test Plans

Unit test plans are designed prior to code walk-throughs. They are intended to exercise and verify the functionality assigned to the CSC. These test plans are not unit tests in the 'classical' sense. Beizer (1990, p.21) defines unit testing as the process of showing that a unit does not satisfy its functional specification and/or does not match its intended design structure. The unit testing efforts for the project under study are better characterized as component testing because they attempt to verify the CSC in terms of its functionality. This functionality is provided by the individual modules or units and the interactions between them. The modules and their internal structure are not directly addressed.

3.4.2.7 Minutes from Reviews and Walk-throughs

Preliminary Design
Detailed Design
Code Walk-throughs

Minutes from reviews and walk-throughs document the inspection of the developing design at the preliminary design review, the detailed design review, and the code walk-throughs. Minutes include all Action Items (AIs) assigned during the review, listings of attendees, and date(s) of the meeting(s). AIs define the responsible person(s) and the required action. This action ranges from simple markups to investigation and re-design. In some cases, AIs are not separately documented but are included as notes and markups to the design. Preliminary design and detailed design reviews inspect the preliminary design

and detailed design respectively. Code walk-throughs review not only the code but also the as-built design and unit test plans.

3.4.2.8 System Trouble Reports

Formal Testing is recorded through the use of System Trouble Reports (STRs) which document the problem and the CIs affected, where CIs are Configuration Items. STRs also specify the severity of the problem, the estimated time to fix, the actual time spent on each affected CSC or CI, and a record of the STRs life-cycle. This life-cycle encompasses the actions taken to address the STR. STR actions are: opening, assigning, correcting, and closing. From this information, each CSC is allocated a defect for each identification of the CSC or a component CI as affected in an STR. For example, if an STR affects five CIs and four of those CIs belong to CSC 'A', then CSC 'A' is assigned four defects. The defect count is defined as the total number of defects assigned to a CSC and is used in the creation and verification of a predictive model.

4 Application and Evaluation of Predictive Software Design Measures

In order to evaluate any predictive software design measures (PSDMs) it is necessary to determine what PSDMs are to be collected and when this collection is to take place. The selection of PSDMs and their time of collection is based upon software design milestones, areas of concern in predictive measurement, and the availability of design artifacts.

With a model for the application of PSDMs defined, prediction equations are created using design data from the test group of CSCs. Evaluation of these prediction equations is based upon their performance in predicting the defect potential of CSCs in the experimental group.

4.1 The Predictive Model

To develop a model for the application of predictive software design measures, three aspects must be defined. The first aspect of the model is determining when to make predictions of defect proneness. The second aspect is determining the design artifacts that are available at each prediction point. The final aspect to be defined is determining which areas of predictive software measurement to address.

Points for Predicting Defect Proneness

In Section 3.4.1.2 of Chapter Three, eight design milestones for the measurement and prediction of defect proneness are defined. These definitions arise directly from the software development methodology used in the software project being researched. The eight design milestones are (in order by time):

1) Requirements allocation
2) Preliminary design
3) Preliminary design review
4) Detailed design
5) Detailed design review
6) As-built design
7) Unit test plan completion
8) Code walk-through

Available Design Artifacts

Developing a model for the application of predictive design measures requires that the design artifacts available at each milestone be known. Section 3.4.2 of Chapter Three presents the design artifacts and their point of availability within the development process.

Predictive Software Measurement Areas

In Section 3.2 of Chapter Three, four areas of concern for software quality prediction are presented. These areas provide coverage of software design characteristics shown to be valuable in predicting defect proneness. The areas of concern are:

* Internal Complexity
* Information Flow
* Defect Categorization
* Change in Design Measures

4.1.1 Selection of Candidate Predictive Software Design Measures

With the structure of the predictive model defined, candidate measures are selected. At each design milestone, a candidate set of PSDMs is drawn from the areas of concern in predictive design measurement. These candidate predictive

measures are judged by selection criteria. Measures not satisfying the criteria are eliminated. These selection criteria are:

* Applicability to the project under consideration
* Ease of collection, calculation, and interpretation
* Accuracy of prediction

Applicability to the Project Under Consideration

Predictive software design measures selected as candidates in this research must be applicable to the project and available design artifacts. Each milestone in the software design process has an associated set of design artifacts. Predictive measures are selected if they can be calculated from the available data. Some predictive measures such as Fan-In, Fan-Out, and McCabe's Cyclomatic Complexity cannot be calculated until the detailed design is complete. This means these measures cannot be selected for use at any milestone before detailed design.

Ease of Collection, Calculation, and Interpretation

A metric that is cumbersome or difficult to collect, calculate, and/or interpret is unattractive to software developers. The amount of effort, the understandability, and the consistency of the interpretation are major factors in the acceptance of a software metric. Effort expended on metrics collection and calculation is effort taken from the rest of the project. As deadlines approach, the temptation to abandon metric collection increases. If the metrics and their intended benefits are not understood by those responsible for their collection and use, the quality and consistency of the data and interpretation suffers.

In the software project analyzed for this research, the design review and walk-through forms provide examples of the problems associated with the collection of metrics. These review forms have fields for entering the amount of

time spent in the review, and the complexity of the component. When the forms are filled out, time and complexity values are not provided, provided inconsistently, or provided as constant values on photocopied forms. Because the initial collection of time and complexity measures is incomplete and of poor quality, the effort expended in collection, as well as any subsequent attempts at interpreting the data, is wasted . Metrics relying on these values for time or complexity must be abandoned or recollected.

Accuracy of Prediction

Any selected measure of defect proneness must provide an accurate prediction. One would be hard pressed to convince a developer to use PSMDs that do not correctly predict defect-proneness. Candidate predictive measures in this research are rejected if they do not accurately identify defect prone areas of the design.

4.1.2 Milestones and Candidate Predictive Measures

The intent in developing and selecting predictive software design measures (PSDMs) is to address the areas of concern in predictive measurement at each design milestone. Due to the availability of design artifacts and the maturity of the developing design, some areas of concern are not applicable at a given design milestone. The following figure summarizes the areas of concern in predictive measurement that can be addressed at each design milestone.[1]

[1] For a more detailed presentation of the selection of predictive software design measures by area of concern for each milestone, please refer to Appendix C.

Table 4.1 Summary of Milestones Versus Areas of Concern

Milestone	Internal Complexity	Information Flow	Defect Categorization	Change in Design Measures
Requirements Allocation	No	No	No	Yes (via initial allocation of requirements)
Preliminary Design	No	No	No	Yes
Preliminary Design Review	No	Yes	Yes	Yes
Detailed Design	Yes	Yes	No	Yes
Detailed Design Review	No	No	Yes	No (deferred to as-built design)
As-Built Design	Yes	Yes	No	Yes
Unit Test Plans	Yes	Yes	No	No
Code Walk-Through	No	No	Yes	No

Requirements Allocation

The allocation of requirements to a functional area is defined as measures of the amount of change between measurement milestones prior to detailed design. The amount of change in the developing design is quantified by the number of requirements and the difference in the number of requirements between design milestones. The requirements counts are related to the internal

complexity and information flow measures of the detailed and as-built designs. Before detailed design, the functional areas of the design have not received any functional decomposition. Until this functional decomposition occurs, no reliable quantification of internal complexity or information flow can be made from requirements counts. These counts are treated as design changes.

Changes in Design Between Preliminary Design and Detailed Design

Changes in the design at the detailed design milestone include a comparison between the number of requirements in the post-review preliminary design and the number of modules and CIs designed. This comparison is based on the fact that the functional areas defined in the preliminary design are decomposed into CIs which are also functionally decomposed into modules. For a given functional area, the number of requirements is related to the number of CIs and modules.

Early Information Flow Measures

Information flow can be measured at preliminary design review by analyzing pictorial representations of control and data flows amongst functional areas of the design. These measures are excluded from the preliminary design review milestone because the design artifacts do not document any significant changes to the pictorial representations arising from the preliminary design review.

Changes in Design from Detailed Design to As-Built Design

The change in design between the detailed and as-built designs includes markups due to investigations and re-designs caused by the detailed design review. The lack of consistently available documentation reflecting only changes caused by the detailed design review forces comparisons to be deferred until the

as-built design. The comparisons are based upon changes due to the detailed design review, further maturation of the design, and problems encountered in coding.

4.2 An Overview of the Experiment

Recall from Section 3.3 of Chapter 3 that the functional areas (CSCs) of the selected software project are separated into a test group and an experimental group. The test group is used to select predictive measures and create prediction equations through a regression analysis. The experimental group is used to evaluate the selected prediction equations.

4.2.1 Preparation

There are two steps in preparation for the selection and evaluation of predictive software design measures (PSDMs) with an actual software project. The first step is to develop a model for the application of predictive measures. Using this model, the second step is to collect the necessary data from the design artifacts.

Developing a Model for the Application of Predictive Measures

Section 4.1 of this document discusses the development of a model for the application of predictive software design measures. There are two important characteristics of the predictive model. The first is that the model provides predictions at established points in the software design process. These points are called milestones. The various functional areas of the developing design are comparable at the milestones because they should be at the same maturity level. The second important characteristic of the predictive model is that it addresses four promising areas of concern in the prediction of defect potential. These areas

of concern are: internal complexity, information flow, defect categorization, and changes in design.

Collection of Data

There are two important points in the collection of data from the design artifacts. First, all collection is performed by a single person. This minimizes the amount of variation in collected values due to differing interpretations of data collection requirements and definitions. The second important point is that all functional areas (CSCs) of the software design are collected during the same time period. This minimizes differences in data collection among the CSCs and between the test and experimental groups.

Effort and Time

Collection of the data used in the selection phase (regression analysis) and evaluation phase (simulation) averages two to three eight hour days per functional group (CSC). This estimation assumes that the necessary design artifacts are available. Once predictive equations are selected for each milestone, the average data collection time for each CSC is roughly an eight hour day. Investing a day's effort into data collection for each CSC may seem a high price to pay, but the time is spread over the entire software design process.

Problems With Information Flow Metrics

The information flow metrics are incomplete and defined as "limited" due to the fact that the CSCs analyzed in this research are only a subset of those in the software design. Consequently, some of the collected data is incomplete. All the information flow metrics are affected by this limited data in some way.

4.2.2 Two Step Analysis of Candidate Predictive Design Measures

With the predictive model defined and the necessary data collected and calculated, predictive software design measures (PSDMs) are selected and evaluated.

Defect data collected during integration testing and Formal Qualification Testing (FQT) is used to compile the actual defect counts in the selection and evaluation of predictive equations and PSMDs. This defect data is extracted from System Trouble Reports (STRs) and assigned to all affected CSCs. A given STR may affect multiple Configuration Items (CIs) in a CSC. Each identified CI is added to the total defects for its parent CSC. In this manner, counts of defects for each CSC are collected and provide the means for comparison of software quality. For the purposes of this research, software quality is reflected by the number of defects: the more defects, the lower the quality of the CSC.

Selection of PSDMs and prediction equations is performed through the creation of multiple regression equations in which more than one distinct predictor (or PSDM) can be used to predict a single dependent value (i.e. number of defects) (Milton & Arnold, 1986, p. 393). Considering the design milestones separately, multiple prediction equations are created and selected from combinations of the potential PSDMs. The test group of CSCs provides the necessary data. Chapter Five of this document presents the results of the first phase of the experiment.

The experimental group of CSC data is used to validate the selected predictive equations from the first phase of the experiment against the associated defect data. These equations are used to predict the number of defects for the five CSCs in the experimental group. A single "best" predictor equation is selected at each design milestone based upon the results of the predictions for the experimental group. Chapter Six of this document presents the results of the second phase of the experiment.

5 Regression Analysis

This chapter documents the creation and selection of prediction equations for each milestone of the software development using predictive software design measures (PSDMs) calculated from the test group of CSCs. Small sets of "select" prediction equations are chosen at each milestone for validation which is included in Chapter Six of this document.

Multiple Regression

Multiple regression analysis uses one or more independent variables to explain or calculate the value of a single dependent variable. In this research, the dependent variable is the number of defects discovered in each CSC and the independent variables are chosen from the PSDMs. Predictive equations are in the form of weighted sums of PSDMs. This means that the values of selected PSDMs are weighted (multiplied) by coefficients and added to produce predicted defect counts.[1] Note that these coefficients can have negative values.

Recall from Chapter Four (4.2) that the test group of CSCs and their associated defect counts are used in the formation of prediction equations through multiple regression. The test group of CSCs provide the PSDM data used as the independent variables in the creation, evaluation, and selection prediction equations at each software design milestone.

Counts of the defects, the dependent variable, are collected from the results of formal testing efforts. These efforts are integration testing and the Formal Qualification Testing (FQT). System Trouble Reports (STRs) document defects detected in formal testing. For each instance that a given CSC is documented as being affected, it is assigned a defect. A CSC may be identified as being affected multiple times in a single STR. Each of these identifications constitutes a defect.

---

[1] Please refer to Appendix E for the PSDM data used to create prediction equations and Appendix F for the coefficients and PSDMs of the selected prediction equations.

Goal of Regression Analysis

The goal of the regression analysis is to produce accurate prediction equations. These equations should also use a minimal number of PSDMs because each PSDM requires a significant amount effort to collect.

Two important concerns govern accuracy of the predictive equations. First, it is important that the predictive equations capture the variability in the relationship between the number of defects and the selected PSDM(s). The proportion or degree of variation in the dependent variable, which is explained by the selected independent variables, is evaluated with the coefficient of determination: R-squared (Younger, 1979, p.234).

The second aspect of accuracy assessment is that the predictive equations must provide accurate estimates of the number of defects discovered. The estimates can be used to rank CSCs by defect potential (i.e. the number of defects to expect) for targeting with tests and/or redesign. These rankings provide general identification of defect-prone areas. The accuracy of the predicted rankings should be greater than the accuracy of the predicted defect counts when compared to the actual rankings and defect counts respectively. When testers want to know which CSCs are the most prone to defects, the accuracy of the rankings by defect potential is more important than the accuracy of the predicted defect counts. Because predicted rankings by defect potential are ordered by the predicted defect counts, the concern in the selection of prediction equations is that the predicted defect counts are as close to the actual defect counts as possible. Exact predictions of the number of defects cannot be expected, but the amount of error (mean sum of squared error) should be kept to a minimum.

5.1 Analysis Procedure

For each design milestone, a large set of potential predictive equations can be generated using a commercially available software package.[2] Many equations are eliminated from consideration because they contain PSDMs that are in conflict. This elimination is based on the formation of logical relationships between the PSDMs.[3] The logical relationships determine those combinations of PSDMs that are acceptable for use in potential predictive equations. Consider the count of parameters as an example. Total parameters passed is defined as the sum of the number of parameters passed into and out of a CSC. A predictive equation cannot use the total parameter count in combination with either the parameter in or parameter out counts since an obvious dependency exits. By avoiding this use of highly related data, additional PSDMs can be considered.

Since the PSDMs are created and selected to address the four areas of concern in predictive measurement[4], combinations should address as many of these areas as possible. Predictive equations consisting of PSDMs from a single measurement area are rejected in favor of equations addressing more areas of concern. For example, a predictive equation using only PSDMs addressing information flow measures is rejected in favor of a predictive equation with PSDMs from multiple areas of concern.

With the set of potential predictive equations reduced to those without data conflicts and addressing as many areas of concern as possible, selections of equations is performed at each milestone. Three variables guide the selection of predictive equations:

---

[2] The SAS statistical software package is used for the creation of predictive equations through multiple regression. Specifically, the REG procedure is run in the R-squared mode to find the best predictive equations using from one to as many as four of the PSDMs.

[3] Please refer to Appendix C for a presentation of the PSDM logical relationships for each design milestone.

[4] Please refer to Chapter Four.

1) R-squared value (RSV),
2) Mean Sum of Squared Error (MSE), and
3) Number of PSDMs included.

R-Squared Value (RSV)

The R-squared value (RSV) is the amount of variation in the dependent variable that is explained by the predictive equation (Milton & Arnold, 1986, p. 421). Good equations have an RSV close to 1 while poor equations have RSVs close to 0. The interpretation of RSV is subjective. An RSV of 0.33 indicates a weak relationship between the actual and the predicted defect counts. Alternatively, this value indicates that the prediction equation explains 33% of all the variablility in the actual defect counts, while 66% of the remaining variability is explained by all other possible predictors taken together (Younger, 1979, p. 8). Comparing two prediction equations from this research effort, the one with the larger RSV is said to explain a greater portion of the variability of the actual defect count.[5]

Mean Sum of Squared Error (MSE)

The mean sum of squared error is the sum of the squared differences between the actual values (number of defects) and the predicted values divided by the degrees of freedom (Younger, 1979, p. 216). Computation of degrees of freedom is based on the number of sample points used in the regression minus the number of terms in the regression equation.[6] The MSE value is an indication of how close the predicted values are to the actual values for the data used in the generation of the prediction equation. Low values mean that the predicted values are close to the actual values.

---

[5] This assumes the prediction equations are both linear with a zero intercept and that the same formula for RSV is used. Please refer to Kvalseth (1985) for more information.

[6] For a sample size of 5, a regression equation using 2 independent variables and no intercept term, the degrees of freedom is 3.

RSV Versus MSE

When considering the changes to RSV and MSE between prediction equations of one or more PSDMs, an improvement in RSV does not guarantee an improvement in MSE. Adding independent variables cannot lower the RSV and usually improves it by explaining more of the variation. It is possible to increase the RSV value by adding independent variables while the MSE increases. The reason for this curious result is that the RSV is based upon the amount of the variability in the dependent variable explained by the independent variables. In improving the explanation of the variability across the sample data set, the sum of squared error is improved via the least squares line fitting, but the number of degrees of freedom is decreased with the addition of an independent variable.

Number of PSDMs Required

Each PSDM requires a significant amount of collection effort. Some PSDMs are simple to collect (e.g. the number of CIs) while others are more difficult (e.g. Information Flow Metric).[7] The use of equations with a few simple PSDMs is preferable to equations containing many or difficult PSDMs if the resulting RSV and MSE are comparable.

5.2 Selection of Prediction Equations By Design Milestone

Data generated by the test group of CSCs are used to create predictive equations that are then examined with the goal of selecting one or more with a high RSV, low MSE, and a minimal number of PSDMs. These selection factors are considered in combination rather than separately. The addition of a PSDM

[7] For the purposes of this research, PSDM difficulty of collection is defined as the number of individual counts or measurements required.

might improve the RSV a small amount but cause a significant improvement in the MSE. The decision to be made here is if the MSE improvement is worth the cost of collecting an additional PSDM. If the decision is not clear, both prediction equations are selected. Each milestone produces at least one prediction equation.

Acronyms are used for the labeling of the PSDMs. A complete list of all acronyms is available in Appendix D. The following sections present the prediction equations selected at each design milestone.

5.2.1 Requirements Allocation (Milestone 1)

Table 5.2.1 presents the best of the prediction equations created at the first design milestone.

Table 5.2.1 Candidate Prediction Equations at Requirements Allocation

Equation Number	PSDMs Used	RSV	MSE
1	**TR**	**0.745**	**2197.9**
2	**FC**	**0.742**	**2217.6**
3	**AR**	**0.726**	**2356.5**
4	DR	0.655	2967.6
5	TR FC	0.755	2809.9
6	DR FC	0.750	2862.7
7	AR DR	0.744	2925.2
8	AR CO	0.742	2955.8
9	AR FC	0.742	2955.9
10	DR CO	0.670	3786.5
11	AR CO FC	0.771	3944.7
12	DR CO FC	0.766	4025.1
13	AR DR FC	0.751	4285.7
14	AR DR CO	0.748	4341.5
15	AR DR CO FC	0.781	7544.8

Note: Selected equations and best values are in boldface.

Selection of Prediction Equations at Milestone 1

The best single-PSDM equations use TR, FC, or AR (shown in bold type). TR is selected as the best prediction equation, with FC, and AR as alternates.

The alternates are selected because they are simpler to collect and result in only a small decrease in expected accuracy (i.e. a small reduction in RSV and a small increase in MSE).

Switching to a prediction equation with two or more PSDMs increases the R-squared value (RSV) only slightly while significantly increasing the mean sum of squared error (MSE). Due to these factors and the effort required to collect additional PSDMs, only the single PSDM equations are selected.

5.2.2 Preliminary Design (Milestone 2)

Table 5.2.2 presents the best prediction equations created at the preliminary design milestone.

Table 5.2.2 Candidate Prediction Equations at the Preliminary Design Milestone

Equation Number	PSDMs Used	RSV	MSE
1	**DR**	**0.845**	**1332.7**
2	TR	0.792	1785.8
3	AR	0.777	1915.1
4	NOD	0.679	2765.7
5	**PTO DR**	**0.942**	**667.3**
6	PTO NOD	0.927	840.5
7	DR CO	0.926	851.6
8	DR ADR	0.925	864.2
9	DR PCR	0.924	866.1
10	**PTO DR PCR**	**0.999**	**10.9**
11	**PTO DR ADR**	**0.999**	**11.6**
12	PTO PFR NOD	0.988	203.8
13	PTO TR DES	0.985	251.7
14	PTO PFR PCR NOD	1.000	0.0
15	PTO PFR ADR NOD	1.000	0.0
16	PTO AR PCR DES	1.000	0.0
17	PTO PFR TR DES	1.000	0.1

Note: Selected equations and best values are in boldface.

Selection of Prediction Equations at Milestone 2

The initial prediction equation selected uses only the derived requirements (DR) PSDM. Adding Parameters To (PTO) to form a two-PSDM equation, increases the RSV and decreases the MSE. The equations using three PSDMs have good results so the two best equations are selected (prediction equations 10 and 11). The four-PSDM prediction equations only offer a minor improvement over the three-PSDM equations while adding an additional metric to calculate. No four-PSDM equations are selected.

5.2.3 Preliminary Design Review (Milestone 3)

Table 5.2.3 presents the best prediction equations created at the preliminary design milestone.

Table 5.2.3 Candidate Prediction Equations at the Preliminary Design Review Milestone

Equation Number	PSDMs Used	RSV	MSE
1	**AIARI**	**0.923**	**659.9**
2	ARIPR	0.904	823.2
3	TR	0.902	843.6
4	DR	0.895	902.3
5	DR, DA23TR	0.996	49.4
6	DR, DPC23TR	0.995	56.2
7	**ARIPR, DPC23DE**	**0.992**	**87.8**
8	**AIARI, DA23TR**	**0.992**	**89.9**
9	AIARI, DPC23TR	0.989	124.4
10	TAI AIARI CO	1.000	1.0
11	**TAPR AIR1 DPC23DE**	**1.000**	**5.4**
12	**ARIPR AR DPC23DE**	**0.999**	**13.1**
13	**ARIPR DA23TR DPC23DE**	**0.999**	**15.8**
14	ARIPR DPC23TR DPC23DE	0.999	17.0
15	DR DE DA23TR	0.999	17.0
16	TAPR ARIPR AIR1 DR	1.000	0.0
17	ARIPR TAI DR CO	1.000	0.0
18	TAI AIARI CO DA23DE	1.000	0.0
19	TAPR AIR1 DPC23TR DA23DE	1.000	0.0
20	AIARI AR DPC23TR DA23DE	1.000	0.0
21	A1PR AIND CO DE	1.000	0.0
22	AIND TR CO DPC23TR	1.000	0.0
23	A1PR TAI TR DPC23DE	1.000	0.0

Note: Selected equations and best values are in boldface.

Selection of Prediction Equations at Milestone 3

The initial single PSDM prediction equation selected uses Action Items requiring redesign and/or investigation (AIARI).

Considering the two-PSDM equations, equations 5 and 6 are rejected because they both only address the requirements. It is desirable to use information from both the Action Items (defect categorization) and changes in design (requirements & requirements changes) so equations 7 and 8 are selected without much loss in RSV or MSE.

The three-PSDM equations are selected with reservations because they use a variety of PSDMs. They also improve both the RSV and MSE, but at the cost of calculating and collecting three PSDMs.

5.2.4 Detailed Design (Milestone 4)

Table 5.2.4 presents the best prediction equations created from PSDMs available at the detailed design milestone.

Table 5.2.4 Prediction Equations at the Detailed Design Milestone

Equation Number	PSDMs Used	RSV	MSE
1	**FO**	**0.969**	**266.5**
2	**C**	**0.956**	**363.0**
3	PDL	0.939	528.4
4	MO	0.934	571.5
5	CI	0.924	657.7
6	**CI MOPRE**	**0.992**	**84.1**
7	**FI C**	**0.991**	**103.2**
8	PM FO	0.989	126.4
9	DC C	0.988	132.4
10	FO UP	0.986	161.3
11	**IFP EX C**	**0.999**	**9.2**
12	**PDLPPRE CI CMO**	**0.999**	**10.8**
13	SC FO CGT	0.999	11.0
14	**PDLPPRE IF EX**	**0.999**	**13.1**
15	DC SC EX	0.999	13.5
16	MO IF EX	0.999	13.7
17	PDL DC FI CCI	1.000	0.0
18	CI DC FI CGT	1.000	0.0
19	PM UP CMO CGT	1.000	0.0
20	DC SC EX CMO	1.000	0.0
21	SC FI EX CGT	1.000	0.0
22	PM DC IF EX	1.000	0.0
23	MO IFP UP CGT	1.000	0.0
24	FO IF EX CCI	1.000	0.0
25	PDLPPRE CI MO CGT	1.000	0.0
26	FO IF EX CMO	1.000	0.0

Note: Selected equations and best values are in boldface.

Selection of Prediction Equations at Milestone 4

The initial single-PSDM selections have a relatively high value of R-squared (RSV) and a low mean sum of squared error (MSE). Selecting the two-PSDM equations 6 and 7 allows two areas of concern in predictive measurement to be addressed. In equation 6, internal complexity and change in design are addressed. The areas addressed in equation 7 are information flow and internal complexity.

Three-PSDM prediction equations provide additional improvement and address more areas of the developing design. Equation 11 is selected but

requires calculating Fan-In, Fan-Out, lines of PDL, exception handlers, exceptions raised, and Cyclomatic Complexity. Equation 12 is less difficult to collect than equation 11 so it is also selected.[8] Equation 14 is selected because it is similar to equation 11 in accuracy, and includes the design change measure Lines of PDL per post-review preliminary design.

The improvement in the RSV and MSE of four-PSDM equations does not justify the increased data collection.

5.2.5 Detailed Design Review (Milestone 5)

Table 5.2.5 presents the best of the prediction equations created from PSDMs available at the detailed design review.

Table 5.2.5 Candidate Prediction Equations at the Detailed Design Review Milestone

Equation Number	PSDMs Used	RSV	MSE
1	**AIT**	**0.972**	**240.7**
2	APP	0.948	448.5
3	APC	0.941	511.0
4	AIARI	0.939	521.3
5	**APC TNDA**	**0.989**	**128.9**
6	**AIT TNDAPC**	**0.988**	**132.1**
7	AIT TNDA	0.988	140.7
8	AIT TNDAPP	0.984	179.8
9	APC TNDAPC	0.980	231.0
10	**APC ARIPP TNDA**	**0.997**	**47.3**
11	**APP ARIPP TNDA**	**0.996**	**68.6**
12	APC AIARIPC TNDA	0.995	91.3
13	AIT AIARI TNDA	0.992	142.5

Note: Selected equations and best values are in boldface.

Selection of Prediction Equations at Milestone 5

The best single-PSDM prediction equation is total Action Items (AIT).

[8] I.e. lines of PDL, requirement count from the post-review preliminary design, the number of CIs, the Cyclomatic Complexity, and the number of modules.

The two-PSDM equations are selected because they improve both the R-squared value (RSV) and mean sum of squared error (MSE). These prediction equations also utilize the CI count sizing measure and include more information from the defect categorization data. Specifically, non-designer action item counts are included.

Two of the three-PSDM prediction equations are selected because both the RSV and MSE improve. These equations also address more of the design characteristics (i.e. LOPDL) and more of the defect categorization (i.e. re-design/investigate Action Items). The three-PSDM prediction equations are selected with reservation because the added coverage of design characteristics is costly. Examining the RSV and MSE shows there is significant improvement which justifies the additional data collection.

5.2.6 As-Built Design (Milestone 6)

Due to the large amount of change between the as-built design and the detailed design, PSDMs for the change in design are considered separately from the other PSDMs. Thus, two groups of equations are selected for the as-built design milestone. The as-built design characteristic PSDMs are examined first, followed by the selection of the change in design PSDM predictive equations.

5.2.6.1 As-Built Design Characteristics

Table 5.2.6.1 presents the best prediction equations created from the design characteristic PSDMs available from the as-built design.

Table 5.2.6.1 Candidate Prediction Equations for the Design Characteristics at the As-Built Design Milestone

Equation Number	PSDMs Used	RSV	MSE
1	**PDL**	**0.995**	**41.3**
2	MO	0.984	138.0
3	FO	0.979	180.1
4	C	0.975	218.6
5	LO	0.951	422.6
6	**FO CKP**	**0.999**	**16.9**
7	**MO C**	**0.998**	**26.7**
8	FO CMO	0.997	30.1
9	PDL EX	0.997	30.8
10	PDL SC	0.997	32.2
11	LO EX CKP	1.000	1.1
12	**FO LO CKP**	**1.000**	**1.2**
13	**FO EX CKP**	**1.000**	**2.1**
14	FO SC CKP	1.000	2.4
15	FO IF CKP	1.000	4.4
16	MO LO CCI	1.000	4.4

Note: Selected equations and best values are in boldface.

Selection of Prediction Equations at Milestone 6 (Design Characteristics)

The "best" single and double PSDM prediction equations are selected based upon the improvement of the MSE value. It should be noted that the amount of collection effort to collect PDL line counts in equation 1 is significantly less than the effort required to collect any of the other selected equations. These other equations are selected because they address more characteristics of the design.

Of the double-PSDM prediction equations, equations 6 and 7 are selected because of their improved R-squared values (RSV) and mean squared error (MSE). Equation 7 is selected because it offers improvement over the single PSDM equations at a lower collection cost than prediction equation 6.

The same concerns about collection effort accompany the selection of the three-PSDM prediction equations. These equations are included because of decreased mean sum of squared error (MSE) and the coverage of more design characteristics. Following this reasoning, equation 11 is rejected because it does not address information flow. Prediction equations 12 and 13 are selected with reservations because while they address both internal complexity and information flow they require more data collection than the selected two metric equations.

No four-PSDM equations are selected due to the apparent accuracy of smaller equations at a lower collection cost.

5.2.6.2 As-Built Design Changes from Detailed Design

Table 5.2.6.2 presents the best prediction equations created from PSDMs measuring the change in the design from the detailed design milestone to the as-built design milestone.

Table 5.2.6.2 Prediction Equations for Changes in the Design from the Detailed Design to the As-Built Design Milestone

Equation Number	PSDMs Used	RSV	MSE
1	**PCPDL**	**0.992**	**70.2**
2	**ADMO**	**0.984**	**138.4**
3	ADPDL	0.967	280.8
4	PCMO	0.961	332.1
5	ADCI	0.947	457.9
6	PCC	0.935	560.3
7	PCFO	0.914	740.4
8	ADC	0.912	756.1
9	ADFO	0.894	910.1
10	**PCPDL ADC**	**0.998**	**27.4**
11	**ADMO PCCMO**	**0.997**	**29.9**
12	ADMO PCFI	0.997	32.1
13	PCPDL PCCI	0.997	34.9
14	PCPM PCSC	0.997	36.2
15	**ADLO PCC ADCKP**	**1.000**	**1.6**
16	**ADMO ADC ADCMO**	**1.000**	**3.4**
17	PCIF PCIFP PCDC	1.000	3.6
18	ADPDL PCMO ADFI	1.000	4.4
19	ADFO ADCKP ADCGT	1.000	4.6

Note: Selected equations and best values are in boldface.

Selection of Prediction Equations at Milestone 6 (Change in Design)

The changes in the design from detailed design to as-built design are drastic. In most cases, PSDM counts double. Due to improvements in the RSV and MSE between the best single-, double- and triple-PSDM prediction equations, the two best of each are selected. It should be noted that each of the change in design PSDMs requires data collection at both the detailed and as-built designs. Because the amount of change is a cause for concern, the multiple PSDM prediction equations are selected with reservations despite the large amount of data collection.

5.2.7 Unit Test Plans (Milestone 7)

Table 5.2.7 presents the best prediction equations created from PSDMs available at the unit test plan design milestone.

Table 5.2.7 Candidate Prediction Equations at the Unit Test Plan Design Milestone

Equation Number	PSDMs Used	RSV	MSE
1	**TSTE**	**0.909**	**785.0**
2	**TVES**	**0.891**	**938.1**
3	TTES	0.849	1302.7
4	TPCI	0.736	2268.7
5	SPCI	0.725	2365.4
6	TSTE TPCI	0.922	893.1
7	TSTE TVES	0.921	905.2
8	TSTE TPCC	0.920	916.6
9	TTES TSTE	0.914	984.0
10	TSTE TPMO	0.914	989.5
11	**TVES TPCC TPMO**	**0.987**	**225.0**
12	**SPCC TPCI VPCI**	**0.985**	**258.2**
13	TTES TPCI SPMO	0.980	345.8
14	TVES TPCC TPKP	0.965	600.0
15	TVES SPCC TPCI	0.955	770.8
16	**TVES TPCC SPCC TPMO**	**1.000**	**1.7**
17	TPCC SPCC VPCC TPCI	1.000	2.3
18	TPCC TPCI SPCI VPMO	1.000	2.5
20	TPCC SPCC TPCI VPCI	1.000	3.0
21	TVES TPCC TPMO SPMO	1.000	4.6

Note: Selected equations and best values are in boldface.

Selection of Prediction Equations at Milestone 7

In the unit test plan design milestone, a desired predictive equation should include the amount of testing and the amount of verification or expected results included with each test. Following this guideline, the two single-PSMD prediction equations are selected because they address the number of tests (TTES) and the number of verify sentences (TVES).

No two-PSDM equations are selected because, although the R-squared values (RSVs) show slight improvement, the mean sum of squared error (MSE) values increase.

The three- and four-PSDM prediction equations are selected because they address more of the characteristics relating to the amount and quality of unit testing. The resulting increase in the required collection effort may be a problem.

5.2.8 Code Walk-Throughs (Milestone 8)

The following table presents the best prediction equations created from the PSDMs available at the code walk-through design milestone.

Table 5.2.8 Candidate Prediction Equations at the Code Walk-Through Design Milestone

Equation Number	PSDMs Used	RSV	MSE
1	**AIND**	**0.967**	**283.7**
2	TAI	0.715	2448.0
3	AIARI	0.715	2450.4
4	AIR1	0.690	2665.5
5	**AIND AIPCC**	**0.991**	**100.5**
6	**AIND AIPCI**	**0.988**	**143.1**
7	AIND AIUT	0.969	359.8
8	AIARI AIND	0.967	376.0
9	AIR1 AIND	0.967	377.6
10	**AI1 AIND AIPCI**	**0.999**	**25.5**
11	**AIARI AIND AIPCI**	**0.998**	**33.3**
12	AIARI AIND AIPCC	0.996	71.2
13	AIND AIPCC AIUT	0.996	74.1
14	AIR1 AIND AIPCC AIPCI	0.999	24.1
15	AIARI AIR1 AIND AIPCI	0.999	50.1
16	AIR1 AIND AIPCI AIUT	0.999	51.0
17	AIARI AIND AIPCI AIUT	0.998	57.8

Note: Selected equations and best values are in boldface.

Selection of Prediction Equations at Milestone 8

Predictive software design measurements available at the code walk-through milestone address defect classification and how these counts compare to sizing measures. Action Items assigned to persons other than the designer (AIND) is the best single PSDM equation. The predictive value of this PSDM improves when combined with total action items per Cyclomatic Complexity or CI

in the two PSDM prediction equations. Ci counts are simpler to collect than Cyclomatic Complexity so despite the lower R-squared value (RSV) and higher mean sum of squared error (MSE), equation 6 might be a better choice than equation 5.

The selected three-PSDM prediction equations add more Action Item classifications to equation 6 while improving the R-squared value (RSV) slightly and the mean sum of squared error (MSE) significantly.

No four-PSDM predictive equations are selected because the additional collection effort is not justified with the slight improvements. Mean sum of squared error (MSE) even increases in some cases.

5.3 Minimizing Data Collection Effort

Another approach could be taken to the selection of prediction equations. An organization implementing the prediction model might desire to have a minimal number of distinct PSDMs to collect across the entire software design process. Selection of this minimal number of distinct PSDMs draws a single PSDM from each software design milestone while minimizing the number of distinct PSDMs. Fewer distinct PSDMs should lower the cost of training personnel and improve the accuracy and consistency of the collected data. This approach sacrifices accuracy for simplicity of implementation. In other words, what represents an acceptable compromise between the most accurate prediction and the least accumulated cost?

To minimize the number of distinct PSDMs while maintaining the accuracy of the predictions, software design milestones are considered in combination. PSDMs available at a given design milestone measure new information and

changes in the design artifacts from the previous milestone.[9] The values of PSDMs that are common to multiple design milestones are expected to change. The definition of the data collection, calculation, and interpretation of these PSDMs remains constant. The accuracy of predictions can be expected to suffer if prediction equations are selected so as to reduce the distinct number of PSDMs. In this selection, PSDMs are chosen to have the highest average R-squared value (RSV) and lowest average sum of squared error (MSE) across the applicable software design milestones. Each design milestone provides a single prediction equation and the results of applying these equations to the experimental group of CSCs is provided in Chapter 6. The results of the selection of minimal distinct PSDM prediction equations are summarized in Table 5.3.

[9] If the design artifacts do not reflect a change in a given PSDM's value between milestones, it is not included in the candidate PSDMs for the milestone occurring later in the design process. For example, total requirements count is not documented as changing after the post-review preliminary design. Therefore, total requirements count is not a candidate PSDM at any design milestone after the preliminary design review.

Table 5.3 Minimal Distinct PSDM Prediction Equation Results

Software Design Milestone	Model Number	PSDMs Used	RSV of Prediction Equation	MSE of Prediction Equation
Requirements Allocation	**1**	**TR**	**0.745**	**2197.9**
	2	DR	0.655	2967.6
Preliminary Design	3	DR	0.845	1332.7
	2	**TR**	**0.792**	**1785.8**
Preliminary Design Review	**3**	**AIARI**	**0.923**	**659.9**
	4	ARIPR	0.904	823.2
	5	TR	0.902	843.6
	6	AIT	0.826	1493.4
Detailed Design	**6**	**FO**	**0.969**	**266.5**
	7	C	0.958	363.0
	8	PDL	0.939	528.4
	9	MO	0.934	571.5
Detailed Design Review	10	AIT	0.972	240.7
	11	APP	0.948	448.5
	12	APC	0.941	511.0
	13	**AIARI**	**0.939**	**521.3**
As-Built Design (Design Characteristics)	14	PDL	0.995	41.3
	15	MO	0.984	138.0
	16	**FO**	**0.979**	**180.1**
	17	C	0.975	218.6
As-Built Design (Changes In Design)	18	PCPDL	0.992	70.2
	19	ADMO	0.984	138.4
	20	ADPDL	0.967	280.8
	21	PCMO	0.961	332.1
	22	PCC	0.935	560.3
	23	**PCFO**	**0.914**	**740.4**
	24	ADC	0.912	756.1
	25	ADFO	0.894	910.1
Unit Test Plans	**26**	**TSTE**	**0.909**	**785.0**
	27	TTES	0.849	1302.7
Code Walk-Throughs	28	TAI	0.715	2448.0
	29	**AIARI**	**0.715**	**2450.4**

Note: Selected prediction equations are in bold-face.

An example of the selection procedure is choosing total requirements (TR) as a prediction equation for both the requirements allocation and the preliminary design based on its higher average RSV (0.768) and lower average MSE (1991.9) than that of the derived requirements (DR) (0.750, and 2150.2 respectively).

In selecting the prediction equation at the preliminary design milestone, the count of Action Items requiring investigation and/or redesign (AIARI) is selected over total requirements because it performs better and is available for selection in the detailed design review and code walk-through design milestones. AIARI's selection improves the expected prediction and does not increase the distinct number of PSDMs.

Selection of a prediction equation at the as-built design milestone to address the change in design from detailed design to as-built design is dependent upon the selection of a prediction equation based upon the design characteristics. The fan-out (FO) PSDM prediction equation performs best on average for both the detailed design milestone and the as-built design milestone. In order to minimize the number of distinct PSDMs used, the change in design prediction equation must use FO. The percent change in fan-out (PCFO) performs better than the absolute difference in fan-out (ADFO) and PCFO is selected.

5.4 Evaluation of Regression Analysis

The creation of prediction equations through multiple regression provides an informative basis for decision making. A small set of selected prediction equations, using a minimal number of PSDMs, are generated at each software design milestone. The majority of the selected equations have R-squared values (RSVs) greater than 0.90. The lowest R-squared values occur at the requirements allocation. Considering the software design process, these lower RSVs are expected. At the requirements allocation milestone, little is known about the design and few PSDMs are available to choose from. This commonly results in lower RSV values.

A minimum distinct PSDM prediction model is created in section 5.3. This includes a single prediction equation selected at each software design milestone. The selection is based upon minimizing the number of distinct PSDMs required across all software design milestones while keeping the RSV as high as possible.

Please refer to Appendix E for the PSDM data used in creating the prediction equations and Appendix F for the coefficients calculated for the selected prediction equations.

6 Evaluation of Selected Prediction Equations

Recall from Chapter Four that the test group of CSCs provide the design artifacts and data for creating potential prediction equations. The most promising of these prediction equations are presented in Chapter Five. A small set of prediction equations are selected for evaluation at each software design milestone. This chapter documents the results of evaluating the selected prediction equations from Chapter Five with PSDM data from the experimental group of CSCs. A single "best" prediction equation is chosen at each software design milestone based upon the results of the evaluation.

In addition, the minimal distinct PSDM prediction model from Section 5.3 of Chapter Five is evaluated with data from the experimental group of CSCs. The results are compared with those of the "best" prediction equations.

6.1 Means of Evaluation

Predictions of defect proneness are calculated using the selected prediction equations from each design milestone applied to data from the experimental group of CSCs. The accuracy of the resulting predictions is assessed by evaluating the predicted number of defects versus the actual number of defects. The actual rankings of CSCs by the number of defects is compared to the predicted rankings of CSCs by the predicted number of defects. The following items are used to evaluate the results:

- Wilcoxon signed rank value.
- Sum of squared differences.
- Corrected sum of squared differences.
- Number of PSDMs required.

Wilcoxon Signed Rank Value

The Wilcoxon signed rank value is calculated by ranking the absolute value of the difference between the predicted number of defects and the actual number of defects discovered during formal testing. The rank values of the positive differences and negative differences are then added up separately and the smaller of these, in terms of absolute value, is taken as the signed rank value. A negative signed rank value means the sum of the ranks for the negative differences is smaller than that of the positive differences. The negative signed rank also means that the prediction equation tends to predict more defects than actually occur because there are more and greater positive differences. Lower sign rank values mean the predictions tend to be high and low by consistent amounts (i.e. many tied rankings). This generally implies more accurate prediction values. With five CSCs being used in the simulation, the Wilcoxon rank sum value will range from -7.5 to 7.5.

Accompanying the signed rank value is a probability (Pr >=|S|). This is the probability that there exists a signed rank value with an absolute value greater than the one calculated. This probability value includes the sample size in its calculation and is used to compare ranked sum values for samples of differing sizes.

Because of the small number of samples used in evaluating the selected prediction equations, the Wilcoxon signed rank value has low power of discrimination (Milton & Arnold, 1986, pp. 314-315 & p. 592). A better approach is to use the signed rank value in conjunction with the number of PSDMs as a tie breaker when the other evaluation statistics fail to identify a single best equation.

Sum of Squared Error

Uncorrected sum of square error (USS) is the sum of the squared differences between the actual and the predicted values or ranks. A low sum of squares indicates more accurate predictions.

The uncorrected sum of squared differences for rank values (RUSS) determines accuracy of the orderings of the CSCs when ranked by the predicted number of defects. A zero RUSS value indicates rankings are correctly predicted. A RUSS value of 2 indicates two adjacent ranks are switched. For a five sample ranking, the worst RUSS value for sum of squared differences of ranks is 40.

Corrected Sum of Squared Differences

Corrected sum of squares (CSS) is the sum of the squared differences minus the average of the squared sum of the differences. This is a measure of the variability in the data. Low corrected sum of squares indicates low variability in the difference between the actual and predicted values.

Number of PSDMs Required

The number of PSDMs used by a prediction equation is an indication of the amount of effort required for data collection. Given two prediction equations with similar accuracy in rankings, USS, and CSS, the preferred equation is the one that requires less data collection effort.

6.2 Evaluation Procedure and Analysis

The evaluation of the selected prediction equations is based upon their performance using the PSDM data from the experimental group of CSCs. The expected numbers of defects are calculated for each of the experimental CSCs using the prediction equations selected at each of the design milestones.[1] These predictions are evaluated based upon their accuracy in predicting the rankings of CSCs by defect potential and the predicted defect counts.

The following sections present a milestone by milestone evaluation of the results of applying the selected prediction equations from Chapter Five to the PSDM data from the experimental CSCs. A single "best" prediction equation is selected at each design milestone based upon the selection criteria discussed above.[2]

6.2.1 Requirements Allocation (Milestone 1)

In Chapter Five, three prediction equations are selected for evaluation at the requirements allocation design milestone. These equations are presented again in Table 6.2.1.1 with their associated R-squared values (RSV) and mean squared error values (MSE).

---

[1] Please refer to Appendix F for the selected prediction equations, PSDMs used, and their calculated coefficients.

[2] Acronyms are used for the labeling of the PSDMs. A complete list of all acronyms is available in Appendix D.

Table 6.2.1.1 Results of Ranking By Defect Potential for the Requirements Allocation Milestone

Equation Number	PSDMs Used	RSV (of Equation)	MSE
1	TR	**0.745**	**2197.9**
2	FC	0.742	2217.6
3	AR	0.726	2356.5

Note: Best values are in boldface.

Evaluation of Selected Prediction Equations at Milestone 1

To evaluate the selected prediction equations, the requirements allocation PSDM data from the experimental group of CSCs is used to calculate predictions of the number of defects detected. Table 6.2.1.2 summarizes the performance of the selected prediction equations.

Table 6.2.1.2 Results of Predicted Defect Counts for the Requirements Allocation Milestone

Equation Number	PSDMs Used	RUSS	USS	CSS	Sign Rank	Pr>=|S|
1	TR	2.0	2110.1	616.9	-7.5	0.063
2	FC	8.0	4294.9	1956.3	**-4.5**	0.313
3	**AR**	**0.0**	**815.8**	**153.3**	-7.5	0.063

Note: Selected equations and best values are in boldface.

Selection of a Best Prediction Equation at Milestone 1

The best prediction equation uses allocated requirements (AR). It correctly predicts the rank of the experimental CSCs by defect proneness (RUSS value of 0.0). The AR prediction equation also has the lowest sum of squared differences (USS) and corrected sum of squared differences (CSS). Table 6.2.1.3 summarizes the results of using AR to predict defect potential.

Table 6.2.1.3 AR Prediction Equation Results

CSC	Actual Rank	Predicted Rank	Actual Number of Defects	Predicted Number of Defects
EI	1	1	10	8.9
EO	2	2	41	26.7
EF	3	3	53	37.8
EE	4	4	64	53.4
EB	5	5	103	86.7

6.2.2 Preliminary Design (Milestone 2)

Recall from Chapter Five that four prediction equations are selected for evaluation at the preliminary design milestone. These equations are presented again in Table 6.2.2.1 with their associated R-squared values (RSV) and mean squared error values (MSE).

Table 6.2.2.1 Results of Ranking By Defect Potential for the Preliminary Design Milestone

Equation Number	PSDMs Used	RSV (of Equation)	MSE
1	DR	0.845	1332.7
5	PTO DR	0.942	667.3
10	PTO DR PCR	**0.999**	**10.9**
11	PTO DR ADR	**0.999**	11.6

Note: Best values are in boldface.

Evaluation of Selected Prediction Equations at Milestone 2

To evaluate the selected prediction equations, the preliminary design PSDM data from the experimental group of CSCs is used to calculate predictions of the number of defects detected. Table 6.2.2.2 summarizes the performance of the selected prediction equations.

Table 6.2.2.2 Results of Predicted Defect Counts for the Preliminary Design Milestone

Equation Number	PSDMs Used	RUSS	USS	CSS	Sign Rank	Pr>=|S|
1	**DR**	8.0	**21334.0**	**16070.0**	**2.5**	0.625
5	PTO DR	8.0	44583.4	31954.0	3.5	0.438
10	PTO DR PCR	8.0	32664.6	24419.0	3.5	0.438
11	PTO DR ADR	8.0	34406.0	25837.0	3.5	0.438

Note: Selected equations and best values are in boldface.

Selection of the Best Prediction Equation at Milestone 2

The prediction equations selected for the preliminary design milestone are not very good. The predicted rankings are all incorrect. The RUSS value of 8 indicates that two CSCs separated by another CSC are in the wrong position. The sum of squared differences (USS) and the corrected sum of squared differences (CSS) values are all high indicating that the predicted defect counts are not accurate. The "best" of these weak prediction equations is the number of derived requirements (DR) because it has the lowest USS and CSS values. Table 6.2.2.3 summarizes the results of using DR as a predictor of defect potential.

Table 6.2.2.3 DR Prediction Equation Results

CSC	Actual Rank	Predicted Rank	Actual Number of Defects	Predicted Number of Defects
EI	1	1	10	0.0
EO	2	4	41	106.1
EF	3	3	53	61.9
EE	4	2	64	35.4
EB	5	5	103	229.9

6.2.3 Preliminary Design Review (Milestone 3)

Recall from Chapter Five that six prediction equations are selected for evaluation at the preliminary design review milestone. These equations are presented again in Table 6.2.3.1 with their associated R-squared values (RSV) and mean squared error values (MSE).

Table 6.2.3.1 Results of Ranking By Defect Potential for the Preliminary Design Review Milestone

Equation Number	PSDMs Used	RSV (of Equation)	MSE
1	**AIARI**	0.923	659.9
7	ARIPR DPC23DE	0.992	87.8
8	AIARI DA23TR	0.992	89.9
11	TAPR AIR1 DPC23DE	1.000	5.4
12	ARIPR AR DPC23DE	0.999	13.1
13	ARIPR DA23TR DPC23DE	0.999	15.8

Note: Selected equations and best values are in boldface.

Evaluation of Selected Prediction Equations at Milestone 3

To evaluate the selected prediction equations, the preliminary design review PSDM data from the experimental group of CSCs is used to calculate predictions of the number of defects detected. Table 6.2.3.2 summarizes the performance of the selected prediction equations.

Table 6.2.3.2 Results of Predicted Defect Counts for the Preliminary Design Review Milestone

Equation Number	PSDMs Used	RUSS	USS	CSS	Sign Rank	Pr>=|S|
1	**AIARI**	**2.0**	**5613.2**	**944.3**	7.5	0.063
7	ARIPR DPC23DE	12.0	242116.6	70903.0	7.5	0.063
8	AIARI DA23TR	**2.0**	4051.4	1182.8	7.5	0.063
11	TAPR AIR1 DPC23DE	20.0	1897782.0	914189.0	7.5	0.063
12	ARIPR AR DPC23DE	12.0	191067.5	55981.0	7.5	0.063
13	ARIPR DA23TR DPC23DE	20.0	154030.5	46443.0	7.5	0.063

Note: Selected equations and best values are in boldface.

Selection of Prediction Equations at Milestone 3

At the preliminary design review milestone, two prediction equations are potentially the best. These are equations 1 and 8. Both have rank sum of squared differences (RUSS) values of 2 indicating that two adjacently ranked CSCs are out of order. Equation 1 has the slightly higher sum of squared differences (USS) and equation 8 has a slightly higher corrected sum of squared differences (CSS) values. The performance of the two prediction equations is very similar. Equation 1 is selected because it uses only one PSDM as opposed to equation 8 which requires two PSDMs. Table 6.2.3.3 presents the results of using Action Items requiring redesign/investigation (AIARI) as a predictor of defect potential

Table 6.2.3.3 AIARI Prediction Equation Results

CSC	Actual Rank	Predicted Rank	Actual Number of Defects	Predicted Number of Defects
EI	1	1	10	41.0
EO	2	2	41	68.4
EF	3	4	53	109.4
EE	4	3	64	82.0
EB	5	5	103	135.0

6.2.4 Detailed Design (Milestone 4)

Recall from Chapter Five that seven prediction equations are selected for evaluation at the detailed design milestone. These equations are presented again in Table 6.2.4.1 with their associated R-squared values (RSV) and mean squared error values (MSE).

Table 6.2.4.1 Results of Ranking By Defect Potential for the Detailed Design Milestone

Equation Number	PSDMs Used	RSV (of Equation)	MSE
1	**FO**	0.969	266.5
2	C	0.958	363.0
6	CI MOPRE	0.993	84.1
7	FI C	0.991	103.2
11	IFP EX C	1.000	9.2
12	PDLPPRE CI CMO	0.999	10.8
14	PDLPPRE IF EX	0.999	13.1

Note: Selected equations and best values are in boldface.

Evaluation of Selected Prediction Equations at Milestone 4

To evaluate the selected prediction equations, the detailed design PSDM data from the experimental group of CSCs is used to calculate predictions of the number of defects detected. Table 6.2.4.2 summarizes the performance of the selected prediction equations.

Table 6.2.4.2 Results of Predicted Defect Counts for the Detailed Design Milestone

Equation Number	PSDMs Used	RUSS	USS	CSS	Sign Rank	Pr>=|S|
1	**FO**	**2.0**	2686.5	2065.9	3.5	0.438
2	C	8.0	3086.4	3010.2	**2.5**	**0.625**
6	CI MOPRE	10.0	50357	25079.0	-7.5	0.063
7	FI C	14.0	5587.8	5413.3	**2.5**	**0.625**
11	IFP EX C	**2.0**	**1979.0**	**1077.0**	6.5	0.125
12	PDLPPRE CI CMO	16.0	156473.0	118401.0	-4.5	0.313
14	PDLPPRE IF EX	8.0	39153.0	18734.0	7.5	0.063

Note: Selected equations and best values are in boldface.

Selection of the Best Prediction Equations at Milestone 4

Choosing the best prediction equation at the design milestone is difficult. Equations 1 and 11 both have the low rank sum of squared differences (RUSS) values (i.e. RUSS value of 2.0). Their sum of squared differences (USS) and

corrected sum of squared differences (CSS) values are close. Equation 11 provides the more accurate prediction of the number of defects (based on USS and CSS values) but at the cost of collecting three PSDMs rather than one in equation 1. Because of this, equation 1 is selected as the best prediction equation. Table 6.2.4.3 summarizes the results of using fan-out (FO) as a predictor of defect potential.

Table 6.2.4.3 FO Prediction Equation Results

CSC	Actual Rank	Predicted Rank	Actual Number of Defects	Predicted Number of Defects
EI	1	1	10	4.9
EO	2	2	41	38.7
EF	3	4	53	103.7
EE	4	3	64	72.1
EB	5	5	103	107.3

6.2.5 Detailed Design Review (Milestone 5)

In Chapter Five, five prediction equations are selected for evaluation at the detailed design review milestone. These equations are presented again in Table 6.2.5.1 with their associated R-squared values (RSV) and mean squared error values (MSE).

Table 6.2.5.1 Results of Ranking By Defect Potential for the Detailed Design Review Milestone

Equation Number	PSDMs Used	RSV (of Equation)	MSE
1	AIT	0.972	240.7
5	APC TNDA	0.989	128.9
6	AIT TNDAPC	0.988	132.1
10	APC ARIPP TNDA	**0.997**	**47.3**
11	APP ARIPP TNDA	0.996	68.6

Note: Best values are in boldface.

Evaluation of Selected Prediction Equations at Milestone 5

At the detailed design review milestone, PSDM data from the experimental group of CSCs is used to evaluate the selected prediction equations. This evaluation is based upon the calculation of predictions for the number of defects detected. Table 6.2.5.2 summarizes the performance of the selected prediction equations.

Table 6.2.5.2 Results of Predicted Defect Counts for the Detailed Design Review Milestone

Equation Number	PSDMs Used	RUSS	USS	CSS	Sign Rank	Pr>=|S|
1	**AIT**	**4.0**	**6746.8**	**6745.8**	**0.5**	**1**
5	APC TNDA	18.0	43087.4	23475.7	5.5	0.188
6	AIT TNDAPC	**4.0**	12400.9	11073.1	1.5	0.813
10	APC ARIPP TNDA	18.0	54845.5	30431.2	6.5	0.125
11	APP ARIPP TNDA	24.0	52305.9	38713.2	2.5	0.625

Note: Selected equations and best values are in boldface.

Selection of Prediction Equations at Milestone 5

At the detailed design review milestone, the best prediction equation in terms of rank sum of squared differences (RUSS), sum of squared differences (USS), and corrected sum of squared differences (CSS) is the total number of Action Items (AIT). It should be noted that none of the selected prediction equations performs particularly well when compared with the results of the predictions at the detailed design and as-built design milestones. Table 6.2.5.3 summarizes the results of using AIT as a predictor of defect potential.

Table 6.2.5.2 AIT Prediction Equation Results

CSC	Actual Rank	Predicted Rank	Actual Number of Defects	Predicted Number of Defects
EI	1	1	10	20.3
EO	2	3	41	43.9
EF	3	2	53	28.9
EE	4	5	64	123.1
EB	5	4	103	52.5

6.2.6 As-Built Design (Milestone 6)

The prediction equations for design characteristics and the change in the design from the detailed design to the as-built design are evaluated separately.

6.2.6.1 Results of As-Built Design Characteristics

Five prediction equations are selected in Chapter Five for evaluation of the design characteristics of the as-built design. These equations are presented again in Table 6.2.6.1.1 with their associated R-squared values (RSV) and mean squared error values (MSE).

Table 6.2.6.1.1 Results of Ranking By Defect Potential for As-Built Design Milestone

Equation Number	PSDMs Used	RSV (of Equation)	MSE
1	PDL	0.995	41.3
6	FO CKP	0.999	16.9
7	MO C	0.998	26.7
12	FO LO CKP	**1.000**	**1.2**
13	FO EX CKP	**1.000**	2.1

Note: Best values are in boldface.

Evaluation of Selected Prediction Equations at Milestone 6 (Design Characteristics)

To evaluate the selected prediction equations for the as-built design characteristics, PSDM data from the experimental group of CSCs is used to calculate predictions of the number of defects detected. Table 6.2.6.1.2 summarizes the performance of the selected prediction equations.

Table 6.2.6.1.2 Results of Predicted Defect Counts for the As-Built Design Milestone

Equation Number	PSDMs Used	RUSS	USS	CSS	Sign Rank	Pr>=|S|
1	PDL	12.0	2524.4	2488.5	-3.5	0.438
6	**FO CKP**	**0.0**	**866.2**	**610.8**	3.5	0.438
7	MO C	12.0	3518.4	2857.7	-3.5	0.438
12	FO LO CKP	2.0	1287.8	1026.7	3.5	0.438
13	FO EX CKP	0.0	833.9	458.3	5.5	0.188

Note: Selected equations and best values are in boldface.

Selection of the Best Prediction Equation at Milestone 6 (Design Characteristics)

The prediction equation using the PSDMs fan-out (FO) and Cyclomatic Complexity per LOPDL (CKP) is the best of the selected equations. A zero value for rank sum of squared differences (RUSS) means the ranking of the experimental CSCs in terms of the number of defects is correct. The low sum of squared differences (USS) and corrected sum of squared differences (CSS) values mean the predicted numbers of defects are generally close to the actual numbers of defects. Table 6.2.6.1.3 summarizes the results of using FO and CKP as predictors of defect potential.

Table 6.2.6.1.3 FO and CKP Prediction Equation Results

CSC	Actual Rank	Predicted Rank	Actual Number of Defects	Predicted Number of Defects
EI	1	1	10	28.0
EO	2	2	41	52.6
EF	3	3	53	59.4
EE	4	4	64	77.4
EB	5	5	103	89.3

6.2.6.2 Results of As-Built Changes in Design

Recall from Chapter Five that six prediction equations are selected for evaluation of the changes in design PSDMs at the as-built design milestone. These equations are presented again in Table 6.2.6.2.1 with their associated R-squared values (RSV) and mean squared error values (MSE).

Table 6.2.6.2.1 Results of Ranking By Defect Potential for the As-Built Design Milestone (Changes)

Model Number	PSDMs Used	RSV of equation	MSE
1	PCPDL	0.992	70.2
2	ADMO	0.984	138.4
3	PCPDL ADC	0.998	27.4
4	ADMO PCCMO	0.997	29.9
5	ADLO PCC ADCKP	**1.000**	**1.6**
6	ADMO ADC ADCMO	**1.000**	3.4

Note: Best values are in boldface.

Evaluation of Selected Prediction Equations at Milestone 6 (Change in Design)

To evaluate the selected prediction equations for the change in design PSDMs, PSDM data from the experimental group of CSCs is used to calculate predictions of the number of defects detected. Table 6.2.6.2.2 summarizes the performance of the selected prediction equations.

Table 6.2.6.2.2 Results of Predicted Defect Counts for the As-Built Design Milestone (Changes)

Equation Number	PSDMs Used	RUSS	USS	CSS	Sign Rank	Pr>=|S|
1	PCPDL	30.0	29225.6	23957.6	2.5	0.625
2	**ADMO**	**6.0**	**5562.5**	**2069.8**	-7.5	0.063
3	PCPDL ADC	30.0	41917.6	30588.7	2.5	0.625
4	ADMO PCCMO	16.0	6132.6	5966.3	**-1.5**	**0.813**
5	ADLO PCC ADCKP	22.0	10039.8	8993.5	-2.5	0.625
6	ADMO ADC ADCMO	32.0	20630.5	18133.4	**1.5**	**0.813**

Note: Selected equations and best values are in boldface.

Selection of the Best Prediction Equation at Milestone 6 (Change in Design)

The best change in design PSDM prediction equation is selected because the predicted rankings are close to the actual. Prediction equation 2 uses the absolute difference in the number of modules between the detailed design and the as-built design (ADMO). The rank sum of squared differences (RUSS) value of 6 means that one CSC is two places higher or lower than it should be.[3] This prediction equation also has the lowest uncorrected sum of squared error (USS) and corrected sum of squared error (CSS) value. Table 6.2.6.2.3 summarizes the results of using ADMO as a predictor of defect potential.

Table 6.2.6.2.3 ADMO Prediction Equation Results

CSC	Rank	Predicted Rank	Actual Number of defects	Predicted Number of Defects
EI	1	1	10	8.8
EO	2	4	41	38.8
EF	3	2	53	15.0
EE	4	3	64	18.8
EB	5	5	103	57.5

[3] Please refer to the CSC EO in Table 6.2.6.2.3.

6.2.7 Unit Test Plans (Milestone 7)

Five prediction equations are selected in Chapter Five for evaluation at the unit test plan milestone. These equations are presented again in Table 6.2.7.1 with their associated R-squared values (RSV) and mean squared error values (MSE).

Table 6.2.7.1 Results of Ranking By Defect Potential for the Unit Test Plan Milestone

Equation Number	PSDMs Used	RSV (of Equation)	MSE
1	TSTE	0.909	785.0
2	TVES	0.891	938.1
11	TVES TPCC TPMO	0.987	225.0
12	SPCC TPCI VPCI	0.985	258.2
16	TVES TPCC SPCC TPMO	**1.000**	**1.7**

Note: Best values are in boldface.

Evaluation of Selected Prediction Equations at Milestone 7

To evaluate the selected prediction equations, the unit test plan PSDM data from the experimental group of CSCs is used to calculate predictions of the number of defects detected. Table 6.2.7.2 summarizes the performance of the selected prediction equations.

Table 6.2.7.2 Results of Predicted Defect Counts for the Unit Test Plan Milestone

Equation Number	PSDMs Used	RUSS	USS	CSS	Sign Rank	Pr>=|S|
1	**TSTE**	**8.0**	**3596.0**	**3546.0**	**0.5**	1
2	TVES	18.0	25804.0	23113.28.0	**0.5**	1
11	TVES TPCC TPMO	20.0	3463283.0	2748502.0	3.5	0.438
12	SPCC TPCI VPCI	22.0	91372.0	47878.67.0	5.5	0.188
16	TVES TPCC SPCC TPMO	18.0	3214087.0	2594184.0	3.5	0.438

Note: Selected equations and best values are in boldface.

Selection of the Best Prediction Equation at Milestone 7

The prediction equation chosen as best at the unit test plan design milestone is equation 1 which uses the total number of test steps (TSTE). While the values of sum of squared differences (USS) and the corrected sum of squared differences (CSS) are relatively low, the rank sum of squared differences (RUSS) value is poor. A RUSS value of 8 means two CSCs separated by one CSC in the rankings should have been switched. In a set of five CSCs, this error in rankings is cause for concern. Table 6.2.7.3 summarizes the results of using TSTE as a predictor of defect potential.

Table 6.2.7.3 TSTE Prediction Equation Results

CSC	Actual Rank	Predicted Rank	Actual Number of Defects	Predicted Number of Defects
EI	1	1	10	9.9
EO	2	2	41	44.5
EF	3	5	53	84.8
EE	4	4	64	64.3
EB	5	3	103	52.3

6.2.8 Code Walk-Throughs (Milestone 8)

Recall from Chapter Five that five prediction equations are selected for evaluation at the code walk-through milestone. These equations are presented again in Table 6.2.8.1 with their associated R-squared values (RSV) and mean squared error values (MSE).

Table 6.2.8.1 Results of Ranking By Defect Potential for the Code Walk-Through Milestone

Equation Number	PSDMs Used	RSV (of Equation)	MSE
1	AIND	0.967	283.7
5	AIND AIPCC	0.991	100.5
6	AIND AIPCI	0.988	143.1
10	AI1 AIND AIPCI	**0.999**	**25.5**
11	AIARI AIND AIPCI	0.998	33.3

Note: Best values are in boldface.

Evaluation of Selected Prediction Equations at Milestone 8

To evaluate the selected prediction equations, the code walk-through PSDM data from the experimental group of CSCs is used to calculate predictions of the number of defects detected. Table 6.2.8.2 summarizes the performance of the selected prediction equations.

Table 6.2.8.2 Results of Predicted Defect Counts for the Code Walk-Through Milestone

Equation Number	PSDMs Used	RUSS	USS	CSS	Sign Rank	Pr>=|S|
1	AIND	0.5	**4348.9**	**973.9**	7.5	0.063
5	**AIND AIPCC**	**0.0**	4630.5	2062.1	6.5	0.125
6	AIND AIPCI	2.0	5538.2	5503.7	1.5	0.813
10	AIR1 AIND AIPCI	2.0	8972.7	8326.5	**-0.5**	**1.000**
11	AIARI AIND AIPCI	2.0	7912.2	7455.7	**-0.5**	**1.000**

Note: Selected equations and best values are in boldface.

Selection of the Best Prediction Equation at Milestone 8

Prediction equation 5, which uses the number of non-designer Action Items (AIND) and the number of Action Items per Cyclomatic Complexity (AIPCC) is chosen as "best". The zero rank sum of squared differences (RUSS) value means the predicted ranking of experimental CSCs in terms of the number of defects is correct. The prediction equation also has the second lowest values for the sum of squared differences (USS) and the corrected sum of squared

differences (CSS). Table 6.2.8.3 summarizes the results of using AIPC as a predictor of defect potential.

Table 6.2.8.3 AIPC Prediction Equation Results

CSC	Actual Rank	Predicted Rank	Actual Number of Defects	Predicted Number of Defects
EI	1	1	10	0.7
EO	2	2	41	50.8
EF	3	3	53	99.0
EE	4	4	64	104.5
EB	5	5	103	129.3

6.3 Evaluation Results

The prediction equations performed reasonably well at all the design milestones. Table 6.3.1 summarizes the results of the best equations at each milestone.

Table 6.4.1 Summary Table of Best Prediction Equations at each Milestone

Milestone	PSDMs Used	RSV	MSE	RUSS	USS	CSS
Requirements Allocation	AR	0.726	2356.5	**0.0**	**815.8**	**153.3**
Preliminary Design Review	DR	0.845	1332.7	8.0	21334.1	16070.5
Preliminary Design Review	AIARI	0.923	659.9	2.0	5613.2	944.3
Detailed Design	FO	0.969	266.5	2.0	2686.5	2065.9
Detailed Design Review	AIT	0.972	240.6	4.0	6746.8	6745.8
As-Built Design (design characteristics)	FO CKP	**0.999**	16.9	**0.0**	866.2	610.8
As-Built Design (change in design)	ADMO	0.984	138.4	6.0	5562.	2069.8
Unit Test Plan	TSTE	0.909	785.0	8.0	3595.8	3546.0
Code Walk-Throughs	AIND AIPCC	0.991	100.5	**0.0**	4630.5	2062.1

Note: Best values are in boldface.

Prediction equations are selected for use in the evaluation based upon the RSV and MSE values. The selected equations are calculated with the experimental CSC PSDM data and evaluated by the RUSS, USS, and CSS values.

The R-squared value (RSV) and the mean sum of squared error (MSE) show how well the equations fit the original data of the test group of CSCs. RSVs close to one indicate that most of the variability in the dependent variable (actual defect counts) is explained by the independent variables (PSDMs). From Table 6.4, the highest RSVs occur at the as-built design milestone (average of 0.991) and the code walk-through design milestone (0.991). This value indicates that the prediction equations at these milestones explain most of the variability in the actual defect counts for the test group of CSCs. Low MSE values at these design milestones indicate that the predicted number of defects is generally close to the actual number of defects for the test group of CSCs.

For evaluation of the selected prediction equations, the experimental group of CSCs provides the PSDM data. The sum of squared differences (USS) and corrected sum of squared differences show how well the prediction equations predict the number of defects for the CSCs of the experimental group. From Table 6.4, the lowest of these values occur at the requirements allocation and as-built design milestones. Predictions are expected to be more accurate at the as-built design milestone due primarily to the maturity of the design. The relatively high accuracy (USS of 815.1 and CSS of 153.3) at the requirements allocation is not expected.

The rank sum of squared differences (RUSS) shows how close the predicted rankings by defect potential are to the actual rankings. The predicted rankings of the experimental CSCs are accurate (RUSS value of 0.0) at the requirements allocation, as-built design (design characteristics), and the code walk-through design milestones.

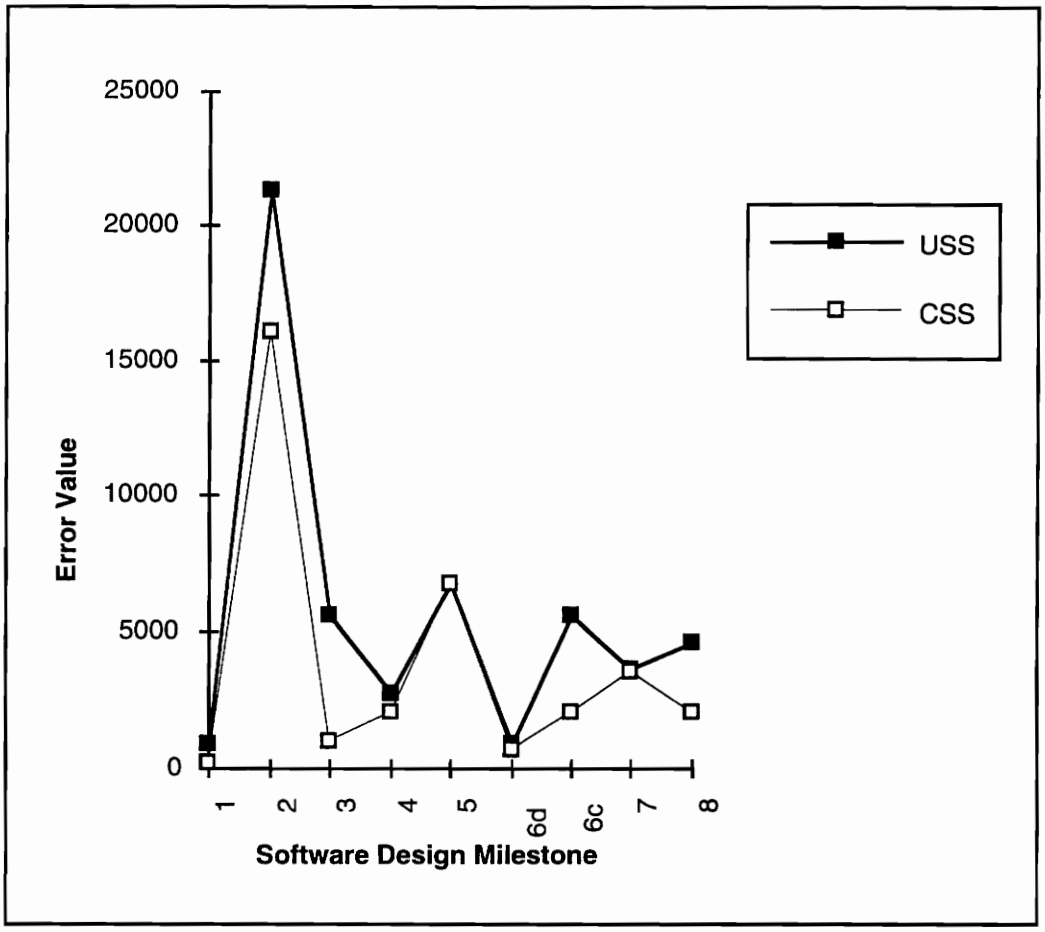


Figure 6.3.1 Prediction USS and CSS Between Design Milestones

Prediction USS and CSS values

The figure 6.3.1 shows the USS and CSS values of the difference between the actual number of defects and the predicted number of defects for each of the design milestones. The most accurate predictions considering both USS and CSS occur at the requirements allocation milestone (milestone 1) and when using the design characteristics at the as-built design milestone (milestone 6d). The worst prediction of the number of defects occurs at the preliminary design milestone (milestone 2).

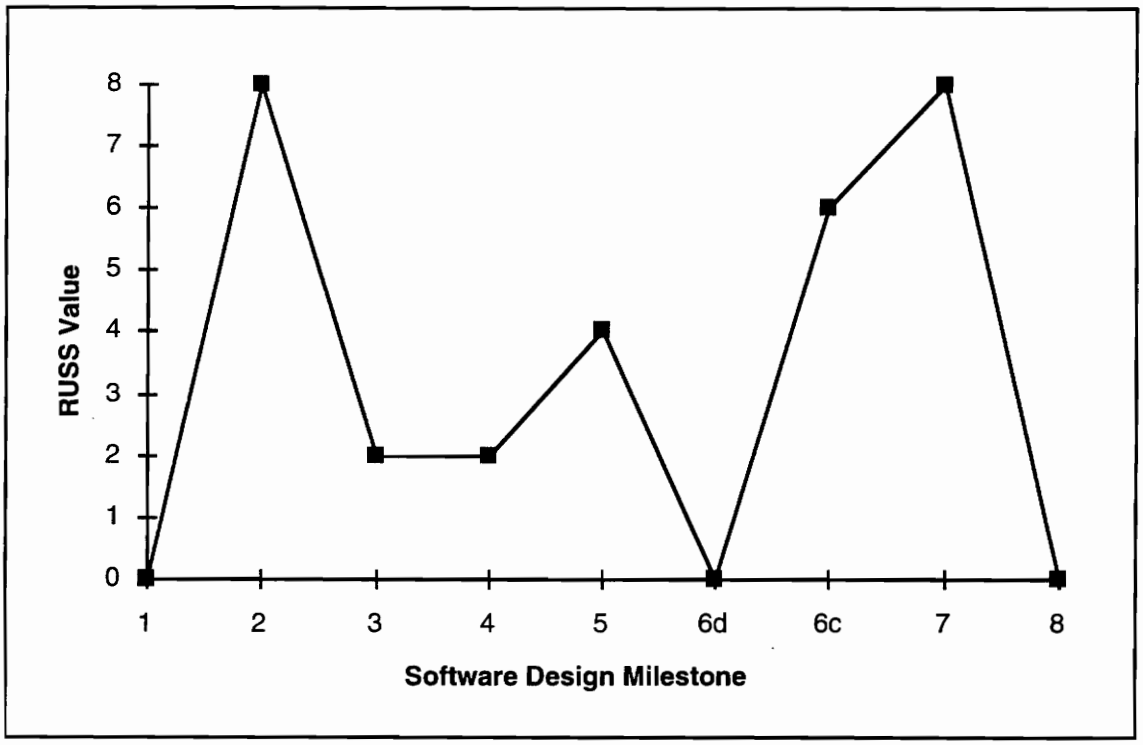


Figure 6.3.2 Accuracy of Rankings by Milestone

Prediction RUSS Values

The figure 6.3.2 shows the accuracy of the ranking of the experimental group of CSCs. This accuracy is measured by the rank sum of squared differences (RUSS).

The best design milestones for predicting the rank of the experimental CSCs by defect potential are the requirements allocation, the as-built design, and the code walk-through. The rank predictions at the preliminary design review and the detailed design are also good, except for two CSCs being transposed (as indicated by the RUSS value of two). The worst predictions of rank occur at the preliminary design and unit test plan milestone. The RUSS value is eight at these milestones.

The Most Valuable Prediction Equation and PSDMs

From the above table, the best prediction point is the as-built design milestone. This prediction equation is comprised of Fan-Out (FO) and Cyclomatic Complexity per thousand lines of PDL (CKP). Three reasons can be offered for concluding that the as-built prediction equation is the best predictor. The first reason is that this equation has the second best values of RSV and MSE calculated at its creation, i.e. the prediction equation fits the test group PSDM data very well. The second reason is that the values of USS and CSS, from the evaluation phase, are also the second best and relatively low; thus the prediction equation performs well in predicting the number of defects from the PSDM data of the experimental group. The final reason for selecting the as-built design milestone prediction equation as the best is that since the rank sum of squared differences is zero, the prediction equation correctly ranks the experimental group CSCs by defect potential.

Accurate predictions at the as-built design milestone are not surprising and should be expected. The design nearly completed at this design phase and all design artifacts are available. Corrections or changes made to the design at the as-build design milestone based upon predictions of defect potential are more costly than corrections or changes made at earlier design milestones. Following this reasoning, prediction equations from the detailed design, preliminary design review, and requirements allocation may be more desirable.

Valuable Areas of Concern in Predictive Measurement

Considering PSDMs by the areas of concern in predictive measurement, Fan-Out is used in both the detailed design and as-built design milestones. This suggests that a predictive measurement scheme should include Fan-Out and that other information flow measures might prove useful.

Cyclomatic Complexity is also used in two of the best prediction equations. It is simple to collect and gives an indication of the number of control paths

through a design. This number of control paths can be used to determine the amount of testing.

Accuracy of Equations Versus the Time of Measurement

For the prediction equations and design milestones addressing the design as a product (i.e. requirements allocation, preliminary design, detailed design, and as-built design) the accuracy improves as the design matures. The improvement in the MSE and RSV values shows that the PSDMs drawn from the test group of CSCs explain more of the variability in the actual number of defects. This is expected due to the increasing quality and number of candidate PSDMs as the software design matures.

When the selected prediction equations are evaluated with data from the experimental group of CSCs, they produce fairly accurate rankings of the CSCs by defect potential.[4] The USS and CSS values are also relatively low. The final "design as product" prediction equation at the as-built design milestone is selected as the best prediction equation.

Poor Prediction Equations, PSDMs, and Milestones

There are three weak areas for predicting defect potential. The first is the review and walk-through milestones. Based on the work of Chillarege et al. (1992) PSDMs addressing defect categorization are expected to be more valuable in predicting the number of defects.[5] Except for the detailed design review milestone, the predicted rankings of CSCs by defect potential are relatively good. The PSDMs addressing defect categorization have a value in

---

[4] The preliminary design milestone provides the weakest prediction of rank with a RUSS value of 8.0.

[5] Empirical results enabled designers to identify areas of a developing design that are prone to defects. This identification is extended to evaluate the maturity of the developing design based upon the types and quantities of defects detected. Please refer to Chillarege et. al. (1992) for more information.

predicting the rankings of CSCs but need refinement and possibly more data to provide accurate predictions of the number of defects.

The second weak area in the prediction of defect potential occurs at the as-built design milestone. The large amount of change from the detailed design to the as-built design is a cause for concern. All the potential prediction equations addressing the various changes in design characteristics are relatively ineffective at predicting the actual number of defects and the subsequent rankings. The absolute difference in the number of modules (ADMO) is the "best" of the prediction equations but it ranks the CSCs incorrectly. The CSC that contains the second least number of defects is ranked as containing the second most number of defects. This suggests that the area of design change requires further investigation.

The final weak area is the unit test plan design milestone. Because the PSDMs and prediction equations are designed to aid testers in identifying defect-prone areas of a developing design, the unit test plans should have been a valuable point of prediction. Other researchers working with this software project have expressed the opinion that the unit testing and unit test plans are amongst the weakest areas of the software development process. This weakness is apparent in the quality and completeness of the unit test plans. The candidate PSDMs all rely on data drawn from these unit test plans, and therefore may be weakened by inconsistencies in the unit test plan data. The unit test plan milestone provides one of the weakest predictions of rank by defect potential.

Milestones dependent upon PSDMs that are more related to the process, such as defect categorization, and unit test plans, do not provide for accurate predictions of the actual number of defects. This potentially indicates one or more of the following:

* Better PSDMs are needed.
* The design artifacts need improved documentation of the design process.
* The software design process needs improvement.
* These milestones are not good points for the prediction of defect potential.

6.4 Minimal Distinct PSDM Prediction Model Results

Table 6.4.1 presents the results of applying the minimal distinct PSDM prediction model to the data from the experimental group of CSCs. Low RUSS values are available at all software design milestones except for the detailed design review, as-built design changes, unit test plans, and the code walk-through. The best software design milestone for predicting the number of defects, based upon the USS values, is at the as-built design using the design characteristic fan-out (FO).

Table 6.4.1 Summary of Minimal Distinct PSDM Prediction Model Results

Software Design Milestone	PSDMs Used	RSV	MSE	RUSS	USS	CSS
1	TR	0.745	2197.9	2.0	2110.1	616.9
2	TR	0.792	1785.8	2.0	10395.4	7666.5
3	AIARI	0.923	659.9	2.0	5613.2	944.3
4	FO	0.969	266.5	2.0	2686.5	2065.9
5	AIARI	0.939	521.3	10.0	4163.1	3646.4
6c	FO	**0.979**	**180.1**	**0.0**	**200.3**	**164.1**
6d	PCFO	0.914	740.4	30.0	19474.7	17313.6
7	TSTE	0.909	785.0	8.0	3595.8	3546.0
8	AIARI	0.715	2450.4	11.5	2864.0	2816.1

Note: Best values are in boldface

The results using the data from the experimental group of CSCs of the minimal distinct PSDM prediction model cannot be directly compared to the results of the "best" prediction equations in Section 6.3. Any comparisons must consider the R-squared value (RSV) and mean sum of squared error (MSE) from the multiple regression using the data from the test group of CSCs in Chapter Five.

Table 6.4.2 shows the percent change in the RSV and MSE that occurs when the prediction equations are selected based upon minimizing the number of distinct PSDMs over all software design milestones. Percent change is

calculated as the ratio of "best" prediction equation value minus the minimal distinct PSDM prediction equation value to the "best" prediction equation value.

Except for the code walk-through design milestone, selection of the minimal distinct PSDM prediction equations change the RSV by less than 10%. Selecting total requirements (TR) over allocated requirements (AR) at the requirements allocation milestone improves the RSV and MSE. The selection of AR over TR in the "best" prediction equations is due to AR performing better than TR when applied to the experimental group of CSCs in Section 6.2.1.

Table 6.4.2 Percent Changes in RSV and MSE in Selecting Minimal Distinct PSDM Prediction Equations Over Best Prediction Equations

Software Design Milestone	PSDMs Used		Percent Change in RSV	Percent Change in MSE
	Best Prediction Equation	Minimal Distinct PSDM Prediction Equation		
Requirements Allocation	AR	TR	2.534	-6.73
Preliminary Design	DR	TR	-6.236	34.00
Preliminary Design Review	AIARI	AIARI	0.000	0.000
Detailed Design	FO	FO	0.000	0.000
Detailed Design Review	AIT	AIARI	-3.353	116.58
As-Built Design (Design Characteristics)	FO CKP	FO	-1.947	965.68
As-Built Design (Changes in Design)	ADMO	PCFO	-7.112	434.97
Unit Test Plans	TSTE	TSTE	0.000	0.000
Code Walk-Through	AIND AIPCC	AIARI	-27.850	2339.31

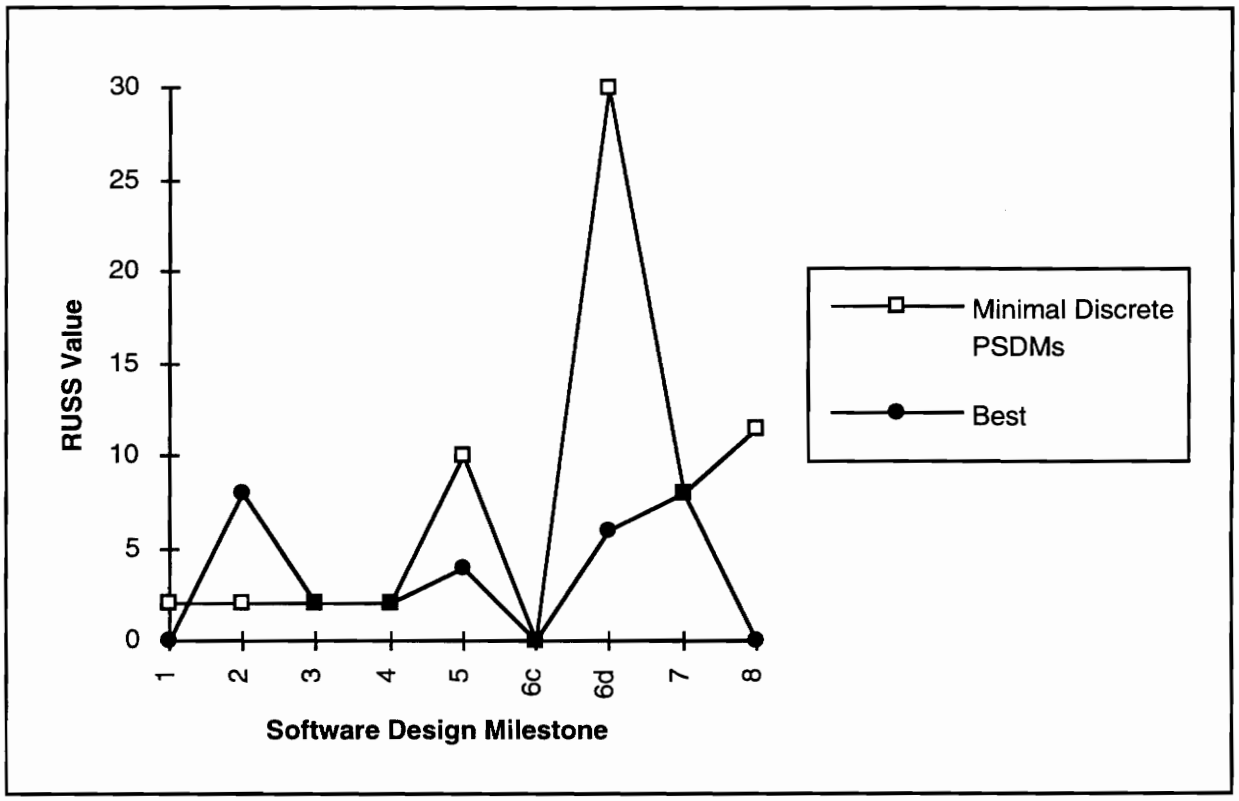


Figure 6.4 RUSS Values of Minimal Distinct PSDM and Best Prediction Equations for Each Software Design Milestone;

RUSS Value Comparison

Because the percent change in RSV values in selecting the minimal distinct PSDM prediction equations over the "best" prediction equations are low, comparisons of the accuracy between the predicted rankings (RUSS values) can be made. Figure 6.4 shows the RUSS values of both the minimal distinct PSDM and "best" prediction equations. RUSS values of zero indicate that the predicted rankings of the experimental group CSCs by defect potential are correct.

RUSS values are better for the minimal distinct PSDM prediction equations than the "best" prediction equations at the preliminary design milestone (2.0 for the minimal versus 8.0 for the "best"). The RUSS values for the minimal distinct PSDMs are worse (higher) than those of the "best" prediction equations at the requirements allocation, detailed design review, as-built design changes,

and code walk-through milestones. With the exception of the as-built design changes prediction equation, the minimal distinct PSDM prediction equations perform nearly as well as the "best" prediction equations in predicting CSC rank.

USS and CSS Values

Table 6.4.3 contains the percent change in the uncorrected sum of squared error and the corrected sum of squared error when the minimal distinct PSDM equations are selected over the "best" prediction equations. Except for milestones where the prediction equations are the same (i.e. preliminary design review, detailed design, and unit test plans), the predicted numbers of defects are less accurate for the minimal distinct PSDM prediction equations when applied to the experimental group of CSCs. This is indicated by positive percent change values in Table 6.5.3. Positive values indicate that the uncorrected sum of squared error (USS) and/or the corrected sum of squared error (CSS) increased. The amount of relative increase is given by the magnitude of the percent change value.

Table 6.4.3 Percent Changes in USS and CSS in Selecting Minimal Distinct PSDM Prediction Equations Over Best Prediction Equations

Software Design Milestone	PSDMs Used		Percent Change in USS	Percent Change in CSS
	Best Prediction Equation	Minimal Distinct PSDM Prediction Equation		
Requirements Allocation	AR	TR	158.6	302.3
Preliminary Design	DR	TR	51.3	52.3
Preliminary Design Review	AIARI	AIARI	0.0	0.0
Detailed Design	FO	FO	0.0	0.0
Detailed Design Review	AIT	AIARI	38.3	46.0
As-Built Design (Design Characteristics)	FO CKP	FO	76.9	73.1
As-Built Design (Changes in Design)	ADMO	PCFO	250.1	736.5
Unit Test Plans	TSTE	TSTE	0.0	0.0
Code Walk-Through	AIND AIPCC	AIARI	38.2	36.6

Summary of Minimal Distinct PSDM Results

The minimal distinct PSDM prediction equations performed well in predicting rank by defect potential when compared with the "best" prediction equations. Predictions of the number of defects are less accurate than those of the "best" prediction equations. The largest difference occurs at the as-built design milestone using the change in design PSDMs. At this point, USS increases 250% and CSS increases 737% when the minimal discrete PSDM model is used instead of the "best" prediction equation. Referring to Table 6.5.3, there is no design milestone where the minimal discrete PSDM model is more accurate (negative percent change) than the "best" prediction equations.

Because the "best" prediction equations included minimizing the number of PSDMs in their selection, many of the prediction equations in the two groups are the same. At these milestones (preliminary design review, detailed design,

and unit test plans) the minimal discrete PSDM model performs the same as the group of "best" prediction equations. With these shared prediction equations, the results in predicting the rank of the experimental group CSCs by defect potential are relatively good for the minimal discrete PSDM model.

Recalling that the driving motivation in creating a minimal discrete PSDM prediction model is to minimize collection training and effort by limiting the number of discrete PSDMs used. The good results in predicting the rankings of the experimental group of CSCs supports using the minimal discrete PSDM prediction model over the "best" prediction equations when time and effort are at a premium and relative rankings are desired.

7 Conclusions

Verification and validation is an important aspect of developing quality software products. Because of the deadlines and limitations imposed by project cost and schedule, testing cannot be exhaustive. If a project is behind schedule, testing efforts may be subject to further reduction in order to meet deadlines. Therefore, testers must maximize the effectiveness of their efforts through the selection and quality of tests. Targeting areas of the design and code identified as being prone to defects is a simple way of improving testing effectiveness.

7.1 Brief Overview of Research and Experimentation

This research concentrates on making the identification of problem areas possible. A candidate set of Predictive Software Design Measures (PSDMs) is developed to identify defect prone areas of the software design prior to formal testing (Chapter Three and Chapter Four). PSDMs are then selected for use in a model for predicting defect potential (Chapter Five).

The predictive model addresses four areas of concern in predictive measurement. These areas are: defect categorization, internal complexity, information flow, and changes in design. Eight points for prediction, called milestones, are defined in the prediction model based upon the software design process and the availability of design artifacts. PSDM data is calculated from these design artifacts.

Prediction equations are formed at each design milestone using the candidate PSDMs, data from the test group of CSCs, and multiple linear regression analysis (Chapter Five). The "best" prediction equations are selected at each milestone based upon the quality of the prediction equations, the areas of concern in predictive measurement, and the amount of data collection required.

The resulting set of selected prediction equations is evaluated by applying PSDM data from the second experimental group of CSCs (Chapter Six). The

predicted rankings and number of defects are compared with the actual rankings and number of defects. Single "best" prediction equations are selected at each design milestone based upon the comparisons between the actual values and the predicted values.

7.1.1 Revisiting the Major Goals

Brief Overview of Major Goals and Results

Recall from Chapter One of this document that the major goals of this research are reflected in the creation of the set of predictive metrics and their consequent use.

Creation:

* **Accurate and early predictions of defect proneness.**

  - Predicted rankings are correct or only slightly off (i.e. two adjacent ranks are switched) in five of the eight design milestones.

  - The most accurate milestone for prediction is the as-built design. This is not surprising due to the fact that, at this point, the design is fully mature. Moreover, the earlier software design phases have revealed design flaws or identified better alternatives.

  - The first design milestone, requirements allocation, milestone one of the two best prediction points for both rank and defect counts.

  - Predictions of limited accuracy are possible as early in the design process as the requirements allocation. The earlier prediction milestones show promise and can, in some cases, correctly predict the rankings of the experimental group of CSCs.

* **Addressing all phases of software design process.**

The creation and definition of the predictive model for the application of PSDMs addresses all phases of the software design process as well as the four areas of concern in predictive measurement.

* **Simple collection and interpretation.**

  - The selected equations from the first phase of the experiment and those prediction equations chosen as "best" all use a low number of PSDMs. This greatly simplifies the collection and interpretation of the PSDM data.

  - The PSDMs selected for use require only simple collections and calculations. Fan-Out and Cyclomatic Complexity require the most effort.

Deployment:

* **Model for test designers to target defect-prone areas of a software product.**

* **Model for designers to identify areas of the design to revisit and possibly redesign.**

  - Predicted rankings are accurate enough to enable testers and designers to target those CSCs that are most prone to defects.

  - The as-built design milestone is the best prediction points but occurs very late in the software design process. One concern is that changes to the software and test designs are more expensive to make at the as-built design than at earlier design milestones. An additional concern is

that the as-built design milestone may be too late in the software design process to improve the software design and/or testing efforts.

Minimal Distinct PSDM Prediction Model

Recall from Chapters Five and Six that selecting prediction equations using a minimal number of distinct PSDMs provides relatively good results in predicting the rank of the experimental group of CSCs by defect potential. The predicted defect counts are not as accurate as those of the "best" selected prediction equations. The major advantage of the minimal distinct PSDM prediction equations is that the number of different PSDMs is reduced so that collection, calculation, and interpretation efforts are less costly.

7.1.2 Revisiting the Minor Goals

Recall from Chapter One of this document that there are two minor goals of this research effort:

* Examining the relationship of size to defect proneness.
  - Performance of lines of PDL as a predictor of the number of defects.
  - Performance of "sizing only" predictors versus "best" predictors

* Collecting Data describing a developing software design.

Size Versus Defect Proneness: Performance of lines of PDL as a Predictor of the Number of Defects

The largest CSCs have the most defects. Largest here means the CSCs with the most lines of PDL, requirements, and CIs. The selected equations all use PSDMs with some dependency on the size of the CSC. The count of lines of PDL is only used as a normalization factor in one PSDM of one of the "best" equations. This supports the notion that lines of PDL or code is not, by itself, an

adequate predictor of software quality. Based upon the results of evaluating the selected equations in Chapter Six of this document, functional quantification appears to provide more accurate predictions of defect potential than lines of PDL. Another alternative, supported by the results of this research, is to use lines of PDL with measures relating to the software's functionality such as Fan-Out and Cyclomatic Complexity. A combination of these measurements is the best prediction equation found in this research when applied at the as-built design milestone.

Size Versus Defect Proneness: "Sizing Only" Predictions

The "sizing only" predictions measure the total number of requirements (TR), lines of PDL (PDL) and the number of unit tests designed (TTES). Because the changes to these PSDMs at the detailed design review and code walk-through design milestones are not documented in the available design artifacts, these design milestones are omitted from consideration.

Table 7.1.2 summarizes the selected "sizing only" prediction equations. The R-squared values (RSVs) and means squared error values (MSEs) reflect how well the prediction equations formed from the PSDM data of the test group of CSCs fit the actual number of defects detected. The rank sum of squared error (RUSS) value shows the error in the rankings of the experimental group of CSCs by the predicted number of defects. The error in these predicted numbers of defects is shown by the uncorrected sum of squared error (USS) and corrected sum of squared error (CSS) values.

Table 7.1.2.1 Results of Sizing PSDM Predictions by Design Milestone

Design Milestone	PSDMs Used	RSV	MSE	RUSS	USS	CSS
Requirements Allocation	TR	0.745	2197.9	**2.0**	**2110.1**	**616.9**
Preliminary Design	TR	0.792	1785.8	**2.0**	10395.4	7666.5
Preliminary Design Review	TR	0.902	843.6	**2.0**	2388.5	2296.3
Detailed Design	PDL	0.939	528.4	6.0	2956.3	2784.8
Detailed Design Review	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*
As-Built Design(Design Changes)	ADPDL	0.967	280.8	12.0	6309.9	5547.4
As-Built Design (Characteristics)	PDL	**0.995**	**41.3**	12.0	2524.4	2488.5
Unit Test Plans	TTES	0.849	1302.7	14.0	5355.6	3315.4
Code Walk-Through	*NA*	*NA*	*NA*	*NA*	*NA*	*NA*

Note: *NA* means not applicable and the best values are in boldface.

From Table 7.1.2.1, the best "sizing only" predictor of rank is the total number of requirements (TR) as shown by its low RUSS value of 2.0. The worst "sizing only" predictor of rank is the number of unit tests designed (TTES) with a RUSS value of 14.0. The predicted rankings of the "sizing only" PSDMs are compared to those of the "best" prediction equations at the applicable milestones in Figure 7.1.2.1. This figure shows that the "sizing only" predictors perform relatively well at the early design milestones when total requirements (TR) is used to make predictions.[1] The predictions of rank by the "sizing only" PSDMs is worse at the later design milestones where lines of PDL (PDL) and the number of unit tests designed (TTES) PSDMs are used as predictors.[2]

[1] TR is used at the requirements allocation (milestone 1), the preliminary design (milestone 2), and the preliminary design review (milestone 3) design milestones.

[2] PDL is used at the detailed design (milestone 4) and the as-built design (milestone 6) milestones. TTES is used at the unit test plan (milestone 7) milestone.

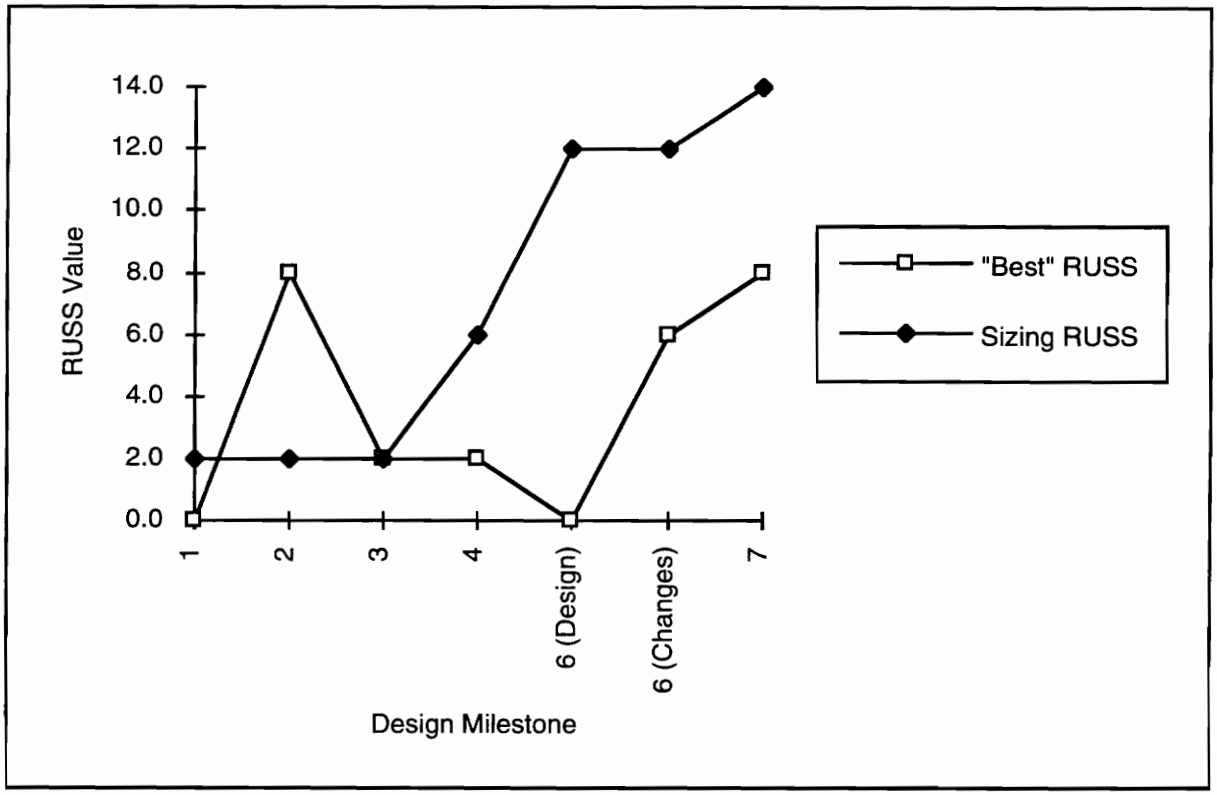


Figure 7.1.2 Summary of "Best" and "Sizing Only" RUSS Values

A potential reason for the poor performance of PDL and TTES versus TR in the "sizing only" predictions can be found in the software design process as presented in Keller and Nance (1991, pp. 10-12). Total requirements (TR) is the starting point for the development process. The development process or model is a set of transformations intended to develop a system implementation (PDL or code) from the original abstraction (requirements). Each transformation (software design phase) adds refinement , alternatives, decisions, and/or corrections to the descriptive context. This descriptive context is the complete set of design artifacts from the requirements allocation to the as-built design. At the early phases of the software development, total requirements characterizes a major portion of the available design artifacts. Because the amount of design artifacts and their level of refinement increases with each software design phase, measures such as PDL and unit test counts characterize a much smaller portion of the available design artifacts. Therefore the decreased accuracy of defect

predictions at later phases may be attributed to decreased characterization of the design artifacts.

Table 7.1.2.2 USS and CSS Values for "Best" and "Sizing Only" Prediction Equations

Design Milestone	Uncorrected Sum of Squared Error (USS)		Corrected Sum of Squared Error (CSS)	
	"Best"	"Sizing Only"	"Best"	"Sizing Only"
Requirements Allocation	815.8	2110.1	153.3	616.9
Preliminary Design	21334.1	10395.4	16070.5	7666.5
Preliminary Design Review	5613.2	2388.5	944.3	2296.3
Detailed Design	2686.5	2956.3	2065.9	2784.8
Detailed Design Review	*NA*	*NA*	*NA*	*NA*
As-Built Design (Design Characteristics)	866.2	2524.4	610.8	2488.5
As-Built Design (Changes in Design)	5562.5	6309.9	2069.8	5547.4
Unit Test Plans	3595.8	5355.6	3546.0	3315.4
Code Walk-Throughs	*NA*	*NA*	*NA*	*NA*

Note: *NA* is not applicable. Milestone numbers are in boldface.

Table 7.1.2.2 summarizes the accuracy of the "best" and "sizing only" prediction equations in predicting the actual number of defects in the experimental group of CSCs. "Sizing only" prediction equation is more accurate than the "best" prediction equation at preliminary design as shown by the lower uncorrected sum of squared error (USS) and corrected sum of squared error (CSS). At the preliminary design review, the "best" prediction equation is less accurate overall (USS value of 5613.2 versus 2388.5) but has a lower variability in the accuracy of the predictions (CSS of 944.3 versus 2296.3) than the "sizing only" prediction equation. The "sizing only" prediction equation at the unit test plans milestone is less accurate overall (USS of 5355.6 versus 3595.6) but has a slightly lower variability in the accuracy of the predictions (CSS of 3315.4 versus 3546.0) than the "best" prediction equations. Except for the preliminary design and preliminary design review milestones, the "sizing only" prediction equations

are less accurate (in terms of USS values) than the "best" prediction equations in predicting the number of defects in the experimental group of CSCs.

Collecting Data Describing a Developing Software Design

The data collected for calculating and evaluating the various PSDMs at the design milestones describes the development of the software design. In order to maintain consistent data collection, the same PSDM data is collected from both the test and experimental groups. Should another PSDM become available, or another predictive model develop which uses the existing PSDM data, this data is available for both the test and experimental CSCs.

7.2 Future Research

The major goal for future research is improving the predictions. There are three areas for further investigation. These areas address improving the predictions by improving the design artifacts, considering other similar software designs, and applying prediction equations to a developing software design.

Improving the Design Artifacts

Predictions are dependent upon the available design artifacts. If these artifacts are of a consistent format and high quality, then the predictions can improve. In this research, the design artifacts available at some design milestones is not in a consistent format. This is especially evident in the unit test plans of the project used in this research. The unit test plans are the least consistent of the design artifacts. Predictions of CSC rank by defect potential are among the worst at the unit test plan milestone.

Considering Similar Software Designs

Another potential means of improving the predictions is to analyze the development of another software design. This design should have similar functionality and development process. The coefficient values produced in the multiple regression analysis are dependent upon the PSDM data of the test set of CSCs. One can assume that these values and the quality of the calculated predictions change when different CSCs or a different software project is the PSDM data source.

Application to a Developing Software Design

The effect and value of in-process defect predictions with respect to the resulting quality of a software design are not examined in this research. The set of PSDMs and prediction equations selected as "best" in this research could also be applied to another software design as it develops. This application could be used to evaluate the value of the predictions for test and software designers.

References

Albrecht, A. J., & Gaffney, J. E. Jr. (1983). Software function, source lines of code, and development effort prediction: A software science. IEEE Transactions on Software Engineering, SE-9(6), 639-648.

Allen, D. M. (1971). Mean square error of prediction as a criterion for selecting variables. Technometrics, 13(3).

Basili, V. R., & Perricone, B. T. (1984). Software errors and complexity: An empirical investigation. Communications of the ACM, 27(1), 42-52.

Beizer, B. (1984). Software testing techniques (2nd ed.). New York, NY: Van Nostrand Reinhold Co., Inc.

Beizer, B. (1990). Software system testing and quality assurance. New York, NY: Van Nostrand Reinhold Co., Inc.

Chillarege, R., Bhandari, I. S., Chaar, J. K., Halliday, M. J., Moebus, D. S., Ray, B. K., & Wong, M-Y. (1992). Orthogonal defect classification - A concept for in-process measurements. IEEE Transactions on Software Engineering, 18(11), 943-955.

Card, D. N., & Glass, R. L. (1990). Measuring software design quality. Englewood Cliffs, NJ: Prentice-Hall.

Deutsch, M. S., & Willis, R. R. (1988). Software quality engineering: A total technical and management approach. Englewood Cliffs, NJ: Prentice-Hall.

Goodman, P. (1993). Practical implementation of software metrics -(International software quality assurance series). London: McGraw-Hill.

Grady, R. B. (1987). Measuring and managing software maintenance. IEEE Software, 4(9), 35-45.

Grady, R. B., & Caswell, D. L. (1987). Software metrics: Establishing Acompany-wide program. Englewood Cliffs, NJ: Prentice-Hall.

Henry, S. & Kafura, D. (1981). Software structure metrics - based on information flow. IEEE Transactions on Software Engineering, 7(5), 510-518.

Henry, S., & Selig, C. (1987). A metric tool for predicting source code quality from a PDL design. Blacksburg, Virginia: Virginia Tech Computer Science Department.

Henry, S., & Selig, C. (1990). Predicting source code complexity at the design stage. IEEE Software, 7(2), 36-44.

Inglls, J. (1985). Standard software quality metrics. AT&T Technical Journal, 65(2), 113-117.

Keller, B. J., & Nance, R. E (1991). Abstraction Refinement: A model of software evolution. Technical Report SRC-91-006. Blacksburg, Virginia: Systems Research Center, VPI&SU.

Kelly, J. C., Sherif, J. S., & Hops, J. (1992). An analysis of defect densities found during software inspections. Journal of Systems Software, 17(2), 111-117.

Khoshgoftaar, T. M., Munson, J. C., Bhattacharya, B. B., & Richardson, G. D. (1992). Predictive modeling techniques of software quality from software measures. IEEE Transactions on Software Engineering, 18(11), 979-986.

Kvalseth, T. O. (1985). A cautionary note about R^2. The American Statistician, 39(4), 279-286.

Longstreet, D. H. (Ed.). (1990). Software maintenance and computers. Los Alamitos, CA: IEEE Computer Society Press.

Low, G. C., & Jeffery, D. R. (1992). Function points in the estimation and evaluation of the software process. IEEE Transactions on Software Engineering, 16(11), 64-71.

McCabe, T., & Butler, C. (1989). Design complexity measurement and testing. Communications of the ACM, 32(12), 1415-1425.

McClure, C. (1992). The three Rs of software automation : Re-engineerirng, repository, reusability. Englewood Cliffs, NJ: Prentice-Hall.

Milton, J. S., & Arnold, J. C. (1986). Probability and statistics in the engineering and computer sciences. New York, NY: McGraw-Hill, Inc.

Mukhopadhyay, T., & Kekre, S. (1992). Software effort models for early estimation of process control applications. IEEE Transactions on Software Engineering, 18(10), 915- 924.

Musa J. D., Iannino, A., and Okumuto, K. (1987). Software reliability: Measurement, prediction, application. New York, NY: McGraw-Hill.

Nance, R. E., Arthur, J. D., Love, R., & Dorsey, E. V. (1993). Software quality measurement: Validation of a foundational approach (Final Report, Year Three). Technical Report SRC-93-005. Blacksburg, Virginia: Systems Research Center, VPI&SU.

Perlis, Al., Sayward, F., & Shaw, M. (Eds.). (1981). Software metrics: An analysis and evaluation. Cambridge, MA: MIT Press.

Plant, R. T. (1992). Expert system development and testing: A knowledge engineer's perspective. Journal of Systems Software, 19, 141-146.

Rosson, C. V. (1988). Management indicators: Assessing product reliability and maintainability. Technical Report SRC-88-011. Blacksburg, Virginia: Systems Research Center, VPI&SU.

Shen, V. Y., Yu, T., Thebaut, S. M., & Paulson, L. R. (1985). Identifying error-prone software - An empirical study. IEEE Transactions on Software Engineering, 11(4), 317-323.

Seddio, C. (1993). Integrated test metrics within a software engineering measurement program at Eastman Kodak Company: A follow-up case study. Journal of Systems Software, 20(3), 227-235.

Sommerville, I. (1992). Software engineering (4th. ed.). New York, NY: Addison-Wesley Publishing.

Statland, N. (1986). Controlling software development. New York, NY: John Wiley & Sons Inc.

Voas, J. M., & Miller, K. W. (1993). Semantic metrics for software testability Journal of Systems Software, 20(3), 207-216.

Voas, J. M., Payne, J. E., & Miller, K. W. (1993). Designing programs that are less likely to hide faults. Journal of Systems Software, 20(1), 93-100.

Younger, M. S. (1979). Handbook for linear regression. Belmont, CA: Wadsworth Inc.

Appendix A Discussion of Excluded Measures

This appendix contains discussions of Albrecht's Function Point analysis and Halstead's Software Science. These are two well known software measurement methods that are excluded from this research.

A.1 Function Point Analysis

What is Function Point Analysis?

Albrecht's function point analysis is a method for determining a size measure for a software program based upon the number and complexity of the functions it performs. Albrecht and Gaffney (1983, p. 639) contend that function point analysis addresses all the functionality of a software program and give three reasons for its use. The first reason is that function point counts can be developed with relative ease early in the design process and provide a more direct relation to program requirements than estimates of lines of code. Secondly, the data required for the calculation of function points is usually available at the start of software design or very soon after. The third reason is that function point values can be used as a measure of productivity.

The calculation of function points requires each function and/or requirement of the program be separated into one of 5 functional areas[1]. These areas are: external inputs, external outputs, external inquires, and master files. Each functional area is assigned an initial weighting based upon previous experience with function points. The weighting of each program function and/or requirement is then adjusted by the analyst's estimation of its complexity. Low and Jeffery (1992, p.71) report variations of function point values within 30% within an organization.

---

[1] Please refer to Albrecht and Gaffney (1983) or Low and Jeffery (1992) for a more detailed explanation of function point calculation.

Why Exclude Function Point Analysis?

There are two major reasons for the exclusion of function point analysis from this research. The first reason is that for the purposes of this research, function point values do not provide enough detail for use in targeting and/or redesigning areas of a design identified as defect-prone. The second reason is that because detailed identifications are desired, the use of function point analysis introduces statistical dependence with the more specific measures.

The intent of this research is to provide in-process feedback identifying defect-prone areas of a developing software design. Testers and designers should be provided with a clear indication of what characteristic(s) support defect proneness. Detailed design characteristics for a functional unit, such as the number of components it contains or the number of inputs/outputs, are desirable. Such measures identify specific design characteristics to be targeted with test cases and revisiting/redesigning. To derive similar targeting information from function point values, the details of the function point value calculation must be recreated or saved and re-analyzed. This appears to be unnecessary labor.

Because function point analysis is composed of some of these design measures and characteristics in the calculation of function point values, its use potentially introduces statistical dependence. The weighting used in function point calculation is analogous to the determination of coefficients in a multi-variate linear regression analysis. Employing function point values in identifying defect-prone areas precludes the use of any predictive metrics that address areas used in function point analysis: inputs, outputs, inquires, and master files. Indirect quantification of functionality, such as the number of components that make up a functional unit, must also be avoided. To eliminate the potential for statistical dependence and provide detailed in-process feedback, function point analysis is not used in this research.

It is important to note that the theoretical basis for function point analysis can be addressed without employing Albrecht's methods and functions.

Quantification of functionality and complexity are possible prior to detailed design through analysis of requirements allocated, derived, and constraints for each functional area of the design. These values are not exposed to the variation in calculated function point values that can be attributed to the analyst's perception of complexity. Once detailed design begins, predictive metrics which are directly traceable to the developing design are available and desirable as in-process feedback for both tester and designer. Use of function point values in addition to directly traceable predictive metrics introduces potential statistical dependence and has limited value in identifying specific design characteristics.

A.2 Halstead's Software Science

What is Halstead's Software Science?

Halstead's Software Science is a collection of complexity measures based upon calculations and counts of program keywords and data variables (McClure, 1992, p. 50). Operators are defined to be reserved language words and arithmetic operators. Operands are defined as data variables and constants in the program.

According to Grady and Caswell (1987, p.197), Halstead believed that the ease of reading and writing software is related to the richness of the vocabulary used in terms of unique operators and operands as well as the number of times each variable is used. Writing and maintaining an algorithm becomes more difficult as the repertoire of operators and operands increases. The greater the average number of times each operator is used, the more difficult it is to remember the current state of the variable. Card (1990, p. 24) further suggests similar reasoning that any programming task consists of selecting and arranging a finite number of operands and operators.

Halstead's Software Science defines a basic set of formulas for program length, estimated program length, volume, effort, and purity ratio. Program length, estimated program length, and volume are calculated from the set

operand and operator counts. This set consists of the counts of total operands, total operators, total unique operators, and unique operands. Effort is the ratio of the volume measure to a constant reflecting the "language level" (e.g. assembler is 0.01 and COBOL is 0.1). The purity ratio is the ratio of the estimated program length value to the program length value.[2]

The value of Halstead's Software Science measures is a subject for debate. McClure supports the use of Software Science measures in evaluating software complexity while Card and Glass criticize the Software Science measures.

McClure (1992, p. 52-53) states three advantages of Halstead's Software Science. The first advantage is that the measures and calculations are simple to automate on code. This automation can also be effectively implemented for formal PDL, as shown by Henry and Selig (1987). The second advantage is that the Software Science measures are applicable to any programming language and are language sensitive[3]. The third advantage of Halstead's Software Science is that it is supported by many studies performed in the software development industry.

Card and Glass (1990, p.26) concede that experimental data and results support the value of Halstead's Software Science despite theoretical objections. They cite theoretical objections based upon the methodology used, incorrect use of human memory models, questionable derivation of equations, and the experimental methods used in verification. Card and Glass also criticize Halstead's Software Science for its language sensitivity which precludes its use in comparing projects in different languages. They suggest that the Software Science measures are more important because of the approach they represent rather than their usefulness in practice.

---

[2] For more information, please refer to Card (1990) and McClure (1992).

[3] This language sensitivity is considered a disadvantage by those concerned with comparisons between systems implemented in different languages.

Why Exclude Halstead's Software Science?

Halstead's Software Science is excluded from this research due to implementation factors. This research derives predictive metrics from design artifacts produced prior to coding and does not address the code itself. The PDL produced in the design process is informal and therefore the Software Science measurements are neither accurately nor simply automated. Manual calculation introduces additional variation due to the analyst's interpretation of the informal PDL and is very expensive in terms of collection time.

Card and Glass (1990, p. 27) suggest that Halstead's Software Science, as it currently exists, can safely be excluded for two reasons. The first reason is that it seems to represent the wrong level of detail for most measurement applications because software developers do not think in terms of individual operators and operands. The second reason is that, in general, Halstead's measures do not perform any better as complexity estimates than simpler measures, such as lines of code.

Appendix B Data Collection for Predictive Software Design Measures

This appendix documents the data collection necessary for the candidate predictive software design measures (PSDMs). This data collection is presented as the required measures for each design artifact.

B.1 Required Data Collection

Data in the form of counts must be extracted from the design artifacts so that candidate predictive software design measures can be calculated and interpreted. The design artifacts analyzed in this research are presented in section 3.4.2 of this document. Recall the artifacts:

Design Artifacts Used (in the order they are produced):

1. Requirements allocation.
2. Functional capabilities section.
3. Preliminary Design.
4. Information Flow Diagrams.
5. Preliminary design review minutes/mark-ups.
6. Post-review preliminary design.
7. Detailed design documents.
8. Detailed design review minutes/mark-ups.
9. As-built design documents.
10. Unit Test Plans.
11. Code walk-through minutes/mark-ups.

The data required from each design artifact is presented in the following sections. This information is valuable for any future attempts to tailor this research to other software development projects.

A Note Regarding the Categorization of Action Items:

Action Items are used to address defect categorization as potential predictors of defect proneness. In order to draw more information from the Action Items, categories are used. Action Items are categorized by the responsible person and the required action.

When Action Items are documented in the minutes from reviews or walk-throughs, they are assigned to a person or group. The following categories for the assigned person are used:

1) Designer.
2) Design Group.
3) Development Group.
4) Customer.

The designer is the person whose work is being reviewed. The design group refers to any other designers working on the same project at the same location. Components for the software system analyzed in this research are developed at various locations. The development group refers to Action Items assigned to designers at other locations or other members of the development group (e.g. testers, configuration control, hardware developers, and project management). Action Items assigned to the customer are those which cannot be resolved by the developer. These are usually due to problems with the project requirements.

The Action Items are also categorized by the required action. This classification is used to address the differing levels of effort needed to resolve an Action Item. The required action categories are:

a) Simple mark-up or typographical correction.
b) Simple correction to design or trivial redesign.
c) Non-trivial redesign required.
d) Investigate problem and possibly redesign.
e) Correction to unit test plan.

The 'a' and 'b' required action categories are straight-forward corrections requiring little or no effort. Non-trivial redesigns ('c') address problems with the design that require more changes than can be simply marked up during the review. The Action Items categorized by 'd' are those in which a potential problem is found requiring investigation outside of the review. These investigations may cause a redesign, minor changes, or have no effect on the design. The final category ('e') is used with categories 'a' through 'd' as a way of identifying Action Items relating to unit test plans.

The two forms of Action Item categorization provide a quantification of the severity of an action item. Action Items assigned to the designer have a lower potential impact on the design and project than those assigned to the customer. Simple markups are less severe than problems requiring investigation and redesign.

B.2 Requirements Allocation

Required Data:
- Allocated requirement counts.
- Derived requirement counts.
- Constraint counts.

The requirements allocation design artifact includes sections listing allocated requirements, derived requirements, and constraints. Requirement and constraint counts are the numbers of separate, numbered statements.

B.3 Functional Capabilities Section

Required Data:
- Count of functional capabilities.

The functional capabilities section is a listing of the functions to be performed. Each function is counted once.

B.4 Preliminary Design

Required Data:
- Allocated requirement counts.
- Derived requirement counts.
- Constraint counts.
- Count of number of sentences in description of execution.

The presentation of the requirements in the preliminary design is in the same format as presented in the requirements allocation design artifact. The same methods for counting allocated requirements, derived requirements, and

constraints are used. The preliminary design also includes a brief description of the functional area's execution. This description is quantified by counting the number of sentences contained in the description.

B.5 Information Flow Diagrams

Required Data:
- Node counts.
- Counts of parameters passed to CSC.
- Counts of parameters passed from CSC.

Information flow diagrams are a pictorial representation of the control and data flows between a functional area and other areas of the design. Node counts are the number of functional areas present in the diagram. Parameters are marked with arrows pointing towards or away from nodes. The count of parameters passed to/from a CSC is the number of parameters marked with arrows away from/towards that CSC's node.

B.6 Preliminary Design Review Minutes/Mark-Ups

Required Data:
- Action Item (AI) categorization:
  i. AI counts by responsible person.
  ii. AI counts by severity of action required.

Action Item categorizations are derived from the documented action items using the categorization scheme presented previously in section B.2. Each documented Action Item is classified by the responsible person and the action required.

B.7 Post-Review Preliminary Design

Required Data:
- Allocated requirement counts.
- Derived requirement counts.
- Constraint counts.
- Count of number of sentences in description of execution.

The presentation of the requirements in the post-review preliminary design is in the same format as presented in the requirements allocation design artifact. The same methods for counting allocated requirements, derived requirements, and constraints are used. The preliminary design also includes a brief description of the functional area's execution. This description is quantified by counting the number of sentences it contains.

B.8 Detailed Design Documents

Required Data:

- Count of CIs.
- Counts of modules per CI.
- Per module counts of:
  - Data structures defined (header of each module).
  - Data types defined (header of each module).
  - Lines of PDL.
  - Loops.
  - Loop exit statements.
  - Extra module exit points.
  - Exceptions Raised.
  - Exception Handlers.
  - Decisions.
  - Decision statements.

- Per CI information flow counts of:
  - Input arguments defined (header of each module).
  - Output arguments defined (header of each module).
  - In/out arguments defined (header of each module).

  - Data structures read from.
  - Data structures written to.

  - Calls to other CSCs.
  - Calls to other CIs in same CSC.
  - Calls from other CSCs in set of those analyzed.
  - Maximum calls per CI.
  - Maximum calls per module.

The above data is drawn from the detailed design sections of the Software Development Folders (SDFs) and is also available in detailed design documents produced for the customer.

B.9 Detailed Design Review Minutes/Mark-Ups

Required Data:
- Action Item categorization:
  i. AI counts by responsible person.
  ii. AI counts by severity of action required.

Action Item categorizations are derived from the documented action items using the categorization scheme previously presented in Section B.2. Each documented Action Item is classified by the responsible person and the action required.

B.10 As-Built Design Documents

Required Data:

Please Note: This is the same as presented in Section B.2.7 Detailed Design Documents.

B.11 Unit Test Plans

Required Data:
- Count tests designed.
- Count of verify sentences/steps.
- Count of total test steps.

Unit test plans designed for the project used in this research are not in a standard format. Designers of the functional area (CSC) design the unit tests. By comparing the variety of unit test plan formats, measures to quantify the amount and quality of testing are selected. These measures are counts of the number of tests, verify sentences/steps, and total test steps. Verify steps are either presented as separately numbered statements, or as concatenated statements in a block of text. Each separate statement is counted as a verify step. The same is true for counting total test steps.

B.12 Code Walk-Through Minutes/Mark-Ups

Required Data:
- Action Item (AI) categorization:
    i. AI counts by responsible person.
    ii. AI counts by severity of action required.

Action Item categorizations are derived from the documented Action Items using the categorization scheme presented previously in Section B.2. Each documented Action Item is Action Item categorizations are derived from the documented Action Items using the categorization scheme previously presented in Section B.2. Each documented Action Item is classified by the responsible person and the action required.

Appendix C Candidate Predictive Software Design Measures

This appendix documents the candidate predictive software design measures. Candidate measures are presented as they relate to the predictive model presented in Section Four of this document. This means the required design artifacts and the predictive software design measures are presented, by areas of concern, for each milestone. Also included in this appendix are the logical groupings and relationships of PSDMs. These groupings and relationships determine valid PSDM combinations for use in predictive equations.

C.1 Candidate Predictive Software Design Measures

This section presents the predictive software design measures that are initially considered[1] for the software project discussed in section 3.2.4 of this document. Included in this presentation are the following:

* The design milestones for application.
* The required design artifacts.
* The addressed areas of concern in predictive measurement.
* The logical groupings and relationships between PSDMs.

Milestones[2] for the Application of Predictive Measures:

1. Requirements Allocation.
2. Preliminary Design.
3. Preliminary Design Review.
4. Detailed Design.
5. Detailed Design Review.
6. As Built Design.
7. Unit Test Plans.
8. Code Walk-Throughs.

---

[1] Please refer to section 5 for the use of candidate predictive software design measures.
[2] Please refer to section 3.3.1.2 for more information on milestones.

Areas of Concern[3] in the Prediction of Defect-Proneness:

1. Internal Complexity.
2. Information Flow.
3. Between Design Phase Changes.
4. Defect Categorization.

C.2 Requirements Allocation Milestone

What Design Artifacts are Analyzed?

1. Requirements allocation.
2. Functional capabilities section.

Measures for:

Internal Complexity

None.

Information Flow

None.

Changes in Design

AR - Allocated requirement counts.
CO - Constraint counts.
DR - Derived requirement counts.
FC - Total Capabilities (Functional Capabilities).
TR - Total Requirements (sum of allocated, derived, and constraints).

Defect Categorization

None.

Logical Groupings and Relationships:

TR or (AR DR CO).

CO cannot be selected alone.

---

[3] Please refer to section 3.4 for more information on areas of concern for predictive software design measures.

C.3 Preliminary Design

What Design Artifacts are Analyzed?

1. Requirements allocation and description.
2. Preliminary design.
3. Information flow diagrams.

Measures for:

Internal Complexity
None.

Information Flow
NOD - Node counts.
PTO - Counts of parameters passed to CSC.
PFR - Counts of parameters passed from CSC.
PAT - Total Parameters passed to or from CSC.

Changes in Design
(Allocation of requirements in preliminary design)
TR - Total Requirements in preliminary design.
AR - Requirements allocated in preliminary design.
DR - Requirements derived in preliminary design.
CO - Requirements Constraints in preliminary design.
DES - Lines of description in preliminary design.

(Change in total requirement allocation)
ADR - Absolute value of difference in total requirements.
PCR - Percent change in total requirements.

Defect Categorization
None.

Logical Groupings and Relationships

TR or (AR DR CO).

PAT or (PTO PFR).

CO cannot be alone.

C.4 Preliminary Design Review

This milestone's design artifacts include review Action Items and post-review preliminary design.

What Design Artifacts are Analyzed?

1. Preliminary design.
2. Preliminary design review minutes/ mark-ups.
3. Post-review preliminary design.

Measures for:

Internal Complexity

None.

Information Flow

None.

Changes in Design

(Post-review preliminary design)

TR - Total requirement counts.
AR - Allocated requirement counts.
DR - Derived requirement counts.
CO - Constraint counts.
DE - Count of number of sentences in description of execution.

(Change in total requirement allocation due to preliminary design review)

DA23TR - Absolute value of difference in total requirements.
PC23TR - Percent change in total requirements.

Defect Categorization

TAI - Total Action Items.
AIR1 - Designer Action Items.
AIARI - Count of Action Items to re-design or investigate.
AIND - Count of Action Items for persons or groups other than designer.
TAPR - Total Action Items per requirement of preliminary design (prior to review).
AIRIPR - Count of Action Items to re-design or investigate per requirement of preliminary design (prior to review).

A1PR - Count of Action Items for designer per requirement of preliminary design (prior to review).
AINDPR - Count of Action Items for persons or groups other than designer per requirement of preliminary design (prior to review).

Logical Groupings and Relationships:

TR or (AR DR CO).

DE cannot be alone.

DA23TR or PC23TR.

TAI or TAIPR.

TAI or (AIND AIR1).

AIND or AINDPR.

AIARI or AIRIPR.

C.5 Detailed Design

What Design Artifacts are Analyzed?

1. Post-review preliminary design.
2. Detailed design documents.

Measures for:

Internal Complexity

PDL - Lines of PDL.
CI - Number of CIs.
MO - Number of Modules.
C - Total Cyclomatic Complexity (number of decisions +1).
CCI - Total Cyclomatic Complexity per CI.
CMO - Total Cyclomatic Complexity per module.
CKP - Total Cyclomatic Complexity per thousand lines of PDL.
CGT - Modules with Cyclomatic Complexity greater than ten.
LO - Loops.
EX - Exception handlers and exceptions raised.
UP - Unconventional programming elements (GOTOs, loop exits, extra procedure exits).

Information Flow *(By CIs and summed (procedural) for CSC)*

FI - Limited enhanced fan-in (Sum of parameters in, data reads, and calls into).

FO - Limited enhanced Fan-out (Sum of calls out, parameters out, and data writes).

IFP - Information Flow Metric with lines of PDL.

IF - Information Flow Metric without lines of PDL.

SC - Structural complexity (for a CSC by CIs is: Sum of Fan-Out(CI)^2).

DC - Data complexity.

Changes in Design

CIPRE - Number of CIs designed per requirement.

MOPRE - Number of modules designed per requirement.

PDLPRE - Lines of PDL per requirement.

Defect Categorization

None.

Logical Groupings and Relationships:

PDL or PDLPRE.

CI or CIPRE.

MO or MOPRE.

C or CCI or CMO or CKP.

(PDL and C) or CKP.

(CI and C) or CCI.

(MO and C) or CMO.

(FI and FO) or IFP.

FI FO or IFP.

IF or IFP.

PDL or IFP.

C.6 Detailed Design Review

What Design Artifacts are Analyzed?

1. Detailed design review Action Items and classifications.

Measures for:

Internal Complexity

None.

Information Flow

None.

Changes in Design

None (deferred to as-built).

Defect Categorization

AIT - Total Action Items.
APC - Action Items per CI.
APP - Action Items per LOPDL.
AIRI - Action Items requiring redesign/ revisiting.
ARIPP - Action Items requiring redesign/ revisiting per LOPDL.
ARIPC - Action Items requiring redesign/ revisiting per CI.
TNDA - Action Items assigned to persons other than the designer.
TNDAPC - Action Items assigned to persons other than the designer per CI.
TNDAPP - Action Items assigned to persons other than the designer per line of PDL.

Logical Groupings and Relationships

AIT or APC or APP.

AIRI or ARIPP or ARIPC.

TNDA or TNDAPC or TNDPP.

C.7 As-Built Design

What Design Artifacts are Analyzed?

1. Detailed design documents.
2. As-built design documents.

Measures for:

Internal Complexity

- PDL - Lines of PDL.
- CI - Number of CIs.
- MO - Number of Modules.
- C - Total Cyclomatic Complexity (number of decisions +1).
- CCI - Total Cyclomatic Complexity per CI.
- CMO - Total Cyclomatic Complexity per module.
- CKP - Total Cyclomatic Complexity per thousand lines of PDL.
- CGT - Modules with Cyclomatic Complexity greater than ten.
- LO - Loops.
- EX - Exception handlers and exceptions raised.
- UP - Unconventional programming elements (GOTOs, loop exits, extra procedure exits).

Information Flow *(By CIs and summed (procedural) for CSC)*

- FI - Limited enhanced fan-in (Sum of parameters in, data reads, and calls into).
- FO - Limited enhanced Fan-out (Sum of calls out, parameters out, and data writes).
- IFP - Information Flow Metric with lines of PDL.
- IF - Information Flow Metric without lines of PDL.
- SC - Structural complexity (for a CSC by CIs is: Sum of Fan-Out(CI)^2).
- DC - Data complexity.

Changes in Design
AD = Absolute Difference from detailed design to as-built design milestones.
PC = Percent Change from detailed design to as-built design milestones.

ADPDL or PCPDL - Lines of PDL.
ADCI or PCCI - Number of CIs.
ADMO or PCMO - Number of Modules.
ADC or PCC - Cyclomatic complexity.
ADFI or PCFI - Fan-In.
ADFO or PCFO - Fan-Out.
ADIFP or PCIFP - Information Flow Metric with PDL.
ADIF or PCIF - Information Flow Metric without PDL.
ADSC or PCSC - Structural complexity.
ADDC or PCDC - Data complexity.
ADEX - Exception handlers and exceptions raised.
ADUP - Unconventional programming elements.
ADLO - Loops.
ADCGT - Modules with Cyclomatic Complexity greater than ten.

Defect Categorization
None.

Logical Groupings and Relationships

C or CCI or CMO or CKP.

(PDL and C) or CKP.

(CI and C) or CCI.

(MO and C) or CMO.

(FI and FO) or IFP.

IF or IFP.

PDL or IFP.

AD*XXX* or PC*XXX*. (i.e. The absolute difference and percent change of a given design characteristic cannot both be used in a predictive equation.)

C.8 Unit Test Plans

What Design Artifacts are Analyzed?

1. Unit test plans.

Measures for:

Internal Complexity

TTES Number of tests.
TSTE - Number of test steps.
TVES - Lines of verify / expected results.
TPCC - Number of tests versus Cyclomatic Complexity.
SPCC - Number of test steps versus Cyclomatic Complexity.
VPCC - Lines of verify / expected results versus Cyclomatic Complexity.
TPKP - Number of tests versus LOPDL.
SPKP - Number of test steps versus LOPDL.
VPKP - Lines of verify / expected results versus LOPDL.
TPCI - Number of tests versus CIs.
SPCI - Number of test steps versus CIs.
VPCI - Lines of verify / expected results versus CIs.

Information Flow

None.

Changes in Design

None.

Defect Categorization

None.

Logical Groupings and Relationships

TTES or TPCC or TPKP or TPCI.

TSTE or SPCC or SPKP or SPCI.

TVES or VPCC or VPKP or VPCI.

C.9 Code Walk-Throughs

What Design Artifacts are Analyzed?

1. Code walk-through Action Items.

Measures for:

Internal Complexity

None.

Information Flow

None.

Changes in Design

None.

Defect Categorization

TAI - Total Action Items.
AIPKP - Action Items per LOPD.
AIPCI - Action Items per CI.
AIPCC - Action Items per Cyclomatic Complexity.
AIND - Action Items for persons other than the designer.
AIR1 - Action Items for designer.
AIRI - Action Items requiring redesign/investigation.
AIUT - Action Items for unit test plans.
AIUTPUT - Action Items for unit test plans per test.
AIUTPCC - Action Items for unit test plans per total Cyclomatic Complexity.

Logical Groupings and Relationships

TAI or AIPKP or AIPCI or AIPCC.

TAI or (AIND and AIR1).

AIUT or AIUTPUT or AIUTPCC.

Appendix D List of Acronyms

The following is a list of acronyms used as variable names for predictive software design measures and as terms in the thesis document.

Acronym List

A1PR	Action Items for designer per requirement of preliminary design (prior to review)
ACI	Affected CIs (number of defects)
ADC	Absolute difference in Cyclomatic Complexity
ADCGT	Absolute difference in modules with Cyclomatic Complexity greater than ten
ADCI	Absolute difference number of CIs
ADDC	Absolute difference in Data complexity
ADFI	Absolute difference in Fan-In
ADFO	Absolute difference in Fan-Out
ADIF	Absolute difference in Information Flow Metric without PDL
ADIFP	Absolute difference in Information Flow Metric with PDL
ADMO	Absolute difference in number of modules
ADPDL	Absolute difference in Lines of PDL
ADR	Absolute difference in total requirements counts
ADSC	Absolute difference in Structural complexity
AIARI	Action Items requiring redesign/ investigation
AIND	Action Items for persons other than the designer
AINDPR	Count of Action Items for persons other than designer per requirement of preliminary design (prior to review)
AIPCC	Total Action Items per Cyclomatic Complexity
AIPCI	Total Action Items per CI
AIPKP	Total Action Items per LOPD
AIR1	Action Items for designer
AIRIPR	Count of action items to re-design or investigate per requirement of preliminary design (prior to review)
AIT	Total Action Items
AIUT	Action Items for unit test plans
AIUTPCC	Unit test Action Items per Cyclomatic Complexity
AIUTPUT	Unit test Action Items per test
APC	Action Items per CI
APP	Action Items per LOPDL
AR	Allocated requirement counts
ARIPC	Action Items requiring redesign/ investigation per CI
ARIPP	Action Items requiring redesign/ investigation per LOPDL
C	Total Cyclomatic Complexity
CCI	Cyclomatic Complexity per CI
CGT	Total Modules with Cyclomatic Complexity greater than 10
CI	Number of CIs as a PSDM

CI	Configuration Item and functional component of a CSC
CIPRE	Number of CIs per number of requirements
CKP	Cyclomatic Complexity per LOPDL
CMO	Cyclomatic Complexity per module
CO	Constraint counts
CSC	Computer System Component (Major functional area of the software design)
CSS	Corrected sum of squared error or difference
DA23TR	Absolute difference in total requirements from preliminary design prior to review to preliminary design after review
DC	Data complexity
DE	Count of number of sentences in description of execution
DPC23DE	Difference as percent change in the number of sentences in the preliminary design description from pre-review to post-review
DR	Derived requirement counts
EX	Exception handlers and raised
FC	Total capabilities (Functional Capabilities design artifact)
FI	Limited enhanced Fan-In
FO	Limited enhanced Fan-Out
IF	Information Flow Metric without lines of PDL
IFP	Information Flow Metric with lines of PDL
LO	Loops
MO	Number of modules
MOPRE	Number of modules per requirement
MSE	Mean squared error
NOD	Node counts in preliminary design information flow diagram
PAT	Total Parameters passed to and from CSC in preliminary design information flow diagram
PC23TR	Percent change in requirements from preliminary design prior to review to preliminary design after review
PCC	Percent change in Cyclomatic Complexity
PCCI	Percent change in number of CIs
PCDC	Percent change in Data complexity
PCFI	Percent change in Fan-In
PCFO	Percent change in Fan-out
PCIF	Percent change in Information Flow Metric without PDL
PCIFP	Percent change in Information Flow Metric with PDL
PCMO	Percent change in number of modules
PCPDL	Percent change in lines of PDL
PCSC	Percent change in Structural complexity
PCTR	Percent change in total requirements counts
PDL	Lines of PDL as a PSDM
PDLPRE	Lines of PDL per requirement
PDV	Predicted defect value

PFR	Counts of parameters passed from CSC in preliminary design information flow diagram
PTO	Counts of parameters passed to CSC in preliminary design information flow diagram
PSDM	Predictive Software Design Measure
RSV	R-squared Value of an equation formed through multiple regression
RUSS	Uncorrected sum of squared differences in the ranking of CSCs by predicted defects
SC	Structural complexity (for a CSC by CIs)
SPCC	Steps per Cyclomatic Complexity in a unit test plan
SPCI	Steps per CI in a unit test plan
SPKP	Steps per thousand lines of PDL in a unit test plan
STR	System trouble report
TAI	Total Action Items
TAPR	Total Action Items per requirement of preliminary design (prior to review)
TNDA	Action Items assigned persons other than the designer
TNDAPC	Action Items assigned persons other than the designer Per CI
TNDAPP	Action Items assigned persons other than the designer Per LOPDL
TPCC	Unit tests Per Cyclomatic Complexity
TPCI	Unit tests per CI
TPKP	Unit tests Per thousand lines of PDL
TR	Total requirement counts
TSTE	Number of test steps in a unit test plan
TTES	Number of tests in a unit test plan
TVES	Lines of verify / expected results in a unit test plan
UP	Unconventional control (GOTOs, loop exits, extra procedure exits)
USS	Uncorrected sum of squared error or difference
VPCC	Verify statements Per Cyclomatic Complexity in a unit test plan
VPCI	Verify statements per CI in a unit test plan
VPKP	Verify statements Per thousand lines of PDL in a unit test plan

Appendix E PSDM Values

Recall from Chapter Five that multiple regression combines one or more independent variables into an equation to predict or calculate a single dependent variable. In this research, the dependent variables are predictive software design metrics (PSDMs) and the independent variable is the number of defects associated with a given functional area. For the functional area or CSC, the value resulting from the regression equations is the sum of PSDM values multiplied by calculated coefficients. The coefficient values are calculated through a "least squares" fitting of a line through the available data points. These data points are the PSDM values.

This appendix documents the PSDM values calculated for each milestone in the creation of the predictive equations of Chapter Five. There are five data values for each PSDM corresponding to the five CSCs in the test group. The number of PSDMs varies for each design milestone.

Design Milestones

1. Requirements allocation.
2. Preliminary design.
3. Preliminary design review.
4. Detailed design.
5. Detailed design review.
6. As-built design.
7. Unit test plans.
8. Code walk-throughs.

E.1. Requirements Allocation

Table E.1 contains the PSDM values used in creating prediction equations at the design milestone one (requirements allocation).

Table E.1 PSDM Data at Requirements Allocation

CSC	Total Defects	Allocated Requirements	Derived Requirements	Constraints	Total Requirements	Functional Capabilities
TL	**24**	20	3	4	27	36
TO	**34**	14	9	10	33	8
TI	**49**	31	0	0	31	46
TR	**111**	56	43	13	112	103
TT	**134**	19	13	5	37	38

The following variable names are assigned to the PSDMs in Table E.1:

ACI - Total Defects (Affected CIs).
AR - Allocated Requirements.
CO - Constraints.
CSC - CSC.
DR - Derived Requirements.
FC - Functional Capabilities.
TR - Total Requirements.

E.2 Preliminary Design

Tables E.2.1, E.2.2, and E.2.3 present the PSDM data used at the second design milestone (Preliminary Design). This data is separated by areas of concern in predictive measurements.

Table E.2.1 is the information flow measures extracted from control flow diagrams.

Table E.2.1 Information Flow PSDM Values at Preliminary Design

CSC	Total Defects	Nodes	Total Parameters	Parameters To CSC	Parameters From CSC
TL	24	5	7	4	3
TO	34	21	38	36	2
TI	49	24	72	25	47
TR	111	14	14	7	7
TT	134	19	23	2	21

Table E.2.2 contains the sizing portion the change in design measures for the preliminary design.

Table E.2.2 Internal Complexity and Sizing PSDMs at Preliminary Design

CSC	Total Defects	Allocated Requirements	Derived Requirements	Constraints	Total Requirements	Description /Execution (Sentences)
TL	24	20	3	4	27	50
TO	34	14	9	10	33	12
TI	49	18	8	3	29	25
TR	111	24	7	2	33	10
TT	134	19	13	5	37	19

Table E.2.3 contains the changes in the total number of requirements allocated to a given CSC from requirements allocation to preliminary design.

Table E.2.3 Change in Design PSDMs at Preliminary Design

CSC	Total Defects	Absolute Difference in Requirements Totals	Percent Change in Requirements Totals
TL	24	0	0.000
TO	34	0	0.000
TI	49	2	6.897
TR	111	79	239.394
TT	134	0	0.000

The following are the variable names assigned to the PSDMs in Tables E.2.1, E.2.2, and E.2.3.

ACI - Total Defects (Affected CIs).
ADR - Absolute difference in requirements totals.
AR - Allocated Requirements.
CO - Constraints.
CSC - CSC.
DES - Description and/or execution sentences.
DR - Derived Requirements.
NOD - Nodes (number of different CSC specified in control flow diagram).
PAT - Total parameters.
PCR - Percent change in requirements totals.
PFR - Parameters passed from CSC.
PTO - Parameters passed to CSC.
TR - Total Requirements.

E.3. Preliminary Design Review

Tables E.3.1, E.3.2, E.3.3 present the PSDM data used at the third design milestone (preliminary design review). This data is separated by areas of concern in predictive measurements.

Tables E.3.1 and E.3.2 contain PSDMs drawn from the Action Items. They address defect categorization as an area of concern in predictive

measurement. The first table contains the basic Action Item data and the second table is this data normalized by the number of requirements reviewed.

Table E.3.1 Defect Data PSDMs at the Preliminary Design Review

CSC	Total Requirements	Total Action Items	Designer Action Items	Redesign / Investigate Action Items	Non-Designer Action Items
TL	24	12	12	2	0
TO	34	25	21	4	4
TI	49	10	8	5	2
TR	111	29	26	5	3
TT	134	24	19	10	5

Table E.3.2 PSDMs for Defect Data Normalized by Requirement Count at the Preliminary Design Review

CSC	Total Defects	Total Action Items Per Requirement	Designer Action Items Per Requirement	Redesign / Investigate Action Items Per Requirement	Non-Designer Action Items Per Requirement
TL	24	0.444	0.444	0.074	0.000
TO	34	0.758	0.636	0.121	0.121
TI	49	0.345	0.276	0.172	0.069
TR	111	0.879	0.788	0.152	0.091
TT	134	0.649	0.514	0.270	0.135

Table E.3.3 shows the PSDMs reflecting the size of each CSC in the post-review preliminary design.

Table E.3.3 Internal Complexity and Sizing PSDMs at the Preliminary Design Review

CSC	Total Defects	Allocated Requirements	Derived Requirements	Constraints	Total Requirements	Description /Execution Sentences
TL	24	20	3	4	27	50
TO	34	14	9	10	33	12
TI	49	13	8	3	24	25
TR	111	44	9	4	57	16
TT	134	21	23	4	48	35

Table E.3.4 contains the PSDMs measuring the change in the design resulting from the preliminary design review.

Table E.3.4 Change in Design PSDMs at the Preliminary Design Review

CSC	Total Defects	Absolute Difference in Total Requirements	Percent Change in Total Requirements	Absolute Change in Description / Execution Sentences	Percent Change in Description / Execution Sentences
TL	24	0	0.000	0	0.000
TO	34	0	0.000	0	0.000
TI	49	5	17.241	0	0.000
TR	111	24	72.727	6	37.500
TT	134	11	29.730	16	45.714

The variable names assigned to the PSDMs of Tables E.3.1, E.3.2, E.3.3, and E.3.4 are given below.

A1PR - Designer Action Items per requirement.
ACI - Total defects (affected CIs).
AIARI - Redesign/investigate Action Items.
AIND - Non-designer Action Items.
AINDPR - Non-designer Action Items per requirement.
AIR1 - Designer Action Items.
AR - Allocated requirements.
ARIPR - Redesign/investigate Action Items per requirement.
CO - Constraints.
CSC - CSC.
DA23DE - Absolute change in description/execution sentences.
DA23TR - Absolute Difference in Total Requirements.
DE - Description/execution sentences.
DPC23DE - Percent change in description/execution sentences.
DPC23TR - Percent change in total requirements.
DR - Derived requirements.
TAI - Total Action Items.
TAPR - Total Action Items per requirement.
TR - Total requirements.

E.4. Detailed Design

Tables E.4.1, E.4.2, E.4.3, E.4.4 present the PSDM values used to create prediction equations for the detailed design milestone. Table E.4.1 contains information flow measures.

Table E.4.1 Information Flow PSDMs at the Detailed Design Milestone

CSC	Total Defects	Fan-In	Fan-Out	Information Flow Metric	Information Flow Metric including LOPDL	Data Complexity	Structural Complexity
TL	24	26	38	976144	237203	0.128	1169
TO	34	232	87	4.07E+08	2.18E+08	0.534	4187
TI	49	128	160	4.19E+08	2.46E+08	0.236	3904
TR	111	175	264	2.13E+09	2.11E+09	0.283	9537
TT	134	112	253	8.03E+08	7.15E+08	0.165	6134

Table E.4.2 contains PSDMs measuring Cyclomatic Complexity characteristics of the detailed design.

Table E.4.2 Cyclomatic Complexity PSDMs at the Detailed Design Milestone

CSC	Total Defects	Cyclomatic Complexity	Cyclomatic Complexity per CI	Cyclomatic Complexity per Module	Cyclomatic Complexity per thousand LOPDL	Modules with Cyclomatic Complexity > 10
TL	24	27	4.5	1.125	93.426	0
TO	34	68	6.8	1.659	93.664	0
TI	49	79	4.2	1.386	82.550	3
TR	111	128	6.1	1.306	54.214	2
TT	134	130	5.7	1.667	50.116	2

Structural and sizing characteristics are contained in table E.4.3. Public modules are those specified as being accessible by other CIs and CSCs. Unconventional programming elements are loop exits, gotos, and extra procedure exits.

Table E.4.3 Internal Complexity and Sizing PSDMs at the Detailed Design Milestone

CSC	Total Defects	PDL Statements	CIs	Modules	Public Modules	Loops	Unconventional Programming Elements	Exceptions (Raised and Handlers)
TL	24	243	6	24	17	1	0	0
TO	34	534	10	41	28	5	0	3
TI	49	586	19	57	52	7	1	3
TR	111	987	21	98	71	5	0	9
TT	134	890	23	78	53	9	0	0

PSDMs in Table E.4.4 address the change in design from preliminary design to detailed design. The number of requirements is used as a normalization factor and is calculated from the post-review preliminary design.

Table E.4.4 Change in Design PSDMs at the Detailed Design Milestone

CSC	Total Defects	PDL Statements Per Requirement	CIs Per Requirement	Modules Per Requirement
TL	24	9.0	0.2	0.889
TO	34	16.2	0.3	1.242
TI	49	24.4	0.8	2.375
TR	111	17.3	0.4	1.719
TT	134	18.5	0.5	1.625

The variable names assigned to the PSDMs of Tables E.4.1, E.4.2, E.4.3, and E.4.4 are given below.

ACI - Total Defects (Affected CIs).
C - Cyclomatic Complexity.
CCI - Cyclomatic Complexity per CI.
CGT - Modules with Cyclomatic Complexity greater than ten.
CI - CIs (Configuration Items).
CIPRE - CIs per preliminary design requirement.
CMO - Cyclomatic Complexity per module.
CPP - Cyclomatic Complexity per thousand LOPDL.
CSC -CSC.
DC - Data Complexity.
EX - Exceptions (raised and handlers).
FI - Fan-In.
FO - Fan-Out.
IF - Information Flow metric (without LOPDL).
IFP - Information Flow metric (with LOPDL).
MO - Modules.
MOPRE - Modules per preliminary design requirement.
PDL - Lines of PDL.
PDLPRE - LOPDL per preliminary design requirement.
PM - Public Modules.
SC - Structural Complexity.
UP - Unconventional programming elements.

E.5. Detailed Design Review

Tables E.5.1 and E.5.2 contain PSDMs calculated from the design artifacts available at the detailed design review milestone address defect categorization and sizing. Action Items provide the defect data, and sizing is addressed as LOPDL, CIs, and modules.

Table E.5.1 contains PSDMs using the total number of Action Items.

Table E.5.1 Defect Data PSDMs at the Detailed Design Review

CSC	Total Defects	Total Action Items	Total Action Items per Thousand LOPDL	Total Action Items per CI	Total Action Items Per Module
TL	24	14	57.613	2.33	0.583
TO	34	20	37.453	2	0.488
TI	49	45	76.792	2.37	0.789
TR	111	122	123.607	5.81	1.245
TT	134	108	121.348	4.7	1.385

Action Items are categorized by the action required and the responsible person(s). Table E.5.2 contains PSDMs calculated from this information with some entries normalized by sizing measures.

Table E.5.2 Categorized Defect Data PSDMs at the Detailed Design Review

CSC	Total Defects	Redesign / Investigate Action Items	Redesign / Investigate Action Items per Thousand LOPDL	Redesign / Investigate Action Items Per CI	Total Non-Designer Action Items	Total Non-Designer Action Items Per Thousand LOPDL	Total Non-Designer Action Items Per CI
TL	24	3	12.346	0.500	0	0.000	0.000
TO	34	1	1.873	0.100	1	1.873	0.100
TI	49	16	27.304	0.842	4	6.826	0.211
TR	111	20	20.263	0.952	3	3.040	0.143
TT	134	25	28.090	1.087	8	8.989	0.348

The following is a list of the variable names assigned to the PSDMs in Tables E.5.1 and E.5.2 from the detailed design review milestone.

ACI - Total Defects (Affected CIs).
AIRI - Redesign/investigate Action Items.
AIRIPC - Redesign/investigate Action Items per CI.
AIT - Total Action Items.
APC - Action Items per CI.
APM - Action Items per module.
APP - Action Items per thousand LOPDL.
ARIPP - Redesign/investigate Action Items per thousand LOPDL.
CSC - CSC.
TNDA - Total non-designer Action Items.
TNDAPC - Total non-designer Action Items per CI.
TNDAPP - Total non-designer Action Items per thousand LOPDL.

E.6. As-Built Design

The definition of the as-built design PSDMs are the same as those calculated at the detailed design milestone. As the design changes, these values also change. The PSDMs relating to the as-built design only are presented first in tables E.6.1, E.6.2, and E.6.3. Those PSDMs comparing the design characteristics of the detailed design and the as-built design are presented last in tables E.6.4 through E.6.9.

Table E.6.1 contains PSDMs addressing information flow.

Table E.6.1 Information Flow PSDMs at the As-Built Design Milestone

CSC	Total Defects	Fan-In	Fan-Out	Information Flow Metric	Information Flow Metric including Thousand LOPDL	Data Complexity	Structural Complexity
TL	24	33	43	2013561	581919	0.364	1110
TO	34	630	86	2.94E+09	2.13E+09	0.793	4568
TI	49	298	151	2.02E+09	1.94E+09	1.007	3248
TR	111	492	522	6.6E+10	1.56E+11	0.593	42003
TT	134	372	552	4.22E+10	1.09E+11	0.315	23362

Table E.6.2 contains PSDMs measuring Cyclomatic Complexity characteristics of the detailed design.

Table E.6.2 Cyclomatic Complexity PSDMs at the As-Built Design Milestone

CSC	Total Defects	Cyclomatic Complexity	Cyclomatic Complexity per CI	Cyclomatic Complexity per Module	Cyclomatic Complexity per thousand LOPDL	Modules with Cyclomatic Complexity > 10
TL	24	34	4.9	0.919	117.647	0
TO	34	112	10.0	2.154	154.270	1
TI	49	151	6.3	1.624	157.785	4
TR	111	274	9.8	1.420	116.053	4
TT	134	458	15.0	2.503	176.561	6

Structural and sizing characteristics are contained in table E.6.3. Public modules are those specified as being accessible by other CIs and CSCs. Unconventional programming elements are loop exits, gotos, and extra procedure exits.

Table E.6.3 Internal Complexity PSDMs at the As-Built Design Milestone

CSC	Total Defects	PDL Statements	CIs	Modules	Public Modules	Loops	Unconventional Programming Elements	Exceptions (Raised and Handlers)
TL	24	289	7	37	22	3	3	0
TO	34	726	11	52	36	5	0	7
TI	49	957	24	93	80	12	23	18
TR	111	2361	28	193	100	38	15	228
TT	134	2594	30	183	103	61	40	64

The variable names assigned to the PSDMs in tables E.6.1, E.6.2, and E.6.3 are given below.

ACI - Total Defects (Affected CIs).
C - Cyclomatic Complexity.
CCI - Cyclomatic Complexity per CI.
CGT - Modules with Cyclomatic Complexity greater than ten.
CI - CIs (Configuration Items).
CIPRE - CIs per preliminary design requirement.
CMO - Cyclomatic Complexity per module.
CPP - Cyclomatic Complexity per thousand LOPDL.
CSC -CSC.
DC - Data Complexity.

EX - Exceptions (Raised and Handlers).
FI - Fan-In.
FO - Fan-Out.
IF - Information Flow metric (without LOPDL).
IFP - Information Flow metric (with LOPDL).
MO - Modules.
MOPRE - Modules per preliminary design requirement.
PDL - Lines of PDL.
PDLPRE - LOPDL per preliminary design requirement.
PM - Public Modules.
SC - Structural Complexity.
UP - Unconventional programming elements.

As-Built Design Changes

The PSDMs in tables E.6.4 through E.6.9 present measures of the change in the design from the detailed design milestone to the as-built design milestone. Absolute difference PSDMs are calculated as the absolute value of the detailed design value subtracted from the as-built design value. The percent change PSDMs are the absolute difference values multiplied by one hundred (percent) and divided by the detailed design value.

Table E.6.4 contains the absolute difference values for the information flow PSDMs.

Table E.6.4 Information Flow Absolute Difference PSDMs

CSC	Total Defects	Absolute Difference in Fan-In	Absolute Difference in Fan-Out	Absolute Difference in Information Flow Metric	Absolute Difference in Information Flow Metric including Thousand LOPDL	Absolute Difference in Data Complexity	Absolute Difference in Structural Complexity
TL	24	7	5	1E+06	344716	0.23543	59
TO	34	398	1	3E+09	2E+09	0.25901	381
TI	49	170	9	2E+09	2E+09	0.77055	656
TR	111	317	258	6E+10	2E+11	0.30972	32466
TT	134	260	299	4E+10	1E+11	0.14929	17228

Table E.6.5 contains the absolute difference values for the Cyclomatic Complexity PSDMs.

Table E.6.5 Cyclomatic Complexity Absolute Difference PSDMs

CSC	Total Defects	Absolute Difference in Cyclomatic Complexity	Absolute Difference in Cyclomatic Complexity per CI	Absolute Difference in Cyclomatic Complexity per Module	Absolute Difference in Cyclomatic Complexity per Thousand LOPDL	Absolute Difference in Modules with Cyclomatic Complexity > 10
TL	24	7	0.3571	0.206	24.221	0
TO	34	44	3.3818	0.495	60.606	1
TI	49	72	2.1338	0.238	75.235	1
TR	111	146	3.6905	0.114	61.838	2
TT	134	328	9.6145	0.836	126.450	4

Table E.6.6 presents the absolute difference values for PSDMs addressing the physical characteristics of the design. Public modules are those specified as being accessible by other CIs and CSCs. Unconventional programming elements are loop exits, gotos, and extra procedure exits.

Table E.6.6 Internal Complexity Absolute Difference PSDMs

CSC	Total Defects	Absolute Difference in PDL Statements	Absolute Difference in CIs	Absolute Difference in Modules	Absolute Difference in Public Modules	Absolute Difference in Loops	Absolute Difference in Unconventional Programming Elements	Absolute Difference in Exceptions (Raised and Handlers)
TL	24	46	1	13	5	2	3	0
TO	34	192	1	11	8	0	0	4
TI	49	371	5	36	28	5	22	15
TR	111	1374	7	95	29	33	15	219
TT	134	1704	7	105	50	52	40	64

Table E.6.7 contains the percent change values for the information flow PSDMs.

Table E.6.7 Information Flow Percent Change PSDMs

CSC	Total Defects	Percent Change in Fan-In	Percent Change in Fan-Out	Percent Change in Information Flow Metric	Percent Change in Information Flow Metric including Thousand LOPDL	Percent Change in Data Complexity	Percent Change in Structural Complexity
TL	24	26.92	13.16	106.3	145.3	183.64	5.05
TO	34	171.54	1.15	620.6	879.6	48.50	9.10
TI	49	132.81	5.63	382.8	688.4	326.47	16.80
TR	111	181.14	97.73	2990.2	7292.1	109.43	340.42
TT	134	232.14	118.18	5151.5	15206.0	90.29	280.86

Table E.6.8 contains the percent change values for the Cyclomatic Complexity PSDMs.

Table E.6.8 Cyclomatic Complexity Percent Change PSDMs

CSC	Total Defects	Percent Change in Cyclomatic Complexity	Percent Change in Cyclomatic Complexity per CI	Percent Change in Cyclomatic Complexity per Module	Percent Change in Cyclomatic Complexity per Thousand LOPDL
TL	24	25.93	7.94	18.32	25.92
TO	34	64.71	49.73	29.86	64.71
TI	49	91.14	51.32	17.15	91.14
TR	111	114.06	60.55	8.695	114.06
TT	134	252.31	170.10	50.16	252.31

Table E.6.9 presents the percent change values for PSDMs addressing the physical characteristics of the design. Public modules are those specified as being accessible by other CIs and CSCs. Unconventional programming elements are loop exits, gotos, and extra procedure exits.

Table E.6.9 Internal Complexity Percent Change PSDMs

CSC	Total Defects	Percent Change in PDL Statements	Percent Change in CIs	Percent Change in Modules	Percent Change in Public Modules
TL	24	18.93	16.67	54.17	29.41
TO	34	35.96	10.00	26.83	28.57
TI	49	63.31	26.32	63.16	53.85
TR	111	139.21	33.33	96.94	40.85
TT	134	191.46	30.43	134.62	94.34

The variable names assigned to the PSDMs of Tables E.6.4 through E.6.9 are listed below.

ACI - Total Defects (Affected CIs).
ADC - Cyclomatic Complexity.
ADCCI - Cyclomatic Complexity per CI.
ADCGT - Modules with Cyclomatic Complexity greater than ten.
ADCI - CIs (Configuration Items).
ADCIPRE - CIs per preliminary design requirement.
ADCMO - Cyclomatic Complexity per module.
ADCPP - Cyclomatic Complexity per thousand LOPDL.
ADDC - Data Complexity.
ADEX - Exceptions (Raised and Handlers).
ADFI - Fan-In.
ADFO - Fan-Out.
ADIF - Information Flow metric (without LOPDL).
ADIFP - Information Flow metric (with LOPDL).
ADMO - Modules.
ADMOPRE - Modules per preliminary design requirement.
ADPDL - Lines of PDL.
ADPDLPRE - LOPDL per preliminary design requirement.
ADPM - Public Modules.
ADSC - Structural Complexity.
ADUP - Unconventional programming elements.
CSC -CSC.
PCC - Cyclomatic Complexity.
PCCCI - Cyclomatic Complexity per CI.
PCCGT - Modules with Cyclomatic Complexity greater than ten.
PCCI - CIs (Configuration Items).
PCCIPRE - CIs per preliminary design requirement.
PCCMO - Cyclomatic Complexity per module.
PCCPP - Cyclomatic Complexity per thousand LOPDL.
PCDC - Data Complexity.
PCEX - Exceptions (Raised and Handlers)

PCFI - Fan-In.
PCFO - Fan-Out.
PCIF - Information Flow metric (without LOPDL).
PCIFP - Information Flow metric (with LOPDL).
PCMO - Modules.
PCMOPRE - Modules per preliminary design requirement.
PCPDL - Lines of PDL.
PCPDLPRE - LOPDL per preliminary design requirement.
PCPM - Public Modules.
PCSC - Structural Complexity.
PCUP - Unconventional programming elements.

E.7. Unit Test Plans

PSDMs calculated at the unit test plan milestone measure the number of tests, the number of individual test steps, and the number of sentences in verifying the expected results. These counts are then normalized by sizing measures and Cyclomatic Complexity.

Table E.7.1 presents the total counts of tests, steps, and verify sentences. Also included are these total count values normalized by the number of CIs.

Table E.7.1 Unit Test Sizes and Per CI PSDMs

CSC	Total Defects	Total Tests	Total Steps	Total Verify Sentences	Tests per CI	Steps per CI	Verify Sentences per CI
TL	24	7	42	80	1.000	6.000	11.429
TO	34	38	140	41	3.454	12.727	3.727
TI	49	77	147	79	3.208	6.125	3.292
TR	111	124	383	159	4.429	13.679	5.679
TT	134	71	246	115	2.367	8.200	3.833

The table below contains the total test count values normalized by lines of PDL and the number of modules.

Table E.7.2 Per PDL and Per Module Unit Test Data PSDMs

CSC	Total Defects	Tests per Thousand LOPDL	Steps per Thousand LOPDL	Verify Sentences per Thousand LOPDL	Tests per Module	Steps per Module	Verify Sentences per Module
TL	24	24.221	145.329	276.817	0.189	1.135	2.162
TO	34	52.342	192.838	56.474	0.731	2.692	0.788
TI	49	80.460	153.605	82.550	0.828	1.581	0.849
TR	111	52.520	162.220	67.344	0.642	1.984	0.823
TT	134	27.371	94.834	44.333	0.388	1.344	0.628

Table E.7.2 contains the total test values normalized by Cyclomatic Complexity values.

Table E.7.3 Per Cyclomatic Complexity Unit Test Data PSDMs

CSC	Total Defects	Tests per Cyclomatic Complexity	Steps per Cyclomatic Complexity	Verify Sentences per Cyclomatic Complexity
TL	24	0.206	1.235	2.353
TO	34	0.339	1.250	0.366
TI	49	0.510	0.974	0.523
TR	111	0.453	1.398	0.580
TT	134	0.155	0.537	0.251

The variable names assigned to the PSDMs in tables E.7.1, E.7.2, and E.7.3 are listed below.

ACI - Total Defects (Affected CIs).
CSC - CSC.
SPCC - Steps per Cyclomatic Complexity.
SPCI - Steps per CI.
SPKP - Steps per thousand LOPDL.
SPMO - Steps per module.
TPCC - Tests per Cyclomatic Complexity.
TPCI - Tests per CI.
TPKP - Tests per thousand LOPDL.
TPMO - Tests per module.
TSTE - Total steps.
TTES - Total tests.
TVES - Total verify sentences.
VPCC - Verify sentences per Cyclomatic Complexity.

VPCI - Verify sentences per CI.
VPKP - Verify sentences per thousand LOPDL.
VPMO - Verify sentences per module.

E.8. Code Walk-Throughs

PSDMs calculated from the design artifacts available at the code walk-through design milestone address defect categorization and sizing. Action Items provide the defects, and sizing is addressed as LOPDL, CIs, and modules.

Table E.8.1 contains PSDMs addressing the various categories of Action Items.

Table E.8.1 Action Item Data PSDMs at the Code Walk-Through Milestone

CSC	Total Defects	Designer Action Items	Redesign / Investigate Action Items	Total Action Items	Unit Test Action Items	Non-Designer Action Items
TL	24	30	8	32	1	2
TO	34	36	9	38	0	2
TI	49	23	9	25	5	2
TR	111	113	31	117	8	4
TT	134	31	9	36	0	5

Table E.8.2 contains the total counts of Action Items normalized by sizing measures. Also included are the unit test Action Item counts normalized by the number of unit tests and Cyclomatic Complexity.

Table E.8.2 Normalized Action Item Data PSDMs at the Code Walk-Through Milestone

CSC	Total Defects	Action Items per Thousand LOPDL	Action Items per CI	Unit Test Action Items per Unit Test	Unit Test Action Items per Cyclomatic Complexity
TL	24	110.727	4.571	0.143	0.029
TO	34	52.342	3.455	0.000	0.000
TI	49	26.123	1.042	0.065	0.033
TR	111	49.555	4.179	0.065	0.029
TT	134	13.878	1.200	0.000	0.000

The variable names assigned to the PSDMs of tables E.8.1 and E.8.2 are listed below.

ACI - Total defects (affected CIs).
AI1 - Designer Action Items.
AIND - Non-designer Action Items.
AIPCI - Action Items per CI.
AIPKP - Action Items per thousand LOPDL.
AIRI - Redesign/Investigate Action Items.
AIUT - Unit test Action Items.
AIUTPCC - Unit test Action Items per Cyclomatic Complexity.
AIUTPUT - Unit test Action Items per unit test.
CSC - CSC.
TAI - Total Action Items.

Appendix F Selected Prediction Equations

This appendix contains the selected prediction equations from Section Five of this document. The predictive software design measures (PSDMs) used and the coefficients calculated are presented for each of the selected prediction equations at each design milestone.

The design milestones are:

1. Requirements Allocation.
2. Preliminary Design.
3. Preliminary Design Review.
4. Detailed Design.
5. Detailed Design Review.
6. As-built Design.
7. Unit Test Plans.
8. Code Walk-Throughs.

F.1 Requirements Allocation

The following predicted defect value (PDV) equations are selected from those created using PSDM data available at the requirements allocation design milestone.

PDV = 1.239 TR
PDV = 1.282 FC
PDV = 2.223 AR

The PSDM variable labels are:

AR - Allocated requirements.
FC - Functional capabilities.
TR - Total requirements.

F.2 Preliminary Design

The following predicted defect value (PDV) equations are selected from those created using PSDM data available at the preliminary design milestone.

PDV = 8.841 DR
PDV = -1.837 PTO + 11.859 DR
PDV = -1.633 PTO + 10.581 DR + 0.203 PCR
PDV = -1.631 PTO + 10.582 DR + 0.613 ADR

The PSDM variable labels are:

ADR - Absolute difference in total requirements from requirements allocation to preliminary design.
DR - Derived requirements.
PCR - Percent change in total requirements from requirements allocation to preliminary design.
PTO - Parameters passed into.

F.3 Preliminary Design Review

The following predicted defect value (PDV) equations are selected from those created using PSDM data available at the preliminary design review milestone.

PDV = 13.671 AIARI
PDV = 270.60 ARIPR + 1.547 DPC23DE
PDV = 9.707 AIARI + 2.642 DA23TR
PDV = 620.90 TAPR - 20.793 AIR1 +2.789 DPC23DE
PDV = 227.50 ARIPR + 0.530 AR + 1.373 DPC23DE
PDV = 265.30 ARIPR + 1.211 DA23TR + 1.088 DPC23DE

The PSDM variable labels are:

AIARI - Action Items to re-design or investigate.

AR - Allocated requirements in post-review preliminary design.

ARIPR - Action Items for persons other than designer per preliminary design requirement (prior to review).

DA23TR - Absolute difference in total preliminary design requirements from pre-review to post-review.

DPC23DE - Difference as percent change in the number of sentences in the preliminary design description from pre-review to post-review.

TAPR - Total Action Items per requirement reviewed.

F.4 Detailed Design

The following predicted defect value (PDV) equations are selected from those created using PSDM data available at the detailed design milestone.

PDV = 0.445 FO

PDV = 0.857 C

PDV = 9.444 CI - 51.525 MOPRE

PDV = -0.218 FI + 1.166 C

PDV = 4.220E-08 IFP - 8.987 EX + 0.803 C

PDV = - 6.643 PDLPRE + 8.962 CI + 29.843 CMO

PDV = 2.502 PDLPRE + 1.09E-07IF -18.182 EX

The PSDM variable labels are:

C - Cyclomatic Complexity.

CI - Configuration Items (functional components that make up a CSC).

CMO - Cyclomatic Complexity per module.

EX - Exception handers and exceptions raised.

FI - Fan-in.

FO - Fan-out.

IFP - Information Flow Metric with multiplication by lines of PDL.

MOPRE - Modules per preliminary design requirement.

PDLPRE - Lines of PDL per preliminary design requirement.

F.5 Detailed Design Review

The following predicted defect value (PDV) equations are selected from those created using PSDM data available at the detailed design review milestone.

PDV = 1.071 AIT

PDV = 14.098 APC + 7.749 TNDA

PDV = 0.813 AIT + 113.20 TNDAPC

PDV = 16.955 APC - 1.153 ARIPP +10.717 TNDA

PDV = 1.007 APP -2.446 ARIPP + 10.074 TNDA

The PSDM variable labels are:

AIT - Total Action Items.

APC - Action Items per CI.

APP - Action Items per line of PDL.

ARIPP - Action Items requiring redesign/investigation per line of PDL.

TNDA - Total non-designer Action Items.

TNDAPC - Total non-designer Action Items per CI reviewed.

F.6 As-Built Design

The following predicted defect value (PDV) equations are selected from those created using PSDM data available at the as-built design milestone. The first group of equations uses PSDMs measuring characteristics of the as-built design and the second group of equations uses PSDMs measuring the change in the design from the detailed design to the as-built design.

Selected as-built design characteristic prediction equations:

PDV = 0.050 PDL
PDV = 0.193 FO + 0.127 CKP
PDV = 0.365 MO + 0.1436 C
PDV = 0.149 FO + 0.472 LO + 0.129 CKP
PDV = 0.214 FO - 0.060 EX + 0.113 CKP

The PSDM variable labels are:

C - Cyclomatic Complexity.
CKP - Cyclomatic Complexity per thousand lines of PDL.
EX - exception handers and exceptions raised.
FO - Fan-out.
LO - Number of looping control structures.
MO - Number of modules.
PDL - lines of PDL.

Prediction equations for the change in design from detailed design to as-built design:

PDV = 0.7438 PCPDL
PDV = 1.2509 ADMO
PDV = 0.9965 PCPDL - 0.1744 ADC
PDV = 1.0734 ADMO + 0.5277 PCCMO
PDV = 4.3368 ADLO - 1.3922 PCC +2.0532 ADCKP
PDV = 1.3085 ADMO -0.1329 ADC + 48.5184 ADCMO

The PSDM variable labels are:

ADC - Absolute difference in the Cyclomatic Complexity.
ADCKP - Absolute difference in the Cyclomatic Complexity per thousand lines of PDL.
ADCMO - Absolute difference in the Cyclomatic Complexity per module.
ADLO - Absolute difference in the number of looping structures.
ADMO - Absolute difference in the number of modules.
PCC - Percent change in the Cyclomatic Complexity.
PCCMO - Percent change in the Cyclomatic Complexity per module.
PCPDL - Percent change in the number of lines of PDL.

F.7 Unit Test Plans

The following predicted defect value (PDV) equations are selected from those created using PSDM data available at the unit test plan design milestone.

PDV = 0.354 TSTE
PDV = 0.762 TVES
PDV = 1.044 TVES -394.30 TPCC +188.60 TPMO
PDV = - 257.10 SPCC +74.876 TPCI + 23.620 VPCI
PDV = 1.126 TVES - 318.40 TPCC -27.247 SPCC + 178.70 TPMO

The PSDM variable labels are:

SPCC - Unit test steps per Cyclomatic Complexity.
TPCC - Unit tests per Cyclomatic Complexity.
TPCI - Unit tests per CI.
TPMO - Unit tests per module.
TSTE - Unit test steps.
TVES - Unit test verify statements and sentences.
VPCI - Unit test verify statements and sentences per CI.

F.8 Code Walk-Throughs

The following predicted defect value (PDV) equations are selected from those created using PSDM data available at the code walk-through design milestone.

PDV = 25.057 AIND
PDV = 28.205 AIND - 33.415 AIPCC
PDV = 29.105 AIND - 5.502 AIPCI
PDV = 0.337 AIR1 + 26.100 AIND - 8.195 AIPCI
PDV = 1.195 AIRI +25.873 AIND -7.880 AIPCI

The PSDM variable names are:

AIND - Non-designer Action Items.
AIPCC - Action Items per Cyclomatic Complexity reviewed.
AIR1 - Designer Action Items.
AIRI - Action Items requiring redesign/investigation .
AIPCI - Action Items per CI reviewed.

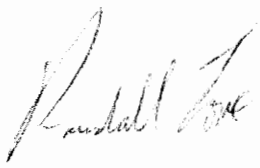
Vita

Name: Randall James Love

Address: 11 Hammersmith Road
Avon, Connecticut

Date of Birth: 25 November 1964

Education:

M.S. Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia
October 1994

B.S. Computer Science and Engineering
University of Connecticut
Storrs, Connecticut
May 1988