# NAVAL POSTGRADUATE SCHOOL
## Monterey, California


# DISSERTATION

**ENHANCEMENTS AND EXTENSIONS OF FORMAL MODELS FOR RISK ASSESSMENT IN SOFTWARE PROJECTS**

by

Michael R. Murrah

September 2002

Dissertation Advisor: Luqi

**Approved for public release; distribution is unlimited**

THIS PAGE INTENTIONALLY LEFT BLANK

| REPORT DOCUMENTATION PAGE | | | Form Approved OMB No. 0704-0188 |
|---|---|---|---|
| Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503. | | | |
| **1. AGENCY USE ONLY** *(Leave blank)* | **2. REPORT DATE** September 2002 | **3. REPORT TYPE AND DATES COVERED** Ph.D. Dissertation | |
| **4. TITLE AND SUBTITLE**: Enhancements and Extensions of Formal Models for Risk Assessment in Software Projects | | | **5. FUNDING NUMBERS** |
| **6. AUTHOR(S)** Murrah, Michael R. | | | |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** Naval Postgraduate School Monterey, CA 93943-5000 | | | **8. PERFORMING ORGANIZATION REPORT NUMBER** |
| **9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)** N/A | | | **10. SPONSORING / MONITORING AGENCY REPORT NUMBER** |
| **11. SUPPLEMENTARY NOTES** The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. | | | |
| **12a. DISTRIBUTION / AVAILABILITY STATEMENT** Approved for public release; distribution is unlimited | | | **12b. DISTRIBUTION CODE** |

**13. ABSTRACT** ***(maximum 200 words)***


The Modified Risk Model is a macro model developed to aid program managers in effectively planning the required effort to deliver software products. The model projects the probability of completing a software project, subject to the available resources supplied by management. This approach to software project risk management is unique because the model's input parameters are derived. Subjective variables are not part of the model. Different program managers would derive the same projections on the same software project.

Risk management is most effective in impacting the project's success if project risks are identified and mitigated early in the software lifecycle. The Modified Risk Model was developed specifically for this purpose. Additionally, the Modified Risk Model is versatile enough to be adapted to any software development activity.


| **14. SUBJECT TERMS** Risk Assessment, Formal Models, Software Estimation Models, Software Metrics, Project Management | | | **15. NUMBER OF PAGES** 371 |
|---|---|---|---|
| | | | **16. PRICE CODE** |
| **17. SECURITY CLASSIFICATION OF REPORT** Unclassified | **18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified | **19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified | **20. LIMITATION OF ABSTRACT** UL |

THIS PAGE INTENTIONALLY LEFT BLANK

**Approved for public release; distribution is unlimited**

**ENHANCEMENTS AND EXTENSIONS OF FORMAL MODELS FOR RISK ASSESSMENT IN SOFTWARE PROJECTS**

Michael R. Murrah
Major, United States Army
B.S., Georgia College, 1993
M.S., University of Missouri – Rolla, 1997
M.S., Naval Postgraduate School, 2002

Submitted in partial fulfillment of the
requirements for the degree of

**DOCTOR OF PHILOSOPHY IN SOFTWARE ENGINEERING**

from the

**NAVAL POSTGRADUATE SCHOOL**
**September 2002**

Author: _____
Michael R. Murrah

Approved by:

_____
Luqi
Professor of Computer Science
Dissertation Supervisor

_____
Valdis Berzins
Professor of Computer Science

_____
Nabendu Chaki
Professor of Computer Science

_____
Dan Dolk
Professor of Information Science

_____
Lawrence Putnam
Quantitative Software Management

Approved by: _____
Christopher Eagle, Chair, Department of Computer Science

Approved by: _____
Carson K. Eoyang, Associate Provost for Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

## ABSTRACT

The Modified Risk Model is a macro model developed to aid program managers in effectively planning the required effort to deliver software products. The model projects the probability of completing a software project, subject to the available resources supplied by management. This approach to software project risk management is unique because the model's input parameters are derived. Subjective variables are not part of the model. Different program managers would derive the same projections on the same software project.

Risk management is most effective in impacting the project's success if project risks are identified and mitigated early in the software lifecycle. The Modified Risk Model was developed specifically for this purpose. Additionally, the Modified Risk Model is versatile enough to be adapted to any software development activity.

Validation of the model occurs in approximately 2,000 software projects. During these preliminary experiments, the Modified Risk Model out performed the macro models of Basic COCOMO and the Simplified Software Equation. However, to date, operational tests have not been conducted on the model.

The Modified Risk Model requires four parametric inputs, all of which are automatically collectable and derived extremely early in the software lifecycle:

- **Organization**. The MRM implements a measure to capture the efficiency of a software development organization.
- **Complexity**. The MRM architecture accommodates interface with the Computer Aided Prototyping System developed at the Naval Postgraduate School. (Dupo02) and this research are capable of deriving key complexity measures from the machine generated specification code. The MRM is capable of using different complexity measures as a "plug-ins"; thus, allowing the model to interface with organizations not equipped with CAPS.
- **Requirements**. A software project can be viewed as a finite set of issues that require resolution prior to project completion. These issues are not fully revealed in the beginning of the process. The MRM captures the stability of the known issues and adjusts projections based on the introduction or deletion of additional issues. As with the other model parameters, requirements volatility is completely adaptable to unique

software development situations. A risk analyst can choose to monitor the change in the project's risk or implement static projections.

- **Management Trade-Offs.** To successfully develop software, a balance must exist between the organization (*efficiency*), product attributes (*complexity*), and project stability (*requirements volatility*). In reality, this is not always the case. It becomes the responsibility of management to balance the equation. Management applies resources (time and people) to achieve a successful balance.

The Modified Risk Model lets management know how well balanced is the software development. The risk analyst also has the ability to derive the *Management Trade-Offs* within a confidence interval. With this information, management can implement any suitable staffing profile to achieve the model's projection.

**TABLE OF CONTENTS**

## LIST OF FIGURES

## LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

## ACKNOWLEDGEMENTS

Acknowledgements often go overlooked, but enough cannot be said about the people who provided instrumental support to the research and writing of this dissertation. I am forever grateful to the many people who believed in the possibility of me accomplishing the research.

First, I have to thank my primary advisor, Professor Luqi, who got me interested in the Ph.D. program soon after my arrival at the Naval Postgraduate School. Without her faith and persuasion, I would not have had the opportunity to pursue my doctorate. The list of support she provided is endless and a simple word of thanks is not enough.

Next, I will be forever indebted to my Ph.D. committee. Each person fulfilled a critical role towards my success. Lawrence Putnam, without your enormous generosity and resources, quite simply, this research could not have happen. Dan Dolk, your attention to detail and valuable insights have ensured the research is a quality product. Valdis Berzins, your knowledge of this material is unsurpassed, I will forever be grateful for your patience and support. Finally, Nabendu Chaki, you have always provided sound advice for the new ideas and solutions. Collectively, each of you made this possible; I will forever been in your debt.

Outside of my committee, multiple displays of encouragement were always available from Man-Tak Shing and Richard Riehle; your efforts make the Software Engineering Program first class. Man-Tak, thank you for sharing your never-ending knowledge of PSDL and CAPS; you have always been an enormous source of reference. And Richard, you have probably forgotten more software engineering knowledge than I could ever hope to learn. My thanks go out to both of you.

I saved the best for last. No words can express the love and gratitude I have towards my best friend and wife, Lesley. Your enduring encouragement and support sustained me when I doubted myself. Balance is always essential to maintain forward momentum, and you have always been instrumental in ensuring we maximize both work and family. Without you, this would not have been possible. Thank you.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. INTRODUCTION

## A. THE IMMATURITY OF SOFTWARE ENGINEERING

Over the past 40 years limited progress has been made to help practitioners project the risk associated with delivering software solutions. "Nearly every software engineering development is plagued with numerous problems leading to late delivery and cost overruns, and sometimes, unsatisfied customers" (Thay81). Today, the situation is marginally better, and in some circumstances, may be considered worse. To find a real world system today, mechanical or electrical, that does not intricately depend on software for satisfactory performance, is considered a harder task than solving Richard Thayer's point in 1981. Yet still, software developments are delivered late, over budget, or cancelled.

Recently, the risks of developing software have gained attention (Sabo97):

> The Department of Defense (DoD), in conjunction with numerous private and academic institutions, has been spending millions of dollars trying to solve the software dilemma – 30% of all projects are outright failures, 55% lack major required functionality, and are grossly over-budget and over-schedule. Almost all are seriously flawed. The pressure is real – billions of dollars are at stake.

Modernization of the DoD has resulted in an ever-increasing dependence on software. This fact can often be overlooked because of the way people interact with certain types of software (embedded software). Many times the software is critical to the safety of the system (safety critical software). Other times the software must perform under precise timing constraints and work concurrently with other functions (real time software). Despite technological advances in the software field, software development remains a costly and one of the highest risk factors on most weapon system programs.

Developing software is still a high-risk activity. Advances in technology and CASE tools provide little progress to improve the management of software development projects (Hall97). The acquisition and development communities, both in government and industry, lack systematic ways of identifying, communicating and resolving technical uncertainty (SEI96).

This dissertation helps improve this outlook. The research extends the field of software engineering by providing definitive evidence that software risk assessment can be conducted early in software development using quantifiable metrics and simple parametric techniques. Many threads of research: input metrics extension, exponential simulation runs, simulation calibration, actual projects validation, and development of an improved model, make it possible to extend the state-of-the-art. The research provides a risk assessment model that has been validated against thousands of post-mortem projects, having application in any software development activity.

## B. THE ISSUE

(Nogo00) presents a formal model for software risk assessment that is used to estimate the probability that adequate development time is scheduled for a software project. For the purpose of this dissertation, the software risk assessment model in (Nogu00) is referred to as the Software Risk Model (SRM). The SRM uses parametrics that are obtained easily and are available early in the software development process. The SRM was developed from a series of experiments conducted on the VitéProject (Levi99) simulation. This unique approach does provide a starting point towards a proven formal model for risk assessment. However, prior to this dissertation, conclusive evidence did not exist that conducting software risk estimation is possible in this manner.

The first and most notable issue when using the SRM is confidence in the model's results. This is due to three factors: the model has been in existence for a limited amount of time, the model has not been exercised on a wide base of real world projects (completed or on-going), and the model was developed using simulation techniques. The first factor noted can only be dealt with in the passage of time. However, a unique opportunity exists to impact the latter two issues.

The SRM research shows promise in estimating the associated risk when developing software systems; yet, the model has not been significantly exercised beyond theoretical simulation. Three "real world" projects to date have been projected with the

estimation model. All three of these projects were projected post-mortem. Model validity has not been demonstrated in the context targeted by its original design, estimating risk early in software project's life cycle.

A second issue in validating SRM is the required input metrics. This problem presents itself as a double-edged sword. A major attraction to using the SRM is the parametrics. These metrics are determined in a definitive, quantifiable manner and can be derived extremely early in the software development process. However, these metrics are quite unique. Currently, outside of the academic environment, it is not common practice to collect these unique metrics in the required form to utilize SRM.

In order to establish confidence in the usefulness and accuracy of the SRM, the model must be exercised against numerous projects. The ideal situation would exercise the model according to its original design; early in the software development cycle. However, the next logical step is to continue to exercise the model on a post-mortem basis. Before this can be accomplished, a mapping must be derived between the SRM parametric inputs and metrics that are frequently collected on completed projects. With the successful development of a mapping, the model can be exercised against additional projects, addressing the second point of the first issue.

The final issue associated with the SRM is the configuration of the VitéProject simulation and the suitability of VitéProject to simulate software development. This simulation derived the foundation data of the SRM and therefore must be accurately examined. It is extremely difficult to establish confidence with a simulation without previous knowledge of how software projects behave. The development of the SRM occurs through configuring VitéProject using Organizational Consultant Expert System (Nogu00). Fictitious software engineering organizations were developed to represent the typical software development organization. This seems to be a suitable approach; however, numerous additional parameters are available in the simulation that must be accounted for. Calibrating the simulation in this manner could yield different results than calibrating the simulation with actual information derived from real projects. If the SRM can be verified by reprogramming the VitéProject configuration this would provide additional assessment to the third point of the first issue.

## C. RESEARCH QUESTIONS

Models are available, as documented in (Nogu00), to project the probability (i.e. the risks) of not applying an adequate schedule to a software development project. However, little confidence is warranted in the usefulness of these models because a negligible number of validation attempts have been conducted against these models outside of simulation. This fact leads to the basic research question addressed by this dissertation:

- **How do the current risk estimation methods perform when exercised on real projects?**

In order to validate the performance of software risk models, techniques have to be developed to facilitate a mapping between real world project attributes currently available and the parametric inputs required in the Software Risk Model (SRM). The Software Risk Model was developed to support the Computer Aided Prototyping System (CAPS). However, the unique model inputs, available from the CAPS environment, are not readily available from real world projects; where a large number of projects exists. This leads to a logical second question:

- **How do the metrics required for the current risk estimation methods correlate to metrics collected in real world projects?**

Preliminary research (John01) (Alex01) (Murr02) has indicated that the Software Risk Model does not project software risks within acceptable tolerances and therefore questions arise as to the true benefit of its use. Subsequently, a third question follows the logical progression of the first two:

- **What are the necessary enhancements evolving the current risk estimation methods to provide improved results when exercised on real projects?**

## D. GENERAL RESEARCH DESIGN

A process of using simulation and real world data to validate and extend the foundation efforts provided in (Nogu00), (Boeh81), and (Putn92) characterizes this research. Despite the recent improvements provided by the previous risk assessment model, this body of work remains un-validated and little confidence is currently warranted in its use. Through validation, and enhancement where necessary, future practitioners can approach software risk assessment through formalization and systematic solutions.

To begin the validation of the software risk model, post-mortem projects are identified whose characteristics best suit successful projections. Detailed analysis is conducted to ensure a satisfactory mapping between the real world projects and the target models. To provide a performance baseline, the model validation introduces two industry standards in software cost estimation: Quantitative Software Management's (QSM®) Simplified Software Equation (SSE) (Putn92), and Boehm's Constructive Cost Model (Basic COCOMO) (Boeh81). Essentially, the validation compares four artifacts: actual performance project, SRM performance projections, SSE performance projections, and Basic COCOMO performance projections. Successful performance of the SRM will warrant no further enhancements. Ultimately, significant issues surfaced during validation that resulted in a dramatic enhancement to the SRM.

Using the models from (Putn92) and (Boeh81), a calibration model is developed to help tune the VitéProject simulation to accurately portray software development. The calibration model enhances the use of the VitéProject simulation for software development simulation, with confidence on the interpretation of the simulator results. However, the research documents severe concerns with the suitability of using VitéProject to simulate software development.

Real project data is utilized to replace the deficiencies identified in the use of VitéProject. Analysis on the real project data provides insights to software development and provides the foundation to the development of a conceptual model that extends the SRM. Modifications to the input parameters of the original model, simulation data, and evidence from the actual projects help derive a new software risk assessment model. The new model is called the Modified Risk Model (MRM).

A new risk assessment model is not any better than previous risk assessment models if its performance cannot be validated. The same rigor applied to the validation of the SRM is repeated for the validation of the MRM. However, the validation of the MRM is conducted against approximately 2,000 software projects, across eight application domains.

## E. CONTRIBUTIONS

The first contribution of this dissertation provides evidence that the mathematical implementation of the Software Risk Model (Nogu00) is not suitable for software risk projections (Chapter IV). The SRM projects software completion times dangerously optimistic. Furthermore, the SRM projects software completion times in a bi-polar fashion; one of two possible projections is produced depending on the *efficiency* of the organization. To achieve this contribution, a successful mapping was developed between the unique input parameters of the SRM and projects in the real world.

The second contribution of this dissertation, and probably more important than the first, is the overwhelming evidence suggesting the VitéProject simulation is not suitable for simulating software development (Chapter V). As documented, the SRM is developed and validated with a single point of failure, the VitéProject simulation. Subsequently, for every flaw identified in the mathematical representation of the SRM, the origins can be traced back to VitéProject.

The third contribution of this dissertation is a collection of techniques that can be utilized to calibrate future simulations and software development models. Chapter V details the development of software development benchmarks. These benchmarks are

useful for comparing the performance of simulations and model projections. Additionally, a technique is provided to tune VitéProject so that researchers can confidently interpret the results from the simulation. However, these results are still subject to the overwhelming evidence presented in contribution two.

The final, and most significant, contribution is the development and validation of the enhanced risk model called the Modified Risk Model. This model promotes the best features of the software risk model and improves its shortcomings. Development of the MRM did not depend on the performance of the VitéProject simulation. Complexity issues in the SRM are addressed and rectified in the MRM. The MRM provides techniques to aid researchers and program managers in using the MRM in any application domain.

## F. ORGANIZATION OF DISSERTATION

Chapter II of this dissertation presents the theoretical foundations for this research. Chapter III describes the conceptual framework of the research and includes the detailed approach to accomplishing the validation – enhancement – validation process. Chapter IV presents the details and methodology used to interface the actual projects to the SRM and provides the validation of the SRM; presenting evidence that an improved model is necessary.

Chapter V discusses the details implementing and calibrating VitéProject for software development. Chapter VI details the development of the MRM resulting from introducing additional projects. Chapter VII documents the validation of the MRM by exercising four artifacts: actual project data, MRM, SSE, and Basic COCOMO. Due to the extensive enhancements in the MRM, Chapter VII revisits the metrics mapping from Chapter IV. Chapter VIII leaves the reader with conclusions and directions for future research. Following the base chapters in this dissertation, appendices are included to detail the data dictionary, project samples, and references. Additionally, Appendices C – F directly support the research presented in Chapters IV – VII respectively.

THIS PAGE INTENTIONALLY LEFT BLANK

# II. THEORETICAL FOUNDATION

## A. RISK AND UNCERTAINTY

Developing software is still a high-risk activity. As surveyed in (Nogu00), research shows that 45 percent of all delayed software deliveries are related to organization issues (vanG91). Software is the main expense in computer systems (Boeh81), (Karo96). Besides the improvements in tools and methodologies, there is little evidence of success in improving the process of moving from the concept to the product. A study published by the Standish Group reveals that the number of software projects that fail has dropped from 40% in 1997 to 26% in 1999. However, the percentage of projects with cost and schedule overruns rose from 33% in 1997 to 46% in 1999 (Reel99).

The DoD recognizes that the management of software risk is the same as management of other types of risk and techniques that apply to hardware programs are equally applicable to software intensive programs (DSMC01). However, some characteristics of software make this type of risk management different primarily because it is difficult to:

- Identify software risk
- Estimate the time and resources required to develop new software, resulting in potential risks in cost and schedule
- Test software completely because of the number of paths that can be followed in the logic of the software
- Develop new programs because of the rapid changes in information technology and an ever-increasing demand for quality software personnel

The key to successful risk management is early planning and aggressive execution. Effective risk management requires involvement from every member in the development process. Risk management should not be looked upon as an additional task

or a separate task for team members to perform. Risk management is a mindset, a culture that must exist in every process performed and project delivered.

(DSMC01) presents a concise representation of risk terminology which is supported in this research:

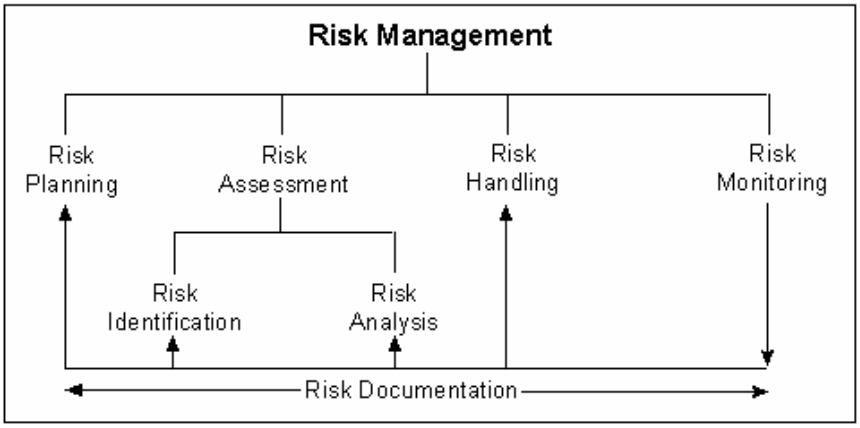



**Figure II-1. Risk Management Structure and Definitions, "DSMC01".**

***Risk*** is a measure of the potential inability to achieve overall program objectives within defined cost, schedule, and technical constraints and has two components: (1) the *probability/likelihood* of failing to achieve a particular outcome and (2) the *consequences/impacts* of failing to achieve that outcome.

***Risk events*** **(**i.e., things that could go wrong for a program or system) are elements of an acquisition program that should be assessed to determine the level of risk. The events should be defined to a level that an individual can comprehend the potential impact and its causes. For example, a potential risk event for a turbine engine could be turbine blade vibration. There could be a series of potential risk events that should be selected, examined, and assessed by subject-matter experts.

The relationship between the two components of risk -- probability and consequence/impact -- is complex. To avoid obscuring the results of an assessment, the risk associated with an event should be characterized in terms of its two components. As part of the assessment there is also a need for backup documentation containing the supporting data and assessment rationale.

***Risk management*** is the act or practice of dealing with risk. It includes planning for risk, assessing (identifying and analyzing) risk areas, developing risk-handling options, monitoring risks to determine how risks have changed, and documenting the overall risk management program.

***Risk planning*** is the process of developing and documenting an organized, comprehensive, and interactive strategy and methods for identifying and tracking risk areas, developing risk-handling plans, performing continuous risk assessments to determine how risks have changed, and assigning adequate resources.

***Risk assessment*** is the process of identifying and analyzing program areas and critical technical process risks to increase the probability/likelihood of meeting cost, schedule, and performance objectives. *Risk identification* is the process of examining the program areas and each critical technical process to identify and document the associated risk. *Risk analysis* is the process of examining each identified risk area or process to refine the description of the risk, isolating the cause, and determining the effects. It includes risk rating and prioritization in which risk events are defined in terms of their probability of occurrence, severity of consequence/impact, and relationship to other risk areas or processes.

***Risk handling*** is the process that identifies, evaluates, selects, and implements options in order to set risk at acceptable levels given program constraints and objectives. This includes the specifics on what should be done, when it should be accomplished, who is responsible, and associated cost and schedule. The most appropriate strategy is selected from these handling options. *Risk handling* is an all-encompassing term whereas risk mitigation is one subset of risk handling.

***Risk monitoring*** is the process that systematically tracks and evaluates the performance of risk handling actions against established metrics throughout the acquisition process and develops further risk handling options, as appropriate. It feeds information back into the other risk management activities of planning, assessment, and handling as shown in Figure II-1.

***Risk documentation*** is recording, maintaining, and reporting assessments, handling analysis and plans, and monitoring results. It includes all plans, reports for the program manager and decision authorities, and reporting forms that may be internal to the developing agency.

## B. IEEE STANDARD FOR SOFTWARE LIFE CYCLE PROCESSES

In March 2001, IEEE published IEEE Std 1540-2001, which may be used independently of any particular software life cycle process standard, or in conjunction with IEEE/EIA 12207.0-1996. Although IEEE/EIA 12207.0-1996 describes a standard process for the acquisition, supply, development, operations, and maintenance of software, this standard does not provide a process for risk management. IEEE Std 1540-2001 risk management standard provides that process.

The purpose of risk management is to identify and mitigate the risks continuously (IEEE01). The risk management process is a continuous process for systematically addressing risk throughout the life cycle of a product or service. IEEE Std 1540-2001 describes this process consists of the following activities:

- Plan and implement risk management
- Manage the project risk profile
- Perform risk analysis
- Perform risk monitoring
- Perform risk treatment
- Evaluate the risk management process

Figure II-2 illustrates the IEEE Std 1540-2001 risk management process. This model and the discussion following is an extract from the standard.

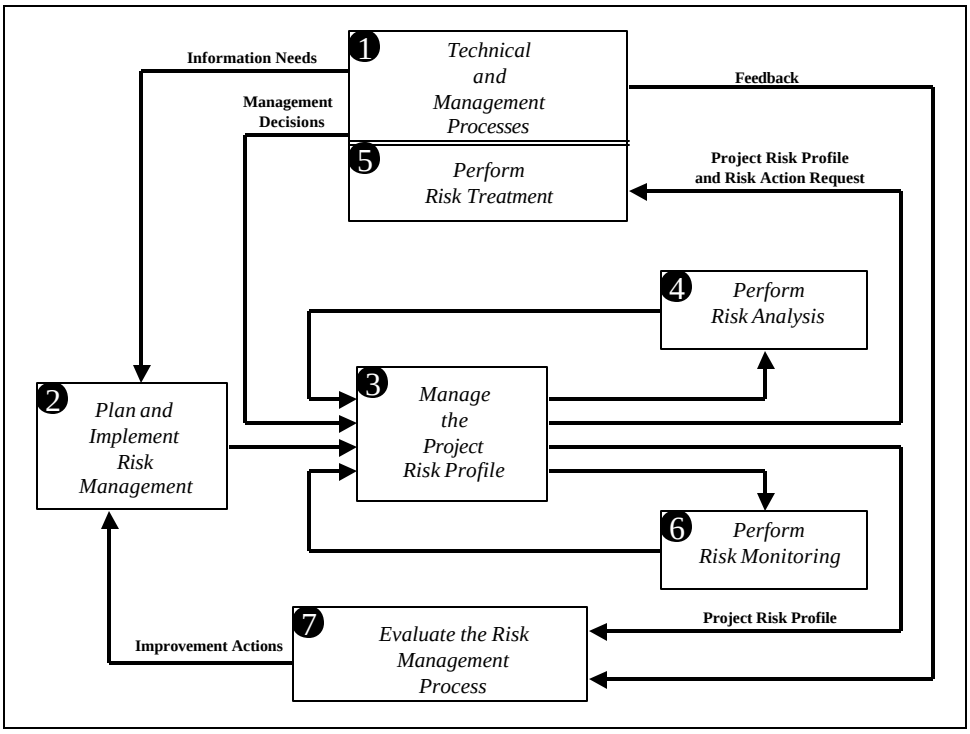



**Figure II-2. IEEE Risk Management Process Model.**

Managerial and technical processes involving the stakeholders define the information requirements (i.e., the information the stakeholders require to make informed decision involving risks), as the risk management process must support ❶. These information requirements are passed to both the "plan and implement risks management" and the "manage the project risk profile" activities. In the "plan and implement risk management" activity ❷, the policies regarding the general guidelines under which risk management will be conducted, the procedures to be used, the specific techniques to be applied, and so forth, are defined.

In the "manage the project risk profile" activity ❸, the current and historical risk management context and risk state information are captured. The project risk profile includes the sum total of all the individual risk profiles (i.e., the current and historical risk information concerning and individual risk), which, in turn, includes all the risk states.

The project risk profile information is continually updated and maintained through "perform risk analysis" activity ❹, which identifies the risks, determines the likelihood and consequences, determines the risk exposures, and prepares risk action requests recommending treatment for risks determined to be above the risk threshold(s).

Treatment recommendations, along with the status of other risks and the treatment status, are sent to management for review ❺. Management decides what risk treatment is implemented for any risk found to be unacceptable. Risk treatment plans are created for risks that require treatment. These plans are coordinated with other management plans and other ongoing activities.

All risks are continually monitored until there no longer is a need to track during the "perform risk monitoring" activity ❻. In addition, new risks are sought out.

Periodic evaluation of the risk management process is required to ensure its effectiveness. During the "evaluate the risk management process" activity ❼, information, including user and other feedback, is captured for improving the process or for improving the organization's or project's ability to manage risk. Improvements defined as a result of evaluation are implemented in the "plan and implement risk management" activity ❷.

The software risk management process is applied continuously throughout the product life cycle. However, activities and tasks of the risk management process interact with the individual risks in an interactive manner once the risk management process begins. For example, in the "perform risk analysis" activity ❹, a risk may be re-estimated several times during the performance of risk evaluation due to an increase in knowledge about the risk gained during the evaluation task itself. The risk management process is not a "waterfall" process.

Although the IEEE Risk Management Process Model provides a framework for organizations to conduct risk management, it fails to address the basic research questions surfaced in this research; identifying quantifiable early collectable metrics in the software process and utilizing these metrics to assess project risk.

Results of this research have applicability in the IEEE 1540-2001 standard. Section 5.1.3, Perform Risk Analysis, describes activities that should be performed to conform to the standard (IEEE01). These activities are listed below:

- Identify the initiating events, hazards, threats, or situations that create risks
- Estimate the likelihood of occurrence, the consequences for each risk, and the expected timing of the risk

- Evaluate each risk or defined combination of risks against its applicable threshold, generate alternatives to treat risks above the risk thresholds, and make recommendations for treatment based on a priority order

This research delivers an enhanced model that provides quantifiable metrics to identify risk and to estimate the impact of these risks on software projects.

## C. DOD RISK MANAGEMENT POLICIES AND PROCEDURES

The DoD produces risk management policy guidance to assist program managers in acquiring and developing software intensive systems. Risk management guidance appears in five key DoD documents:

- DoD Directive (DoDD) 5000.1, the Defense Acquisition System
- DoD Instruction (DoDI) 5000.2, Operation of the Defense Acquisition System
- DoD Regulation 5000.2-R (Interim), Mandatory Procedures for Major Defense Acquisition (MDAPs) and Major Automated Information System (MAIS) Acquisition Programs
- DoD Directive 5000.4, OSD Cost Analysis Improvement Group
- DoD Manual 5000.4-M, Cost Analysis Guidance and Procedures

Each of these five documents presents the strong need for conducting risk management. However, collectively, the guidance is not sufficient to enable the establishment of an effective risk management program. The following are applicable verbatim extracts of sections of the DoD 5000 series of documents that address risk management as part or acquiring software intensive systems.

**1. DoDD 5000.1**

**The Defense Acquisition System, 23 October 2000**

**Para 4.5.4. Simulation-Based Acquisition.** Program managers shall plan and budget for effective use of modeling and simulation to reduce the time, resources, and risk associated with the entire acquisition process; increase the quality, military worth, and supportability of fielded systems; and reduce total ownership costs throughout the system life cycle.

**2. DoD Instruction 5000.2.**

**Operation of the Defense Acquisition System, 23 October 2000**

**(w/Chg 1 d&d 4 Jan 2001)**

**Para 4.7.3.2.1.1. Begin Development and Develop and Demonstrate Systems – General.** The purpose of the System Development and Demonstration phase is to develop a system, reduce program risk, ensure operational supportability, design for producibility, ensure affordability, ensure protection of Critical Program Information, and demonstrate system integration, interoperability, and utility.

**Para 4.7.3.2.3.4.1 Entry into System Development and Demonstration.** Milestone B approval can lead to System Integration or System Demonstration. Regardless of the approach recommended, PMs and other acquisition managers shall continually assess program risks. Risks must be well understood, and risk management approaches developed, before decision authorities can authorize a program to proceed into the next phase of the acquisition process. Risk management is an organized method of identifying and measuring risk and developing, selecting, and managing options for handling these risks. The types of risk include, but are not limited to, schedule, cost, technical feasibility, threat, risk of technical obsolescence, security, software management,

dependencies between a new program and other programs, and risk of creating a monopoly for future procurements.

**Para 4.7.3.3.2.1. Entrance Criteria.** Technology maturity (with an independent technology readiness assessment), system and relevant mission area (operational) architectures, mature software capability, demonstrated system integration or demonstrated commercial products in a relevant environment, and no significant manufacturing risks.

### 3. DoD Regulation 5000.2-R.

**Mandatory Procedures for Major Defense Acquisition Programs (MDAPs) and Major Automated Information System (MAIS) Acquisition Programs (Interim Regulation), 4 January 2001**

**Para 2.5 Risk.** The acquisition strategy shall address risk management. The PM shall identify the risk areas of the program and integrate risk management within overall program management. The strategy shall explain how the risk management effort shall reduce system-level risk to acceptable levels by the interim progress review preceding system demonstration and by Milestone C.

**Para 3.1 Test and Evaluation (T&E) Overview.** The T&E strategy shall provide information about risk and risk mitigation, provide empirical data to validate models and simulations, evaluate technical performance and system maturity, and determine whether systems are operationally effective, suitable, and survivable against the threat detailed in the System Threat Assessment (see 6.2.2).

**Para 4.2 Analysis of Alternatives (AoA).** Analyzing alternatives is part of the Cost as an Independent Variable process. Alternatives analysis shall broadly examine multiple elements of project or program alternatives including technical risk and maturity, price, and costs.

**Para 5.2.5 Open Systems Design.** PMs shall use an open systems approach to achieve the following objectives;

- To mitigate the risks associated with technology obsolescence, being locked into proprietary technology, and reliance on a single source of supply over the life of a system;

**Para 5.2.6 Software Management.** The PM shall manage and engineer software-intensive systems using best processes and practices known to reduce cost, schedule, and performance risks.

**Para 5.2.6.1 General.** The PM shall base software systems design and development on systems engineering principles, to include the following:

- Select the programming language in context of the systems and software engineering factors that influence overall life-cycle costs, risks, and the potential for interoperability;…
- …However, if the prospective contractor does not meet full compliance, a risk mitigation plan and schedule shall be prepared to describe, in detail, actions that will be taken to remove deficiencies uncovered in the evaluation process. The risk mitigation plan shall require PM approval.…
- Assess information operations risks (DoDD S-3600.1) using techniques such as independent expert reviews;…

**Para 5.2.6.3 Review of Software-Intensive Programs.** An independent expert review team shall review programs and report on technology and development risk, cost, schedule, design, development, project management processes and the application of systems and software engineering best practices.

**Para 5.2.6.4 Software Security Considerations.** The following security considerations apply to software management:

- When employing COTS software, the contracting process shall give preference during product selection/evaluation to those vendors who can demonstrate that they took efforts to minimize the security risks associated with foreign nationals that have developed, modified or remediated the COTS software being offered.…

**Para 5.2.7 COTS Considerations.** The use of commercial items often requires changes in the way systems are conceived, acquired, and sustained, to include:…

- The PM shall develop an appropriate T&E strategy for commercial items to include evaluating potential commercial items in a system test bed, when practical; focusing test beds on high-risk items; and testing commercial-item upgrades for unanticipated side effects in areas such as security, safety, reliability, and performance.…
- Programs are encouraged to use code-scanning tools, within the scope and limitations of the licensing agreements, to ensure both COTS and GOTS software do not pose any information assurance or security **risks**.

**4. DoD Directive (DoDD) 5000.4.**

**OSD Cost Analysis Improvement Group (CAIG), November 24, 1992**

**Para D.1.h Risk Assessment.** The CAIG Chair report, in support of a milestone review, shall include quantitative assessments of the risk in the estimate of life-cycle costs. In developing an assessment of cost risk, the CAIG shall consider the validity of such programmatic assumptions of the CARDs as EMD schedules, rates of utilization of test assets, production ramp rates, and buy rates, consistent with historical information. The CAIG shall also consider uncertainties in in-puts to any cost estimating relationships used in its estimates, as well as the uncertainties inherent in the calibration of the CERs, and shall consider uncertainties in the factors used in making any estimates by analogy. The CAIG shall consider cost and schedule risk implications of available assessments of

the program's technical risks, and may include the results in its cost-risk assessments. The CAIG may consider information on risk provided by any source, although primary reliance will be on the technical risk assessments that are the responsibility of the sponsoring DoD components, and of other OSD offices, in accordance with their functional responsibilities.

**5. DoD 5000.4-M.**

**Cost Analysis Guidance and Procedures, December 1992**

**Chapter 2: Para 2.0 Risk.** This section identifies the program manager's assessment of the program and the measures being taken or planned to reduce those risks. Relevant sources of risk include: design concept, technology development, test requirements, schedule, acquisition strategy, funding availability, contract stability, or any other aspect that might cause a significant deviation from the planned program. Any related external technology programs (planned or on-going) should be identified, their potential contribution to the program described, and their funding prospects and potential for success assessed. This section should identify these risks for each acquisition phase (DEM/VAL, EMD, productions and deployment, and O& S).

**Para 2.B.9 Sensitivity Analysis.** The sensitivity of projected costs to critical program assumptions shall be examined. Aspects of the program to be subjected to sensitivity analysis shall be identified in the DoD CCA of program assumptions. The analysis shall include factors such as learning curve assumptions; technical risk, (i.e., the risk of more development and/or production effort, changes in performance characteristics, schedule alterations, and variations in testing requirements; and acquisition strategy (multiyear procurement, dual sourcing, etc.).)

## D. ESTIMATION MODELS AND TOOLS

### 1. Nogueira

(Nogu00) developed four software risk estimation models that show promise in determining a software projects' associated risk early in the software development life cycle. The models accomplish early estimation by utilizing a set of quantifiable metrics that can be collected from the beginning of project development. In actuality, the requirements volatility metric is estimated during the first development cycle and during subsequent development cycles is quantifiable. After each software development iteration, the input metrics can be updated to dynamically reduce the error in the model's projections. (Nogu00) documents that the SRM can project software development completion times within the following ranges of accuracy.

- Maximum error = 127 days (28%)
- Average error = 19 days
- Error standard deviation = 23 days
- 5% of projects with error >= 25%
- 65% with 5% < error < 25%
- 30% with error < 5%

The minimum required parametric inputs, to support risk assessment in the SRM are the following:

#### *a. Efficiency (EF)*

The *efficiency* of the organization is determined by observing the fit between people and their roles (Nogu00). Dr. Nogueira's indicates that the *efficiency* of an organization can be directly calculated by computing the ratio of direct time (working and correcting errors) divided by the idle time (time spent without work to do).

### *b. Requirements Volatility (RV)*

Requirements volatility expresses how difficult the requirement elicitation process is. Derive the requirements volatility by implementing the following formula (Nogu00).

**Requirements Volatility = Birth Rate Percentage + Death Rate Percentage**

Birth Rate Percentage (BR%) = the percentage of new requirements incorporated in each cycle of the software evolution process as calculated by:

**BR% = (New Requirements / Total Requirements) * 100 percent**

Death Rate Percentage (DR%) = the percentage of requirements that are dropped by the customer in each cycle of the evolution process as calculated by:

**DR% = (Deleted Requirements / Total Requirements) * 100 percent**

### *c. Complexity (CX)*

Complexity has a direct impact on quality because the likelihood that a component fails is directly related to its complexity (Nogu00). The complexity metrics can be determined in two forms: *Large Granular Complexity* and *Fine Granular Complexity.* These two forms of complexity can be directly determined from software specifications written in the Prototype System Description Language (PSDL) (Luqi90).

Large Granular Complexity (LGC) expresses the relational complexity of the system as a function of the number of operators (O), data streams (D), and types (T)

$$\mathbf{LGC = O + D + T}$$

Fine Granular Complexity (FGC) expresses the relational complexity of each operator in the system and is a function of the fan-in and fan-out data streams related to the operator (Nogu00).

$$\mathbf{FGC = fan\text{-}in + fan\text{-}out}$$

(Nogu00) serves as the foundational basis for this research. As addressed in Chapter I, the SRM has issues of its own. The rest of this dissertation demonstrates how to exploit the strengths of the SRM and mitigate its issues.

#### *d. Generic Model Behavior*

Figure II-3 and Figure II-4 demonstrates the generic behavior that can be expected with using the SRM. These figures were established using assumption distributions (Chapter IV). For the analysis, 10,000 unique simulations were executed using a Monte Carlo simulator. The resulting forecast is presented in Figure II-3 below.

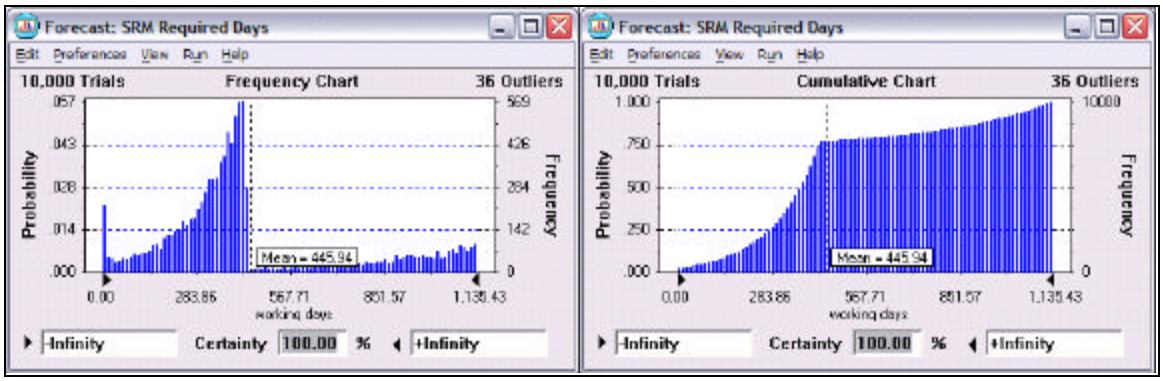



**Figure II-3. Simulated Software Risk Model.**

The distribution on the left side of Figure II-3 illustrates the frequency distribution of the 10,000 simulations. The right-side illustration is the cumulative distribution. Figure II-3 illustrates some intriguing behavior of the software risk model. Specifically, with these assumptions (the distribution limits), there exist a strong probability of projecting a requirement of zero days to complete the project.

More interesting, is the steep reduction in frequency around 440 days. This apparent discontinuity requires additional investigation. Figure II-4 below illustrates that approximately 77% percent of all of the projects were projected to be complete by the mean amount of days (446). Essentially, this indicates that only 23% of the projects will ever extend past 20 months[1].

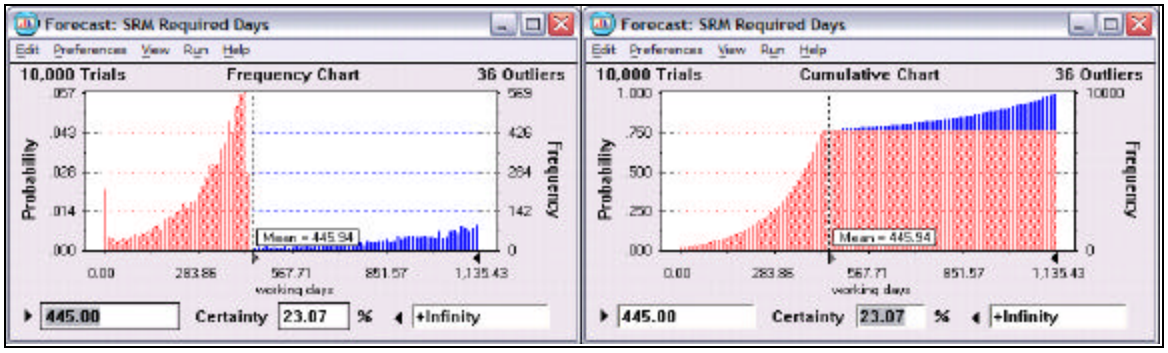



**Figure II-4. Simulated Software Risk Model at the Mean.**

[1] (Nogu00) indicates that a calendar month is equivalent to 22 working days.

The distribution of the 10,000 projects indicates the possibility of two distinct situations. First, there exist subsets of input assumptions that cause the software risk model to project in the range of zero days to approximately 440. Second, an additional subset of input assumptions causes the Software Risk Model to extend the range of possibilities out through 1,135 days (4.2 years). The analysis on the Software Risk Model behavior is curious in the least. Chapters III and IV continue to examine the behavior of the SRM.

### 2. Putnam

Lawrence H. Putnam, Sr. is a world-renowned author, lecturer, and consultant on software productivity, quality, and lifecycle estimating and has written over 30 technical papers and four books on the subject. He founded Quantitative Software Management (QSM®), Inc. in 1978. QSM® is the most senior software management firm in the industry (QSM00).

QSM®'s methods, tools, and training provide pro-active solutions for software productivity measurement and improvement, cost and schedule estimating, size estimating, and runaway project prevention. Many of the world's major software producers are represented in the QSM® historical project database. Companies who use QSM®'s services and products come from a variety of industry: micro code development, real-time avionics and weapons systems, process control and systems software to large financial, banking and MIS systems.

QSM® has dedicated over twenty years to collecting and analyzing data from software development projects. During this time, QSM® has assembled detailed data on more than 5,000 software systems developed since 1980 (Putn92).

QSM® developed and maintains software tools to help companies manage their software development effort. These tools are incorporated into what QSM® calls the SLIM (Software LIfecycle Management) suite of tools. SLIM encapsulates the essence of QSM®'s service to the industry. Utilizing these tools, along with access to the projects

stored in QSM®'s software project database, provides a unique opportunity to experiment and validate the SRM. The tools available from QSM® are the following (QSM00):

**SLIM Metrics**. SLIM Metrics provides the user a comprehensive and customizable platform to query, conduct statistical analysis, graphing, and reporting tools necessary to assess and compare the performance of projects.

**SLIM Control**. SLIM Control is a management tool for project tracking, forecasting and presentation of software project performance while under development. The core measurement and comparisons include: schedule, effort, cost, staffing and reliability.

**SLIM Estimate**. SLIM Estimate is a management tool for estimation, analysis and presentation of software project characteristics. These characteristics include: schedule, effort and quality. Each estimate has an associated probability range.

**SLIM Master Plan**. SLIM Master Plan is an analysis tool that accepts time-series measurement data from multiple sources and provides a single chart portrayal of each measurement, either by individual source or in aggregate.

This dissertation does **not** use the equations portrayed in the SLIM Tool Suite. Furthermore, the SLIM Tool Suite is **not** used to determine **any** project *durations* or required *effort*. The SLIM Tool Suite conducts specialized analysis on software development. The level of detail provided by SLIM is derived from detailed information that is traditionally not available for the rough-order-of-magnitude projections used in the validation of the SRM or MRM. This research used the SLIM Tool Suite to facilitate the extraction of project attributes from the QSM® database. For **all** calculations representing Putnam's Software Equation, this dissertation implements a simplified version, one that can be utilized without the SLIM Tool Suite and one that depends on industry averages for key input information.

## 3. Simplified Software Equation

### *a. Background*

Peter Norden of IBM Development Laboratory, in Poughkeepsie, New York, developed a life-cycle manpower model in the 1950s. The model projected the required manpower necessary over a hardware project's development period. He rationalized that since manpower is a large fraction of the total cost in software development, the model could also project expenditure rates. Norden found that this manpower curve could also be represented by an algebraic equation, enabling an estimator the ability to make useful computations involving project duration and effort. The equation used a special instance of the Weibull family of curves, named after the 19th century physicist Lord Rayleigh (Putn96):

$$y = 2Kate^{-at^2}$$

where

$y$ = manpower rate at each point on the curve (such as people per month)

$K$ = effort (such as person months), which is the area under curve

$t$ = development time

$a$ = a constant governing the time to peak manpower

Putnam, an alumnus of the Naval Postgraduate School, introduced in the 1970's a model applying the Norden's concepts. The use of the Rayleigh curve, as documented by (Putn80) and (Boeh81) is a reasonably good fit for the manpower projections. Putnam observed that a strong correlation between lines of code and schedule, manpower and defects exists (Nogu00). Putnam's model is based on the following assumptions (Lond87):

- A development project is a finite sequence of purposeful, temporally ordered activities, operating on a homogeneous set of problem elements, to meet a specified set of objectives

- The number of problem elements is unknown but finite
- Problems are detected, recognized and solved by applying effort
- The occurrence of problem solving follows a Poisson process
- The number of people working in the project is proportional to the number of problems to be resolved at that time

### *b. Equations*

The following is a summary of Putnam's equations utilized in this dissertation. These equations represent the basic algorithmic structure; however, the equations need to be calibrated to the specific organization to maximize the accuracy in the projections. These are reproduced from (Putn92):

- Software Equation

$$\mathrm{PP} = \frac{\mathrm{SLOC}}{(\frac{E}{B})^{\frac{1}{3}} * t_d^{\frac{4}{3}}}$$

- Minimum time to do a project in months

$$t_{d\text{-min}} = 8.14(\frac{\mathrm{SLOC}}{\mathrm{PP}})^{0.43}$$

- Expected effort at minimum time, in man-months

$$E = 180 B t_{\text{d-min}}^3$$

- Peak man-power, in people

$$Y'_{\max} = 1.5\frac{E}{t_{d-\min}}$$

- Average man-power, in people

$$Y'_{ave} = \frac{E}{t_{d-\min}}$$

- Minimum time for functional design, months

$$t_{func} = \frac{t_{d-\min}}{3}$$

- Minimum effort for functional design, man-months

$$E_{func} = \mathrm{Fr}(E)$$

where

- PP = productivity parameter

- SLOC = source lines of code
- $E$ = effort, man-months
- $B$ = special skills factor (table value)
- $t_d$ = minimum time to develop the main build
- $Y'_{max}$ = the maximum man power
- $Y'_{ave}$ = the average man power
- Fr($E$) = a fraction of the effort (table value)

Although still in use, this model is not generally believed to be especially accurate by authors such as (Cont86). One of the most criticized features is its prediction that development effort scales inversely as the fourth power of the development time, leading to severe cost increases for compressed schedules (Stut96). However, there relation of the Software Equation has been verified with data from more than 5,000 projects. Hundreds of software organizations have used the Software Equation to project schedules and effort. The estimated schedule has typically been within plus or minus ten percent of the eventual actual schedule. The variance of estimated effort has been somewhat greater, but generally less than 20 percent. The Software Equation has stood the test of time (Putn96).

### *c. Generic Model Behavior*

Figure II-5 and Figure II-6 demonstrates the generic behavior that can be expected with using the Simplified Software Equation (SSE). These figures were established by projecting a *systems software* development, using the same assumptions that provided the inputs to Figure II-3. For the analysis, 10,000 unique simulations were executed using a Monte Carlo simulator. The resulting forecast is presented in Figure II-5 below.

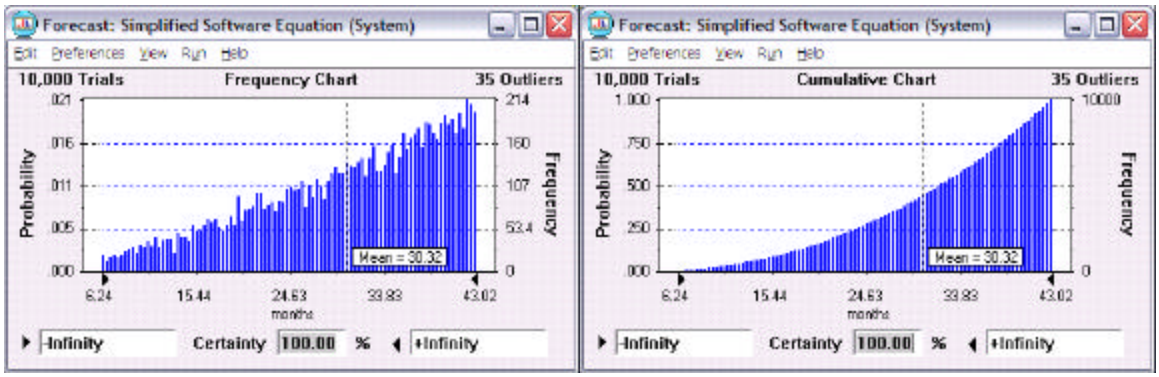



**Figure II-5. Simulated Simplified Software Equation.**

The distribution on the left side of Figure II-5 illustrates the frequency distribution of the 10,000 simulations. The right-side illustration is the cumulative distribution.

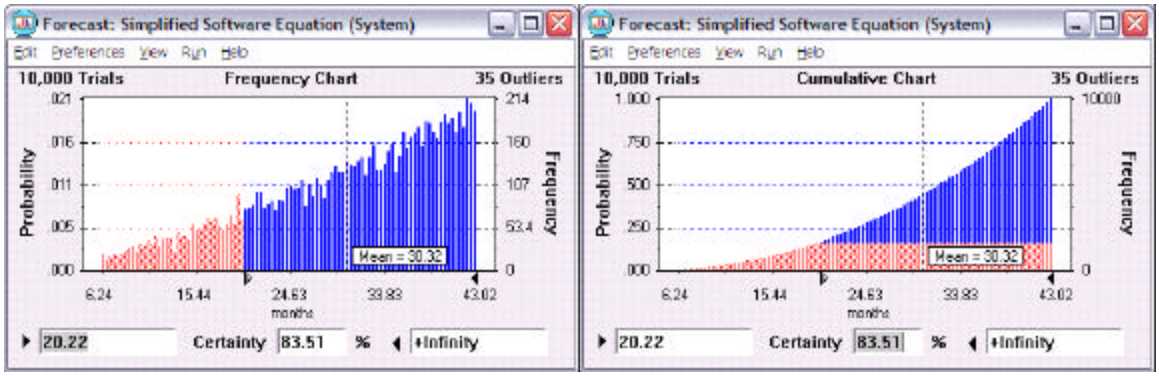



**Figure II-6. Simulated Simplified Software Equation at the Mean.**

- The Simplified Software Equation projects a smooth projection of the project durations (mean 30 months). Figure II-6 overlays the mean time projected from the SRM (Figure II-3). Approximately 83% of the project durations are above the average projection of the SRM; indicating that the SRM could be projecting project durations too optimistically.

The (PEH99) provides a concise listing of three families of parametric commercial software models. The following is an extract of the handbook. There are many sophisticated parametric software estimating models that use multiple parameters

to compute software costs and effort. These next three models in this section discuss three common software parametric models and provide a basic understanding of some common attributes. The three models are the Constructive Cost Model (COCOMO), PRICE Software Model (PRICE S®, and Galorath Software Evaluation and Estimation of Resources Software Estimating Model (SEER-SEM®). For each of these models the background, principal inputs (i.e., parameters), and principal outputs are discussed.

## 4. COCOMO

Barry Boehm, formerly of the TRW Corporation, developed the COCOMO parametric software model and published its first edition in 1981. COCOMO is not a proprietary model and is completely described in (Boeh81). In actuality, only a pocket calculator with an exponential key is required to execute COCOMO, although many computerized versions are available in the marketplace. During the last several years, Dr. Boehm *et al* have made substantial revisions to COCOMO.

### *a. COCOMO 81*

COCOMO 81 (the first version of COCOMO) is a regression-based model that considers 63 programs in three modes depending on the development environment: *embedded*, *semi-detached*, and *organic*. Separate equations relating the *effort* in man-months (MM) to program size in thousands of delivered source instructions (KDSI) are established for each mode.

There are three levels within COCOMO 81: *basic*, *intermediate*, and *detailed*. The *basic* level consists of three simple models with single parameter *effort* and *schedule* equations. Effort equations relate MM to KDSI; and the schedule equations relate months of time needed (M) to MM, computed from the effort equations. The *basic* level of COCOMO is often used to develop rough-order-of-magnitude (ROM) estimates for software costs and is used exclusively in this dissertation. Figure II-7 lists effort and schedule equations for the three modes of Basic COCOMO 81 used in this dissertation.

The Intermediate COCOMO 81 contains nominal equations, also shown in Figure II-7, which are similar to the basic equations for each mode (except for different effort coefficients). The Intermediate COCOMO contains 15 multipliers, which adjust the result of the nominal equations to reflect a program's unique attributes.

Finally, the Detailed COCOMO 81 adjusts the multipliers for each phase of the software life cycle. All COCOMO levels are designed for use on any program. Also, all three COCOMO 81 levels allow an adjustment to KDSI for reused or modified software. COCOMO 81 computes "effective KDSI" based on the percentages of redesign, recoding, and retesting required.

| Mode | Basic *Effort* | Basic and Nominal *Intermediate Schedule* | Nominal *Intermediate Effort* |
|---|---|---|---|
| **ORGANIC** | $MM = 2.4 (KDSI)^{1.05}$ | $M = 2.5 (MM)^{0.38}$ | $MM = 3.2 (KDSI)^{1.05}$ |
| **Semi-Detached** | $MM = 3.0 (KDSI)^{1.12}$ | $M = 2.5 (MM)^{0.35}$ | $MM = 3.0 (KDSI)^{1.12}$ |
| **Embedded** | $MM = 3.6 (KDSI)^{1.20}$ | $M = 2.5 (MM)^{0.32}$ | $MM = 2.8 (KDSI)^{1.20}$ |

**Figure II-7. Elementary COCOMO 81 Equations.**

### *b. COCOMO 81 Inputs*

The following discussion focuses on the Intermediate level of COCOMO 81. The primary Intermediate COCOMO input (i.e., cost driver) is program size, in KDSI. However, there are 15 additional attributes that must be assessed. These attributes, shown in Figure II-8, are classified into the following four categories:

- **Product Attributes**. Product attributes describe the environment the program operates in. The three attributes in this category are: Reliability Requirements (RELY), Database Size (DATA), and Product Complexity (CPLX).
- **Computer Attributes**. Computer attributes describe the relationship between a program and its host or developmental computer. The four

attributes in this category are: Execution Time Constraints (TIME), Main Storage Constraints (STOR), Virtual Machine Volatility (VIRT), and Computer Turnaround Time (TURN).

- **Personnel Attributes**. Personnel attributes describe the capability and experience of personnel assigned to the program. The five attributes in this category include: Analyst Capability (ACAP), Applications Experience (AEXP), Programmer Capability (PCAP), Programming Language Experience (LEXP), and Virtual Machine Experience (VEXP).
- **Project Attributes**. Project attributes describe selected project management facets of a program. The three attributes in this category include: Use of Modern Programming Practices (MODP), Use of Software Tools (TOOL), and Required Development Schedule (SCED).

| | Rating | | | | | |
|---|---|---|---|---|---|---|
| **Attributes** | **VL** | **LO** | **NM** | **HI** | **VH** | **XH** |
| Required Reliability (RELY) | 0.75 | 0.88 | 1.00 | 1.15 | 1.40 | |
| Database Size (DATA) | | 0.94 | 1.00 | 1.08 | 1.16 | |
| Product Complexity (CPLX) | 0.70 | 0.85 | 1.00 | 1.15 | 1.30 | 1.65 |
| Execution Time Constraints (TIME) | | | 1.00 | 1.11 | 1.30 | 1.66 |
| Main Storage Constraint (STOR) | | | 1.00 | 1.06 | 1.21 | 1.56 |
| Virtual Machine Volatility (VIRT) | | 0.87 | 1.00 | 1.15 | 1.30 | |
| Computer Turnaround Time (TURN) | | 0.87 | 1.00 | 1.07 | 1.15 | |
| Analyst Capability (ACAP) | 1.46 | 1.19 | 1.00 | 0.86 | 0.71 | |
| Applications Experience (AEXP) | 1.29 | 1.13 | 1.00 | 0.91 | 0.82 | |
| Programmer Capability (PCAP) | 1.42 | 1.17 | 1.00 | 0.86 | 0.70 | |
| Virtual Machine Experience (VEXP) | 1.21 | 1.10 | 1.00 | 0.90 | | |
| Programming Language Experience (LEXP) | 1.14 | 1.07 | 1.00 | 0.95 | | |
| Use of Modern Programming Practices (MODP) | 1.24 | 1.10 | 1.00 | 0.91 | 0.82 | |
| Use of Software Tools (TOOL) | 1.24 | 1.10 | 1.00 | 0.91 | 0.83 | |
| Required Development Schedule (SCED) | 1.23 | 1.08 | 1.00 | 1.04 | 1.10 | |

**Figure II-8. Intermediate COCOMO 81 Effort Multipliers.**

Tables included in Chapter VIII of (Boeh81) describe ratings from "very low" to "extremely high," that must be assessed for each of the 15 attributes identified above. For example, the reliability factor (i.e., RELY) would be rated "very low" if an error in a specific software system caused only slight inconvenience; "nominal" if an error could result in moderate, recoverable losses; or "very high" if potential loss to

human life is at stake. Figure II-8, extracted from (Boeh81), shows the numerical values assigned to specific ratings of Very Low (VL), Low (LO), Nominal (NM), High (HI), Very High (VH), and Extra High (XH) for each of the 15 attributes. (Note: all "nominal" attributes would have no effect on the effort required). (Boeh81) provides detailed guidance on these attributes.

### *c. COCOMO 81 Outputs*

The output of the Intermediate COCOMO model is simply the *effort* in man-months for the project being estimated, and a *schedule (time)* in months. The effort output can easily be converted to a monetary value if the cost per MM is known. It is also possible to determine the allocation of the overall effort to various phases of the software life cycle, or time periods, using information presented in (Boeh81).

### *d. Generic Model Behavior*

Figure II-9 and Figure II-10 demonstrates the generic behavior that can be expected with using the Basic COCOMO (*semi-detached*). These figures were established using the same assumptions that provided the inputs to Figure II-3. For the analysis, 10,000 unique simulations were executed using a Monte Carlo simulator. The resulting forecast is presented in Figure II-9 below.

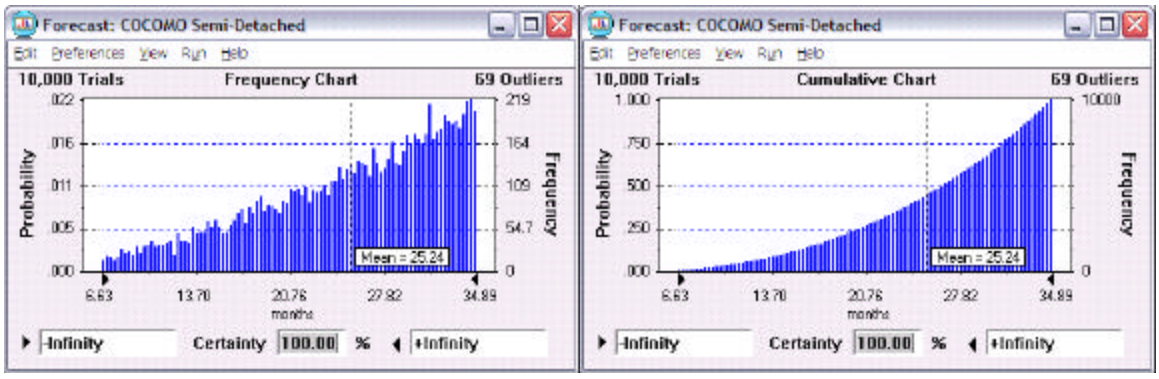



**Figure II-9. Simulated Basic COCOMO.**

The distribution on the left side of Figure II-9 illustrates the frequency distribution of the 10,000 simulations. The right-side illustration is the cumulative distribution.

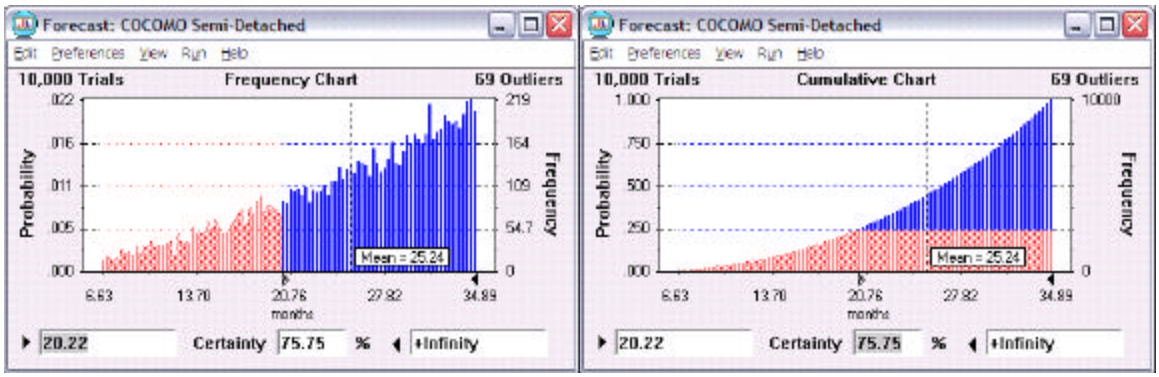



**Figure II-10. Simulated Basic COCOMO at the Mean.**

The Basic COCOMO, as demonstrated with the Simplified Software Equation, projects a smooth projection of the project durations (mean 25 months). Figure II-10 overlays the mean time projected from the SRM (Figure II-3). Approximately 75% of the project durations occur above the average projection of the SRM; again, an indication that the SRM could be projecting project durations too optimistically.

### 5. Price S®

The PRICE S® commercial model was developed by PRICE Systems, LLC to support software cost estimation. PRICE Systems, LLC also developed a hardware model, PRICE H®; hardware operations and support cost estimating model, PRICE HL®; and a microcircuit and electronic module model, PRICE M®. The PRICE S® model is partly proprietary in that all equations are not published, though most are described in the PRICE S® Reference Manual. This model is applicable to all types of software projects, and it considers all software life cycle phases. In addition to the software life cycle phases, it also considers system concept and operational testing phases. Additional information on the PRICE family of tools can be obtained from the model vendor.

### *a. PRICE S® Inputs*

The principal inputs for PRICE S® are grouped into the following nine categories:

- **Project Magnitude.** Reflects the size of the software to be developed or supported. Size can be input as SLOC, function points, or predictive object points.
- **Program Application (APPL).** Provides a measure of the type (or types) of software, described by one of seven categories: Mathematical, String Manipulation, Data Storage and Retrieval, On-Line, Real-Time, Interactive, or Operating System.
- **Productivity Factor (PROFAC).** PROFAC is a calibration parameter that relates the software program to the productivity and efficiency of personnel and management practices.
- **Design Inventory.** Provides for the amount of software inventory available for use (i.e., reuse). Two parameters: New Design (NEWD) and New Code (NEWC) indicate the amount of newness for each software type.
- **Utilization (UTIL).** Reflects the extent of processor loading relative to speed and memory capacity. Values above 50 percent usually increase effort.
- **Customer Specifications and Reliability Requirements.** The platform (PLTFM) parameter provides a measure of the level of testing and documentation that will be needed.
- **Development Environment.** Three complexity parameters (CPLX1, CPLX2, and CPLXM) measure unique project conditions such as multiple site development, requirements volatility, use of tools (e.g., CASE tools), and other factors.
- **Difficulty.** Ratings for internal (INTEGI) and external (INTEGE) integration.
- **Development Process.** Reflects the process being used. Choices include waterfall, spiral, evolutionary, and incremental development.

### *b. PRICE S® Outputs*

PRICE S® computes an *effort* estimate in man-months that may be converted to cost in dollars or other currency units. The effort is allocated among three

stages of software development: Design, Code, and Test. The *effort* is also subdivided into five activities: Systems Engineering; Programming; Configuration and Quality Control; Documentation; and Program Management. PRICE S® also computes a development schedule in months, and provides a schedule effects option that compares an input schedule with that computed by the model. This option also shows penalties for compressing the user's schedule compared to the model's predicted schedule.

PRICE S® provides several optional outputs including resources-complexity and instructions-application sensitivity matrices, plus resource expenditure profiles. It also provides an "at-a-glance" output option to rapidly view the effects of changing selected input parameters. This option is useful for performing "what-if" type trade studies.

## 6. SEER-SEM®

SEER-SEM® is one of a family of tools offered by Galorath Associates. The family also includes hardware cost estimating and hardware-life cycle (SEER-HLC®) models, a software-sizing (SEER-SSM®) model, an integrated-circuit (SEER-IC®) model, and a design-for-manufacturability (SEER-DFM®) tool. SEER-SEM® is partly proprietary in that not all equations are published. However, some relationships are described in the SEER-SEM® User's Manual. SEER-SEM® is applicable to all program types, as well as most phases of the software development life cycle. More information on the SEER family of tools can be obtained from the model vendor.

### *a. SEER-SEM® Inputs*

SEER-SEM® inputs can be divided into three categories: Size, Knowledge-Base Inputs, and Input Parameters. These inputs are described in further detail below.

- **Size.** Size can be input in one of three formats: SLOC, Function Points, or Proxies (proxies allow the user to specify his or her own size measure, which the model later coverts to SLOC). In addition, all software is categorized as "New," "Pre-exists Designed for Reuse," or "Preexists not Designed for Reuse." For pre-existing software, users must specify the amount of software deleted, plus the percentages of redesign, reimplementation, and retest required to modify or reuse the program for the current application. Because the model uses the Program Evaluation and Review Technique (PERT), users must input a "minimum," "most likely," and "maximum" value for all size inputs.
- **Knowledge-Base Inputs**. SEER-SEM® contains knowledge bases for different types of software. Knowledge bases assign default values to the input parameters described below, based on the type of software selected. Users must address these inputs to specify the knowledge base to be used by the model:
- **Platform.** The operating environment of the program (e.g., avionics, ground-based, or manned space).
- **Application.** The overall software function (e.g., command and control, mission planning, or testing).
- **Acquisition Method.** The method in which the software is to be acquired (e.g., development, modification, or re-engineering).
- **Development Method.** The method used for development (e.g., waterfall, evolutionary, or spiral).
- **Development Standard.** The standard used in development and the degree of tailoring (e.g., MIL-STD-498 weapons, ANSI J-STD 016 full, ANSI J-STD 016 nominal, or commercial).
- **Class.** This input is primarily for user-defined knowledge bases.
- **COTS Component Type.** The type of COTS program (if any), such as class library, database, or applications. (COTS relates to activities associated with incorporating commercial software components into development activities).
- **Input Parameters.** SEER-SEM® contains over 30 input parameters, from which users can refine the estimates. Similar to COCOMO 81 and COCOMO II, the input values generally range from "very low" to "extra high." As in size, users must specify a "least," "greatest," and "most likely," value for each input. The selected knowledge base computes default values for most input parameters. Therefore, if users are unfamiliar with a particular parameter, the knowledge-base default values may be utilized. The primary categories of input parameters and a brief description of each follows:

- **Personnel Capability and Experience.** The seven parameters in this category, similar to the "personnel attributes" of COCOMO 81, measure the caliber of personnel used on the project. These inputs are: Analyst Capability, Applications Experience, Programmer Capability, Language Experience, Host-Development System Experience, Target System Experience, and Practices and Methods Experience.
- **Development Support Environment.** The nine parameters in this category are similar to the project attributes and some of the computer attributes in COCOMO 81. They include: usage of modern development practices, usage of automated tools, turnaround time, terminal response time, multiple site development, resource dedication, resource and support location, host system volatility, and target system volatility.
- **Product Development Requirements.** The five parameters in this category include: requirements volatility, rehosting from development to target computer, specification level, test level, and quality assurance level. The last three parameters are identical to the reliability attributes described above for COCOMO 81.
- **Reusability Requirements.** The two parameters in this category measure the degree of reuse needed for future programs and the percentage of software affected by reusability requirements.
- **Development Environment Complexity.** The four parameters in this category measure the language complexity, host development system complexity, application class complexity, and the impact of process improvements.
- **Target Environment.** The seven parameters in this category are similar to some of the computer attributes of COCOMO 81, but focus on the target computer. These include special display requirements, memory constraints, time constraints, real-time code, target-system complexity, target-system volatility, and security (this is the most sensitive input parameter in the model).
- **Other Input Parameters.** There are also special inputs for schedule constraints, labor rates, integration requirements, personnel costs, metrics, and software support.

### *b. SEER-SEM® Outputs*

SEER-SEM® allows the users to select a variety of outputs. The model provides labor estimates in the categories of management, systems engineering, design, code, data, test, configuration management, and quality assurance. A "quick estimate" provides a snapshot of size, effort, schedule, ETR, and other selected outputs anytime

during the estimating process. Optional outputs include a basic estimate, staffing by month, cost by month, cost by activity, man-months by activity, delivered defects, and a SEI maturity rating.

### 7. Keshlaf and Hashim

Keshlaf and Hashim recognize there exists important weaknesses in the existing approaches for software risk estimation. The research is based on the premise that risk documentation and concentrating on top risks are the best ways to save developers time and effort and produce good results in reducing software risks (Kesh00). The authors have designed an improved software risk model and developed an implementation tool; called *SoftRisk*, based off the model. Figure II-11 details the characteristics of the *SoftRisk's* model. Following Figure II-11 is an extract of their research explaining the procedure.

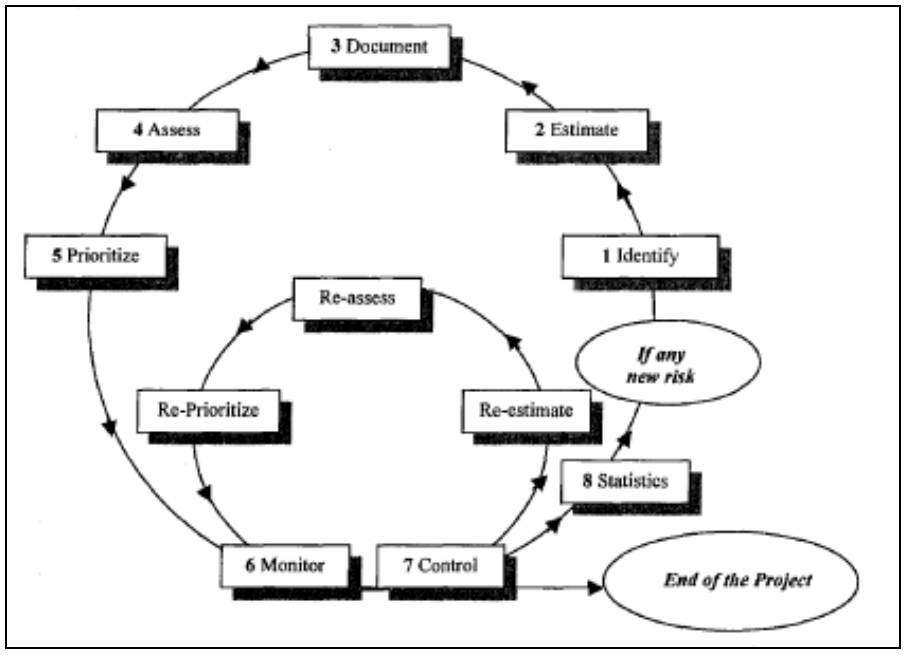



**Figure II-11. SoftRisk Model.**

**Step 1: Risk Identification**. All potential risks must be identified. Two forms are suggested to be used, one for general risks data which can be used by any software development project, whereas, the second is for specific projects risks data which affect the desired project.

**Step 2: Risk Probability and Magnitude Estimation.** Each identified risk (Specific risk) is estimated in terms of its probability and magnitude. A special checklist can be used to assist in estimating both probabilities and magnitudes of risks. The estimation should be under one of the following categories:

1 – Negligible2 – Very Low3 - Medium

4 – High5 – Very High6 – Extra High

**Step 3: Risk Documentation.** All generic and specific risks data must be documented in order to use them for tracking project's situation, statistical operations, and future risk predictions.

**Step 4: Risk Assessment.** Each risk must be assessed based on its probability and magnitude. It is advisable to use risk exposure (RE) formula (Boeh83):

**RE = Risk Probability * Risk Magnitude**

**Step 5: Prioritization and Highlighting Highest Ten Risks.** All risks are sorted based on RE values and the top ten risks for each inspection are prioritized and listed.

**Step 6: Monitoring (Graphic Representation).** A line graph is recommended to be used to represent RE values. The graph is divided into three zones: red for catastrophic, yellow for medium, and green for negligible risks.

**Step 7: Controlling.** Based on the severity of the risk, a suitable reduction technique is performed. It could be mitigation, contingency, or crisis plan. Re-estimation, reassessment, and action documentation must be carried out after performing any one of the reduction techniques.

**Step 8: Performing Statistical Operation and Going Back to Step 1.** Any suitable statistical operation can be carried out if need be. However, if the project is still ongoing it is highly recommended to continue performing risk management steps to avoid any problems.

SoftRisk requires the user to make qualitative estimates about the associated risk probability and magnitude estimation. The tool takes the qualitative inputs and converts this information into quantitative data for internal calculations. Approaching risk management in this fashion does not facilitate removing subjective inputs from the model. The basic problem of deriving quantitative metrics early in the project still exists.

The SoftRisk model does recognize the importance of documenting software development risks. It even introduces risk reduction advice and documents reduction actions. However, all of the model's results are derived from a subjective basis.

Finally, SoftRisk has not been thoroughly tested to establish its validity. The preliminary tests were conducted to establish accurate functionality of the tool itself, not testing the results of the tool. Additionally, the authors recognize the fact that the SoftRisk tool only prepares the risks data to become ready for future statistical operations.

### 8. Mitre Corporation

Software development activities can gain valuable information and insights from examining what other development activities have done to mitigate risk when developing software projects. The Mitre Corporation has developed a commercial tool called the *Risk Assessment and Management Program* (RAMP), which is a risk management information system that provides interactive support for identifying, analyzing, and sharing risk mitigation experience.

Operating on an Internet platform, RAMP lets authorized users access project-specific risk mitigation experiences across the corporation from locations anywhere in the world (Garv97). RAMP contains a database of systems engineering-related project risks and mitigation strategies, along with more than 1,000 links to risk-relevant resources and contacts around the world.

According to (Garv97), RAMP provides users the ability to:

- Identify and collaborate with domain experts in specific risk and technology areas
- Query experts via e-mail
- Examine RAMP's risk information templates that document lesson-learned summaries and mitigation strategies on similar projects
- Create custom portfolios of similar projects, the risks, and mitigation strategies

One major drawback to RAMP is this tool does not directly help software developers access project risks in a quantifiable manner. Benefits exist in being able to consult with other practitioners and become aware of overlooked project risks on the software development project. Additionally, by examining patterns among the software risk, insights can be derived to help verify which software metrics influence software estimation.

## E. SOFTWARE METRICS

### 1. Software Metrics Roadmap

This body of research addresses issues associated with how the field of software engineering currently collects metrics for software projects. The research identifies the shortcomings in traditional metrics approaches, which are often driven by regression-based models for cost estimation and defects prediction. (Fent00) feels that the traditional approaches provide little support for managers wishing to use measurement to analyze and minimize risk.

The fundamental problems with regression models often lean to a misunderstanding about cause and effect (Fent00). The research documents a short example of using regression analysis in predicting road fatalities. Using regression analysis, a model will indicate that the number of fatalities on the highway is less during the months of January and February. Interpreted in naïve manner, this could lend managers into falsely believing that it is safer to drive on the highways during these

months. However other factors identified only by causal relationships will indicate there are several other factors that cause fewer fatalities during the mentioned months. The real reason for fewer fatalities is the weather is more likely to be bad during these months and bad weather can cause treacherous road conditions. In these circumstances, fewer people drive cars or drive more slowly.

The author's notion of a causal model is telling the story that is missing from the naïve approach – software practitioners can use it to help make intelligent decision for risk reduction, and identify factors that can be controlled or influenced. Modeling can provide an explanatory structure to explain events that can then be quantified.

Much of software metrics has been driven by the need for resource prediction models (Putn92), (Boeh81). Usually this work has involved regression-based models in the form of *effort* = f(size). Ignoring the fact that solution size cannot possibly cause effort to be expended, such models cannot be used effectively to provide decision support for risk assessment.

(Fent00) offers up an extended definition of software metrics and divide the subject into two components:

- The component concerned with defining the actual measures (in other words the numerical metrics)
- The component concerned with how to collect, manage, and use the measures

The rationale for almost all individual metrics has been motivated by one of two activities (Fent00):

- The desire to assess or predict effort/cost of development processes
- The desire to assess or predict quality of software products

Causal models are introduced. The great challenge for the software metrics community is to produce models of the software development and testing process which take account of the crucial concepts missing from classical regression-based approaches (Fent00). Specifically, models are needed that can handle:

- Diverse process and product variables
- Empirical evidence and expert judgment
- Genuine cause and effect relationships
- Uncertainty
- Incomplete information

This dissertation utilizes causal analysis, simulation, real project data, and statistical-based approach to model the behavior of software development. The enhancements offered by the Modified Risk Model, address the issue (Fent00) surfaces about effort = f(size). Additionally, the risk assessment models use a combination of multiple parametric inputs to help avoid the pitfalls of falsely misunderstanding the cause and effect relationship.

### 2. Moyniham

Tony Moynihan conducted a survey of a homogenous group of project managers that revealed a surprising diversity of risk management concerns. He approached the research by surveying 14 experienced application systems developers, all located in Ireland. In the interviews with each of the developers, he used a technique of personal construct elicitation. It works as follows: a personal construct is a bipolar distinction, or scales, which a person uses when contrasting different people, objects, situations, and so on. For example, say a personal distinction between dogs is the likelihood of whether or not a dog will bite you. So, when comparing dogs, or when confronted by a particular dog, think in terms of "Will it or won't it bite me?" (Moyn97)

To survey the program managers' constructs, Monahan asked each of program managers to list the development projects they had managed over the last couple of years. Then three projects from the list were selected randomly and the question was asked: In what important ways are any two of these three projects the same, but different from the third, in terms of important situational factors considered when planning the project? This process was repeated with different triads of projects until all of the projects were depleted and no new constructs were introduced.

The personal constructs identified by the 14 developers were then assigned to various themes and compared to Barki's (Bark93) risk variables and the Software Engineering Institute (SEI) Taxonomy-Based Risk Identification Instrument (Carr93). The research continues to detail the differences, similarities, omissions, and one-to-one correlations between the constructs, Barki's, and SEI.

As one might expect, the software developer's personal constructs reflect most of the risk factors identified in the software project management literature. But their constructs also contain several rich elaborations to some of these factors. For example, the concept of requirements uncertainty. This is addressed by seven different personal constructs, each of which captures a subtly different but important facet of the concept. Given this level of elaboration on a single concept, the concern is will it ever be possible to capture all the subtleties of projects on any reasonably sized checklist (Moyn97).

There exist other constructs that receive little attention in the mainstream literature. For example, take aspects of where the project's control resides and how it is exercised, and aspects of the interface between developer and client organizations. These omissions may be a direct consequence of differences between the contexts in which the different studies upon which the literature is based were carried out. The notion of building single, all-encompassing risk taxonomy for use by all software developers is probably unrealistic. Researchers need different risk taxonomies for different project contexts (Moyn97).

(Moyn97) feels that if a larger body of project managers (beyond the original 14) would have been interviewed, it is unlikely that the broad findings would have been

different. Even within the 14 interviews, the variation across program managers in the foci of their constructs provokes questions that can be only answered by further study:

- How, if at all, does a program managers' professional training or experience influence the way in which he or she construes projects?
- Do different program managers in the same organization tend, over time, to see things in much the same way and thus converge on the same constructs?
- Is it possible to build a useful taxonomy of program managers based on the foci of their constructs, one that yields categories such as the politician, the technician, and so on?

Better understanding the conceptual lenses with which project managers approach software projects – and the biases that tint those lenses – may help researchers isolate and avoid behavior that reduces management effectiveness while promoting behavior that increases it (Moyn97).

This research provides some interesting insights to developing or verifying a set of metrics in which to develop a risk assessment model. Additionally, the research suggests which measurements, currently found in existing literature, produce negligible effects on software risk.

It remains important to scope software risk models and exercise them according to their original designs. (Mony97) illustrates that software risk assessment continues to be an unformalized problem that relies on checklist and taxonomies. For a software risk model to maintain relevance, the model must be generic enough to provide rough-order-of-magnitude estimates, yet structured enough to eliminate ambiguities and bias amount analyst. The Modified Risk Model accomplishes this requirement.

### 3. Function Point

The objective of estimating software attributes is to provide a means of quantitatively describing a software product while it is still an abstract concept. This is very similar to the aircraft industry using the estimated weight of a new aircraft design to develop project plans and estimate cost or determining the development cost estimate of an electronic box by determining an estimation of the number of circuit boards and connectors required to implement the specification. *Function Point* measurements, more recently renamed Functional Size measurement (SEI97), provide a technique that allows software engineers to size computer resources and develop cost and schedule estimates.

The idea of "quantitatively" describing a software product can occur because the idea of measuring the functionality is normalized. *Functionality* cannot be measured directly, so techniques have been developed to indirectly derive the functionality. The traditional *functional complexity* metric has been introduced by (Albr79) and (Albr83). *Function Point Analysis* has become generally accepted as an effective way to (SEI97):

- Estimate a software project's size (and in part, duration)
- Establish productivity rates in function points per hour
- Evaluate support requirements
- Estimate system change costs
- Normalize the comparison of software modules

*Function points* are calculated by completing Table II-1. Five information domain characteristics are determined and counts are provided in the appropriate table location. Each of the informational domains is detailed below the table (Pres01).

| | Simple | Weight | Medium | Weight | Complex | Weight | Total |
|---|---|---|---|---|---|---|---|
| Inputs | ( | * 3) + | ( | * 4) + | ( | * 6) = | |
| Outputs | ( | * 4) + | ( | * 5) + | ( | * 7) = | |
| Queries | ( | * 3) + | ( | * 4) + | ( | * 6) = | |
| Files | ( | * 7) + | ( | * 10) + | ( | * 15) = | |
| Interfaces | ( | * 5) + | ( | * 7) + | ( | * 10) = | |
| | | | | | NAFP | = | Σ |

**Table II-1. Function Point Calculation (Zuse97).**

- **Number of user inputs.** Each user input that provides distinct application-oriented data to the software is counted. Inputs should be distinguished from inquiries, which are counted separately.
- **Number of user outputs.** Each user output that provides application-oriented information to the user is counted. In this context output refers to reports, screens, error messages, etc. Individual data items within a report are not counted separately.
- **Number of user inquiries.** An inquiry is defined as an on-line input that results in the generation of some immediate software response in the form of an on-line output. Each distinct inquiry is counted.
- **Number of files.** Each logical master file (i.e., a logical grouping of data that may be one part of a large database or a separate file) is counted.
- **Number of external interfaces.** All machine-readable interfaces (i.e., data files on storage media) that are used to transmit information to another system are counted.

The result of the total is called Function Points not adjusted. Fourteen adjustment factors, whose values are in the range of zero to five, describing the environment are added. Finally the *function points* are calculated by the formula (Pres01):

$$FP = NAFP + [0.65 + 0.01 * \sum(F_i)]$$

where,

NAFP is the non-adjusted Function points
$F_i$ is each of the fourteen adjustment factors

Once a user has the adjusted *function point* value, an estimate of the SLOC can be estimated. (Putn92) uses a technique called "backfiring" to baseline *function point* values

to SLOC. Used in this way, this technique is similar to (Jone94) procedure to convert function point into a universal sizing measure. In (Jone94), the translation is dependent on the development language.

The software risk models considered in this dissertation all implement a sizing measure in some form. For instance, the SRM and MRM both depend on obtaining their sizing metric from the LGC of PSDL code. As demonstrated throughout this dissertation, LGC is a logarithmic conversion to SLOC. Additionally, Basic COCOMO and the Simplified Software Equation both utilize SLOC as their sizing measure. *Function point* analysis can provide the sizing measure for any of the estimation models and subsequently is considered a complement to this research.

## F. SIMULATION TECHNIQUES

### 1. VitéProject 2.0

VitéProject is a commercial modeling and simulation tool based on the Virtual Design Team (VDT). Contingency theory is the foundation of the VDT directed by Dr. Raymond Levitt at Stanford (Jin96). The underlying model of VitéProject, based on VDT-2, has been validated on three levels at the Center for Integrated Facility Engineering at Stanford University (CIFE):

- Micro-level analysis, using toy problems
- Meso-level analysis, using toy problems and experiments
- Macro-level analysis, by testing for authenticity, reproducibility, generalizability, and prospective

A full accounting of VitéProject validation strategy can be found in the dissertations of (Nogu00) and (Thom99). The SRM development uses the VitéProject 2.0 in an attempt to apply CPM/Pert modeling techniques to software engineering. (Nogu00) parameterized VitéProject to simulate 16 different software project

developments. The previous research implemented 30 simulations for each scenario development.

VitéProject 2.0 has several limitations hampering its potential. First, VitéProject limits a maximum of 100 simulated runs for a given scenario. Second, the resulting data is presented in summary format. The summary format provides very limited ability to conduct histograms or a sensitivity analysis.

To change software project scenarios, users must make changes to each individual activity/actors in a given scenario, before a new scenario could be simulated. This process, even for the smallest organizational structures, becomes time consuming and inconvenient.

(Alex01), in a communication with Vité Incorporated, discovered that Vité no longer supports VitéProject 2.0. The new product is SimVision 3.0. Although the analytical engine within VitéProject is the same as SimVision, SimVision provides a much needed user interface enhancement. However, this research benefits from the decomposed architecture of the original VitéProject 2.0.

### 2. Visio 5.0

VitéProject modeling and simulation software requires a graphical input of the organization's structure. Visio 5.0 provides the graphical interface for VitéProject 2.0. Visio opens all VitéProject documents via the VitéProject stencil, which is akin to a template in Microsoft® Word. The initial organizational structure for a given scenario, as well as all changes to a scenario or an organization's parameters occur within Visio.

Users can initiate a simulation from Visio, via a VitéProject add-in, or from VitéProject itself. If the scenario is executed from Visio, all data in the scenario drawing is first written to the VitéProject database file prior to the simulation run. If the simulation is run from Vité, the simulation relies solely on the data currently saved in the VitéProject database file.

Figure II-12 presents the simulated organization and the simulated software process used in the development of the SRM. The process presents only four cycles of evolution. Each cycle contains the activities represented in Evolutionary Software Development (Nogo00).

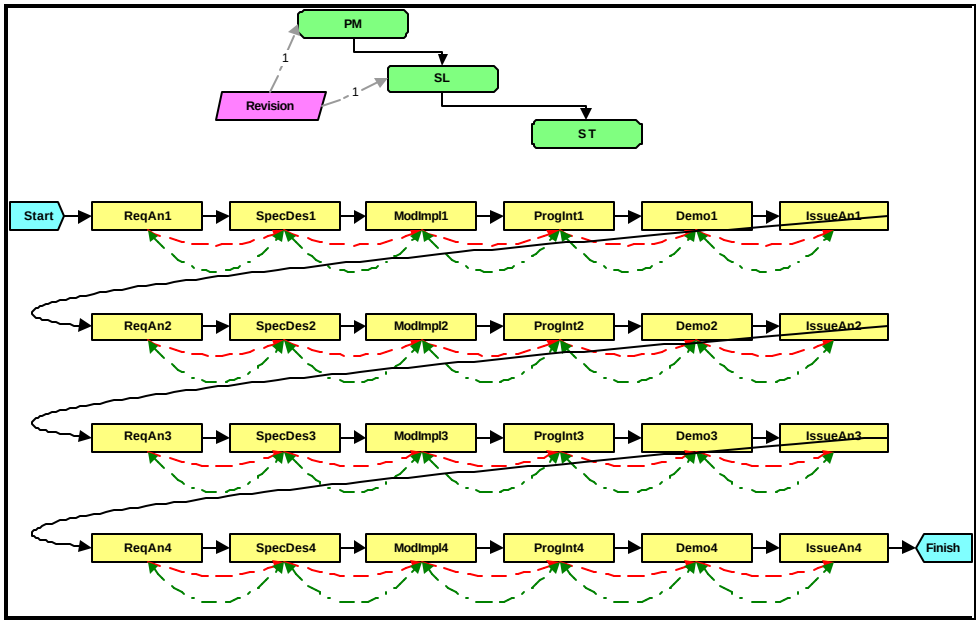



**Figure II-12. Organization Representation for VitéProject[2].**

### 3. Monte Carlo

The word simulation refers to any analytical method meant to imitate a real-life system, especially when other analyses are too mathematically complex or too difficult to reproduce. Without the aid of simulation, a spreadsheet model will only reveal a single outcome, generally the most likely or average scenario. Spreadsheet risk analysis uses both a spreadsheet model and simulation to automatically analyze the effect of varying inputs on outputs of the modeled system.

---

[2] The detailed description of the notation can be found on the VitéProject user manual (Levi99). Rectangles indicate tasks. Rounded-corner rectangles indicate roles. Parallelograms indicate meetings. Double-headed-dashed arrows indicate information dependencies between tasks. Dashed arrows indicate problem dependencies between tasks. Normal arrows indicate precedence dependencies between tasks.

One type of spreadsheet simulation is Monte Carlo simulation, which randomly generates values for uncertain variables over and over to simulate a model. Monte Carlo simulation was named for Monte Carlo, Monaco, where the primary attractions are casinos containing games of chance. Games of chance such as roulette wheels, dice, and slot machines, exhibit random behavior.

The random behavior in games of chance is similar to how Monte Carlo simulation selects variable values at random to simulate a model. When a die is rolled, the outcome of either a 1, 2, 3, 4, 5, or 6 will surface, however, it is not known which for any particular roll. It is the same with the variables that have a known range of values but an uncertain value for any particular time or event (e.g. interest rates, staffing needs, stock prices, inventory, and phone calls per minute).

For each uncertain variable (one that has a range of possible values), the possible values are defined with a probability distribution. The selected distribution is based on the conditions surrounding that variable. The most common distribution types include: *normal, triangular, lognormal, and uniform*; however several others exist. To add this sort of function to an MS Excel spreadsheet, the equation that represents this distribution must be known. A simulation calculates multiple scenarios of a model by repeatedly sampling values from the probability distributions for the uncertain variables and using those values for the cell. [Crys93]

THIS PAGE INTENTIONALLY LEFT BLANK

# III. RESEARCH FRAMEWORK

This research delivers a validated software risk model that is developed to aid program managers in effectively planning the required *effort* to deliver software products. A fundamental roadblock in this type of research is finding accessible data against which to validate the findings.

One of the steps in the research is validating a specific risk model, the SRM, against data that was not available when the SRM was developed. The insight gained from performing this validation allowed significant extensions and modifications to the SRM; in effect, creating a new model with more robustness, wider applicability, and better predictive characteristics. The data used for this validation is from the proprietary database of Quantitative Software Management (QSM®).

## A. BACKGROUND

Simulation is simply the use of a computer model to "mimic" the behavior of a complicated system and thereby gain insight into the performance of that system under a variety of circumstances. A model is a representation and an abstraction of anything such as a system, concept, problem, or phenomena (Balc94). There exists substantial literature on simulations and models. Of more importance, is a structured methodology to aid in the development of simulations or models.

(Balc98) provides a ten-step life cycle model useful to guide the development of simulation models, Figure III-1. The process of developing simulations strongly correlates with developing software risk models. This research utilizes Balci's life cycle model to help assess the current state of software risk models and continues using the life cycle through the extension of software risk models.

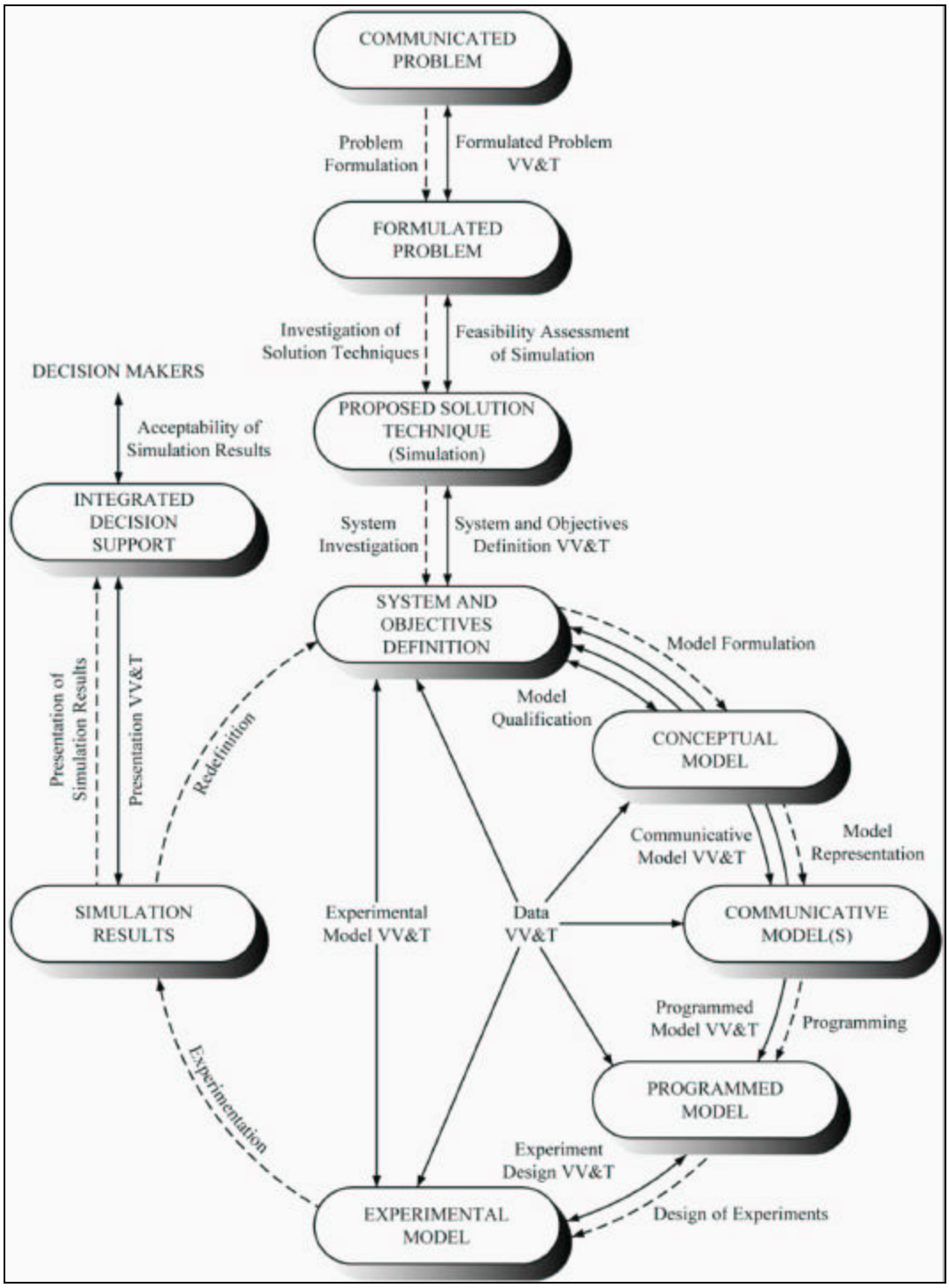



**Figure III-1. The Life Cycle of a Simulation Study, from "Balc94".**

The previous development of software risk models can be considered immature in terms of completing Balci's ten-step process. (Nogu00) completes the first half of the development process: *problem formulation, investigation of solution techniques, system investigation, model formulation,* and *model representation*. Although the life cycle is not considered a linear process, the SRM currently resides as a conceptual model.

- **Problem Formulation.** Stakeholders, through personal experience, research groups, consultants, etc, identify that a problem exists. Frequently, the problem must be re-formulated so that it can be communicated sufficiently well to facilitate further action. (Nogu00) identifies that a need exists in software risk identification. The identified need is the desire to project software development risks early in the software development process; through quantifiable parametrics capable of being collected unobtrusively.
- **Investigation of Solution Techniques.** Frequently, a problem can be solved by several different courses of action. Trade-Offs must be examined to determine which solutions best meet the users needs; both technically and economically. With modeling, the best choice for solving the problem may not be necessarily by conducting simulation. (Nogu00) documents the use of the VitéProject simulation to facilitate the development of the SRM. The simulation is used to replicate the time required to produce 16 software development scenarios. Each of the 16 scenarios was simulation 30 times.
- **System Investigation.** System investigation is about identifying the influencing factors of the problem's domain. (Shan75) identifies six major system characteristics that should be examined with respect to the study objectives identified in the formulation of the problem: *change, environment, counterintuitive behavior, drift to low performance, interdependency,* and *organization.* (Nogu00) concentrates on the *environment*, *interdependency* and *organization*. The parametric inputs are derived from study of the organizational interdependencies between project, process, and product risk (the environment).
- **Model Formulation.** The person modeling the formulated problem must clearly understand the issues and envision the model's representation to meet its needs. Model formulation is a partial requirement for model design that constitutes model formulation and model representation. (Nogu00) conducted background research to survey techniques implemented by previous research. A method was needed to accomplish two things, (1) how to represent software development in a simulation, and (2) how to interpret the simulation results.

- **Model Representation.** A communicative model is a model that can be translated to other people. It is a representation without bias or ambiguity. This is the process of capturing a thought in a modelers mind and adding structure so that it can be communicated. (Nogu00) established an organizational representation in the VitéProject simulation to mimic software development. Organizational Consultant was utilized to derive the parametric descriptions to generate the simulation.

Validation is required to mature the existing software risk models. Validation, verification, and testing (VV&T) occur throughout the entire model development process. Only through substantial VV&T can potential flaws be identified and confidence obtained in the model's performance. Validating a model exclusively with simulation, fails to prevent potential *Type II* errors.

The intent of this dissertation is to expand the field of software risk models by subjecting the foundational study of (Nogu00) to the later half of the simulation life cycle: *programming, design of experiments, experimentation, redefinition, and presentation of the simulation results.* This research utilizes both simulated and real project data to conduct the VV&T. Chapters IV and VII detail the validation criteria.

- **Programming.** Before experimentation can occur with the model, the model must be translated into an executable simulation. In terms of life cycle steps, programming is the process of transforming a communicative model into a programmed model. This research implements programming in two ways: (1) the SRM is programmed as a self-driven simulation, and (2) the SRM is programmed as a trace-driven simulation. Both self and trace driven simulation provide independent and valuable insight to the behavior of the model.
- **Design of Experiments.** Design of experiment constitutes the plan to test the executable model. Real world projects are utilized to exercise the model's performance. The project data is utilized in three ways: direct comparison, self-driven simulation stimulus, and trace-driven simulation stimulus. The accuracy of the model is determined by five evaluating criteria: *actual error, absolute error, balance, under estimation error*, and *over estimation error*. The models are compared to the actual project data as well as to each other.
- **Experimentation.** Experimentation is the act of implementing the executable version of the experiment design to produce simulation results.

(Shan75) explains there are six primary purposes of simulation; four of which apply to this research: evaluation of system behavior, sensitivity analysis, forecasting, and optimization.

- **Redefinition.** If a model fails to perform as intended (Type II error) then it becomes necessary to redefine the model. (Balc94) defines five situations when a model may need refinement: (1) updating the experimental model to represent a current form of the system, (2) altering the model to obtaining another set of results, (3) changing the model for maintenance, (4) modifying the model for other use, or (5) redefining a new system to model for studying an alternative solution to the problem. This dissertation redefines the work of (Nogu00) because of four of the five points. This research does not modify or enhance the SRM for maintenance (point three).
- **Presentation of the Simulation Results.** The benefit of simulation is determined after analysis is conducted on the documented simulation results. These results must be interpreted and presented to the decision makers (ultimately the stakeholder who initiated the problem research). This research documents the behavior of four different software risk models. Comparisons are made between each of these models and actual project data.

## B. RESEARCH DESIGN

Four models are available, as documented in (Nogu00), to project the probability (i.e. the risks) of not applying an adequate *schedule* to a software development project. Each model is designed with an increasing degree of accuracy, based on:

- Metrics from the three risk factors
- Weibull cumulative density function
- The derivation of the time

However, little confidence is warranted in the usefulness of these models[3] because of the negligible number of projects exercised against the models (Chapter IV demonstrates the specific characteristics from the previous validation attempts). For all practical purposes, little validation has been conducted against these models outside of

[3] Model 4 (SRM), the most accuract model from (Nogu00), is the only model validated in this dissertation or any other study.

the simulation utilized for the SRM's development. This fact leads to the basic research question addressed by this dissertation:

- **How do the current risk estimation methods perform when exercised on real projects?**

To determine the model's performance, the research examined the characteristics of the three projects that have previously been exercised and attempted to identify comparable projects within the QSM® database. This was a logical starting point and proved necessary to maximize the accuracy in the models projections. The available research of (Nogu00), (John01) provided a baseline of the types of projects to consider. The initial validation of current risk estimation methods is against a set of 112 software projects.

To validate the performance of software risk models, techniques have to be developed to facilitate a mapping between real world project attributes currently available and the parametric inputs required in the SRM. The SRM was developed to support the Computer Aided Prototyping System (CAPS). However, the unique model inputs, available from the CAPS environment, are not readily available from real world projects; where a large number of projects exist. This leads to a logical second question:

- **How do the metrics required for the current risk estimation methods correlate to metrics collected in real world projects?**

Correlations were determined between the available software project data and the parametrics of the SRM. The SRM utilizes three primary input metrics: *efficiency*, *requirements volatility,* and *complexity;* each with a unique collection process. Metrics, in this form, are not readily available in the QSM® database. A complete listing of the "default" metrics available in the QSM® database is included with the *data dictionary* in Appendix A.

A close examination of the QSM® database identified suitable metrics for mapping: Productivity Index, Requirements Change, Duration, Project Staffing, and Effective Source Lines of Code. However, the most appropriate way to map the metrics was not immediately clear. The research considered 27 combinations of SRM's input metrics. Chapter IV provides the details of the mapping and demonstrates the results of the validation.

An issue in planning the research was determining when to proceed with exercising the models. In actuality, mapping the metrics and exercising the models occurred in iteration. An experiment was set up according to an input mapping and all of the project data is entered into the SRM. After documenting the results, the experiment proceeds to the next mapping configuration.

Studying the performance of the SRM in isolation does not provide as useful insight as comparing the model against other parametric models. Four basic artifacts were utilized in the research, all of which can provide software estimations. The first artifact is the SRM. The second artifact is the collection of real world projects captured in the QSM® database. The third artifact is the Simplified Software Equation and the fourth artifact is the Constructive Cost Model. Using these four artifacts, redundancy is instilled in the experimentation process. Success of the SRM is not only determined by the actual accuracy with the project database, but against the accuracy of what this dissertation is calling the "industry standards"; the Simplified Software Equation and Basic COCOMO models.

Preliminary research (John01) (Alex01) (Murr02) indicate the Software Risk Model does not project software risks within acceptable tolerances and therefore questions arise as to the true benefit of its use. Subsequently, a third question follows the logical progression of the first two:

- **What are the necessary enhancements evolving the current risk estimation methods to provide improved results when exercised on real projects?**

Completing the validation of the SRM revealed several issues that severely limit the model's usefulness; it became necessary to revisit the core assumptions that serve as the SRM foundation. One major assumption challenged is the accuracy of using VitéProject to provide a representation of the software development process. Prior to this research, no baseline existed to help analysts tune the VitéProject simulation for software development. It is extremely challenging to develop a simulation-based model if a baseline is not available for calibration. The absence of a calibrated VitéProject surrounds the development of the SRM.

This research facilitated the completion of an Application Program Interface (API) (Chapter V). The API allows users to automatically record the simulation results from VitéProject, previously a time consuming task. Additionally, a large number of simulations can occur through batch jobs, greatly increasing the efficiency of the analysis. This research implements in excess of one million simulations to establish a calibrated simulation. The Simplified Software Equation and the Constructive Cost Model provide a baseline calibration for the simulation (Chapter V and Appendix D).

Observing an increase in the number of simulated data points revealed that accounting for the complexity of a software project using the SRM's *Large Granular Complexity* measure alone is not sufficient. Insights were explored that led to the development of two independent measures to replace the original concept of the LGC. This issue is addressed in Chapter VI.

Exponentially increasing the simulation executions also allowed interpretation of the simulation from different perspectives. During the mapping process explained in Chapter IV, difficulties surfaced in the justification of the staffing size implemented in the SRM. The SRM projects the probability as a dependent variable for an independent number of days. However, (Nogu00) does not adequately address the staffing size to achieve this projection. These limitations, along with evidence generated from the

calibration, required the VitéProject calibration to represent the required *effort* instead of the previous implementation, *duration.*

Calibrating the VitéProject simulator provided valuable insight as to the suitability of VitéProject as a software simulation tool. The simulation data contained discontinuities that could not readily be explained (Chapter V). Subsequently, the research discontinued use of VitéProject and required an alternative approach to enhancing the SRM. Analysis of an increased number of real-world software applications provided a method to replace the need for simulation. Details of the simulation discontinuities are in Chapter V.

The insight gained from performing this validation allowed significant extensions and modifications to the SRM; in effect, creating a new model with more robustness, wider applicability, and better predictive characteristics. The new development, the Modified Risk Model (MRM), combines four primary parametric inputs to project the required *effort* in a software development. The MRM implements two key input parameters, *Efficiency (EF)* and *Requirements Volatility (RV),* exactly as the original implementation in the SRM. The MRM extends the *Complexity (CX)* measure to more accurately portray the *complexity* and *volume* of software. As in the SRM, analysis of the PSDL source code derives the extended complexity measures.

Implementing the MRM requires the analyst to establish the staffing size and determine the number of available months. The MRM treats this input as a level staffing profile (i.e. the same number of personnel applied to the project each month). The available effort is a function of the available months and staff. The MRM translates the available effort along with the other four parameters (*efficiency, requirements volatility, functional complexity,* and *functional size*) through the three-variable Weibull distribution to project the probability of completing a software development project (Chapter VI).

To establish confidence in the accuracy of the MRM, the MRM is validated against the same criteria as the SRM validation; except against a larger project base. Validating a model using real project data is not precise due to the large deviation in software development efforts. However, the efforts expand the validation to include the

performance of the "baseline" models[4] (Chapter VII). Examining the MRM with this level of scrutiny, not only provides confidence in the model against actual projects, comparisons are provided with the industry's best practices.

[4] Simplified Software Equation and Basic COCOMO.

# IV. COMPARISON STUDY ON CURRENT RISK MODELS

Although research on the SRM shows promise in estimating the associated risk when developing software systems, theoretical simulation is the primary means of its validation. Simulations drove the development of the SRM and simulations provided the validation. The SRM has projected the performance of three "real world" projects to date (Nogu00, John01). All three of these projects were projected post-mortem. Conclusive validation in the context targeted by its original design, that is estimating risk early in a software project's life cycle, does not exist.

Validating the SRM presents unique challenges due to input metrics required by the model. This problem is a double-edged sword. A major attraction to using the SRM is these metrics, which are determined in a definitive and quantifiable manner and derived extremely early in the software development process. However, outside of the academic environment, it is not common practice to collect these unique metrics in the required form to utilize SRM.

In order to help establish confidence in the usefulness and accuracy of the SRM, it is necessary to project its performance on numerous real world projects. To date the model has not been implemented (early in the software development cycle) according to its original design. However, the next logical step is to continue to exercise the model on a post-mortem basis. To accomplish this, it is necessary to derive a mapping between the SRM metrics and metrics collected on actual historical projects. The development of such a mapping would extend the application of the SRM to an expanded project base. This chapter covers the details of validating the SRM as laid out in phases one, two, and three of the research design in Chapter 3.

## A. PROJECT IDENTIFICATION

This phase involves determining the characteristics of the three projects currently documented in the previous risk assessment work, and identifying comparable projects

within the QSM® database. This is a logical starting point from which to construct a useful mapping of the metrics used in SRM to metrics used in real world applications.

### 1. Project Criteria for SRM Application

Several criteria have been set forth for identifying projects which can be evaluated effectively by the SRM.

- The projects should be recent DoD development projects utilizing the Software Engineering Institute, Capability Maturity Model, (CMM) level 2 processes or better which have the necessary metrics and data available
- The projects should have worked through the lifecycle phases of an evolutionary development, in this case an evolutionary spiral model
- The projects should have used Object-Oriented Methodology (OOM)
- The projects should contain multiple Computer Software Configuration Items (CSCIs)
- The project should be coded in Ada[5]
- The software should be real-time embedded
- The project should use a Computer Aided Software Engineering, (CASE) tool (John01)

During the development of the SRM, the model was applied to a large project developed by the Uruguayan Navy (the project is a simulator developed for war gaming (SIMTAS) consisting of 75,240 lines of code in Pascal); the model predicted 17 months instead of 18 months (Nogu00).

Using this project as a baseline, the model was then applied to two software projects with development characteristics identified in Table IV-1.

[5] The SRM utilized Ada to derive its complexity algorithm.

| Criteria | Project A | Project B |
|---|---|---|
| SEI CMM Level | Level 3 | Level 2 then 3 |
| Life Cycle Model | Evolutionary Spiral | Incremental |
| Methodology | Object-Oriented | Functional Decomposition |
| # Computer Software Configuration Items | Five | Six |
| Primary Language | Ada | Ada, Assembler |
| Application Type | Real time embedded | Real time embedded |
| CASE Tool Suite | Rational Rose | Requirements Traceability Matrix |

**Table IV-1. Real World Validation Projects**

### 2. Additional Projects for SRM Validation

In order to perform a more meaningful validation of SRM, the QSM® database was consulted for projects meeting the criteria identified above. The constraints were relaxed to accept projects that were Ada based and predominately real time or engineering applications at a minimum, but may not have satisfied the other criteria.

The result of the initial request was a subset of 112 software projects[6]. Figure IV-1 categorizes the projects according to the application type. The majority of the projects are from the Avionics industry while the least number of projects represent Microcode and Real-time systems.

[6] This dissertation refers to the "original 112" software projects, however, during validation one project was discarded due to inconsistent information. The actual validation occurs against 111 software projects.

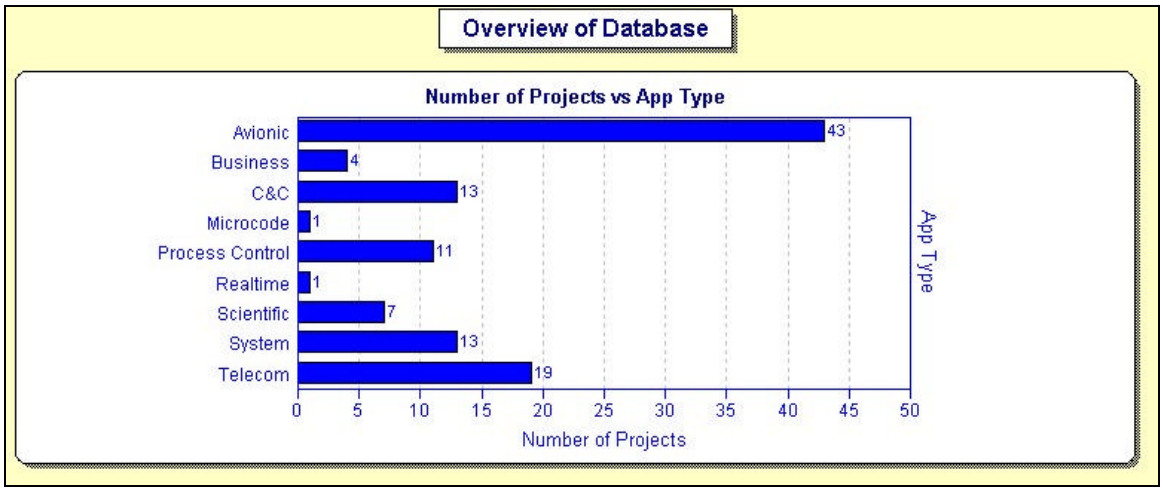



**Figure IV-1. Projects vs. Application Type.**

In the subset of initial projects available for evaluation, there is a total of 26 different organizations. Figure IV-2 displays the number of projects captured in the database by each organization. In an effort to preserve proprietary information, a generic identifier has replaced organization names.

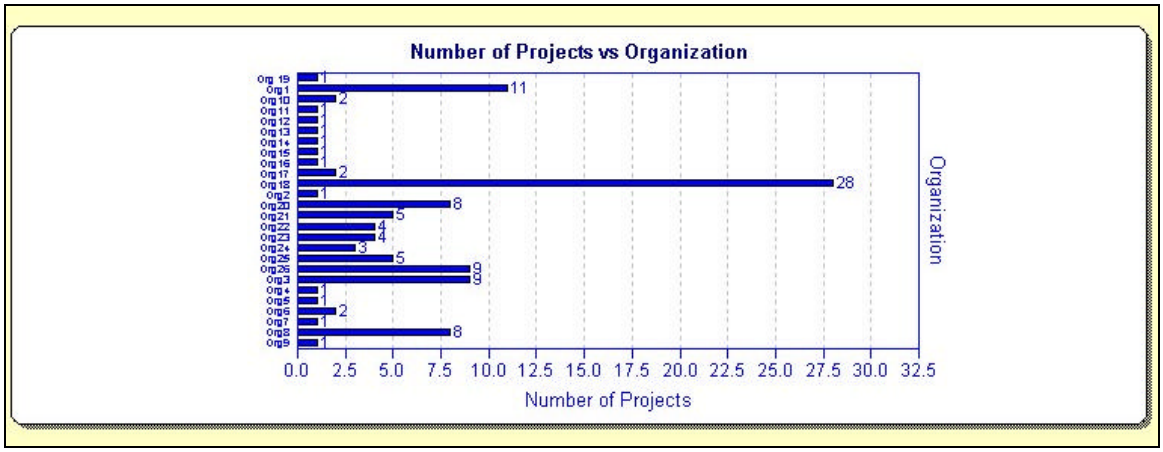



**Figure IV-2. Projects vs. Organization.**

To further clarify the effort expended to produce the software projects, Figure IV-3 considers the actual expended calendar time to complete each software development phase.

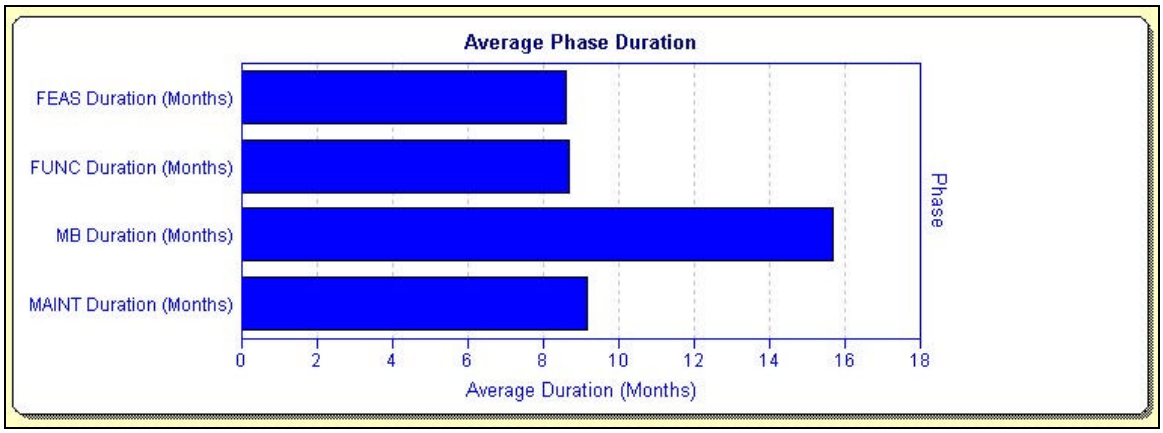



**Figure IV-3. Average Phase Duration.**

Finally, Figure IV-4 shows schedule slippages. For example, during the main build, 33% of all of the projects exceeded their initial estimation by as much as 25%. Probably more alarming is the fact that out of the 112 projects under consideration, only 13% finished on time or earlier than their schedule estimates.

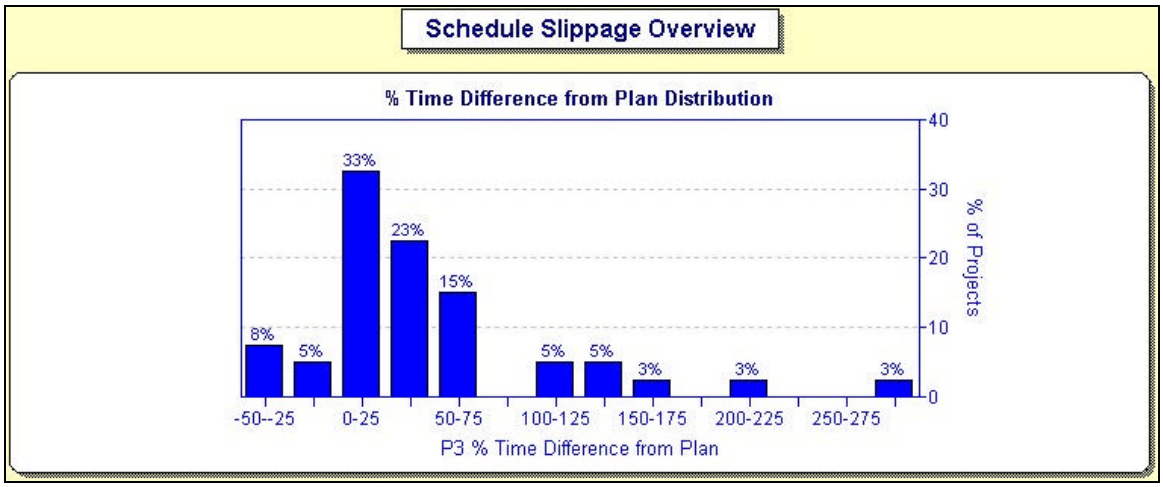



**Figure IV-4. Schedule Slippage Overview.**

## B. DEVELOPMENT OF MAPPING

This phase involves identifying suitable metrics from the QSM® data dictionary and applying these metrics to the original risk assessment model in order to document the model's accuracy. Appendix A contains QSM® data dictionary.

Projects in the QSM® repository have many attributes. However, only a small fraction of the data base metrics appears amenable to analysis when implementing the risk assessment model. Figure IV-5 presents a partial list of available metrics from QSM®. The colored cells are the actual measures that were considered, in some fashion, to interface with the risk assessment model.

| | | | | |
|---|---|---|---|---|
| Project Name | Status | Confidence | Record Creation Date | Last Modified |
| Application Type | Description | Size | New | Modified |
| Unmodified | Requirements | Productivity Index | Defects First Month after Delivery | Phase Acronym |
| Start Date | End Date | Months | Effort | Cost |
| Peak Staff | Staffing Shape | Organization | Division | Country |
| Design Complexity | Development Classification | Industry Sector | Function Unit | Counting Method |
| Gearing Factor | Language | Percent of Total Size | Language Type | Effort Units |
| Person Hours per Month | Labor Rate | Labor Rate Unit | Development Environment | Operational Environment |
| Day / Week | Hours / Day | Defects | Time (months) | Effort |
| Cost | Peak staff | Size and Requirements Growth | System Benefit / Effectiveness | Significant Factors |

*Majority of Available Measures from QSM®*

**Figure IV-5. Available Measures from QSM®.**

The SRM input parameters are unique in the sense that traditional software practitioners do not document *Efficiency*, *Requirements Volatility*, and *Complexity* according to the SRM definition. Due to this limitation, there is not one unique mapping that can be established to exercise the risk assessment model, rather it is necessary to develop a suite of potential metrics mapping scenarios. After studying the available metrics from QSM® and the original metric specification of the risk assessment model, a set of 27 unique test cases were developed. (Which range from the most logical interpretation of the metric definitions to an exaggerated modification of the metric elements.) The intent is to map each of the 112 projects, according to each of the 27 test case scenarios, and document the accuracy of the risk assessment model. (Appendix C details each of the 27 scenarios.)

Figure IV-6 shows how the available metrics from QSM® were mapped to the risk assessment model.

| Legend | EF | PI Breakpoint | | CX | | |
|---|---|---|---|---|---|---|
| | A | 10 | | A | Conversion KLOC = 40 * LGC | |
| | B | 13 | | B | Conversion KLOC = 30 * LGC | |
| | C | 18 | | C | Conversion KLOC = 20 * LGC | |
| | | | | | | |
| | RV | | | | | |
| | A | Req Growth % = Number used | | | | |
| | B | 0's in Req Growth replaced by 25% | | | | |
| | C | All Req Growth % increased by 25% | | | | |
| | | | | | | |

**Figure IV-6. Legend for Interface.**

Recall that the risk assessment model requires three primary elements to derive a projection date for software completion: *Efficiency*, *Requirements Volatility*, and *Complexity*[7]. Additionally, either a desired number of days[8] or a desired completion percentage is supplied to the risk assessment model. The next section reviews the three primary risk assessment elements and describes how measures from the QSM® database are mapped.

### 1. Efficiency (EF)

> The *efficiency* of the organization can be measured observing the fit between people and their roles in the software process. The skill match between the person and the job is required to estimate the speed in processing information and the rate of exceptions, which in turn affect the *efficiency*. The number of people and the turnover affect the *efficiency* as consequence of productivity losses due to training, learning curves and communications… (Nogu00)

Low and high *efficiency* scenarios are determined by the ratio of the percentage of direct time over the percentage of idle time. The resulting ratio is calculated by

[7] The first digit in the legend is *always* the efficiency and the second digit is *always* the requirements volatility. A third digit, when dealing with the SRM is the complexity.

[8] Do not confuse the number of days as effort in man-days. The SRM does not project the risk based on effort. Time in days is considered to be elapsed time. The practice of converting working days to calendar months is 22 working days per month (Nogu00).

$$EF = Direct\%/Idle\% \quad (4.1)$$

The breakpoint for the low and high *efficiency* indicator occurs when idle time exceeds approximately 33%. Low *efficiency* organizations, defined by idle time in excess of 33%, produce a resulting EF ratio less than 2.0. Calibrated ranges for a low *efficiency* organization are approximately 33% - 55% idle time. High *efficiency* organizations are organizations with idle time less than 33%.

The primary data field mapped from the QSM® database to represent EF is the productivity index.

> Organizations represented in the database, use a metric that is called a "productivity index" or PI. The PI of an organization, for a particular project, is a management scale from one to 36, corresponding to the productivity parameter, that represents the overall process productivity achieved by an organization during the main build. The PI is a measure that quantifies the net effect of everything that makes projects different from one another (QSMa).

In order to map the recorded productivity index to the risk assessment model, assumptions are made that when an organization exceeds a particular productivity threshold, the organization is considered either a high or low *efficiency* organization. The average productivity index in the data is around 10 ½. Using the average as a point of reference, the experiments were exercised with three different breakpoints.

As referenced in the legend, Figure IV-6 above, and Appendix C, an organization operating with a productivity index greater than 10 (mean value), 13 (plus one standard deviation), or 18 (plus two standard deviations) would be considered a highly efficient organization respectively. Therefore, the documented results demonstrate the sensitivity of the risk assessment model to adjusting these factors.

### 2. Requirements Volatility (RV)

The SRM utilizes a measure intended to capture the complications experienced during requirements elucidation. This measure is determined by recording the percentage of requirements from two different aspects. First, a determination is made of the percentage of requirements that have been eliminated from a software project (Death Rate, DR%), then conversely, the percentage of requirements that have been added to a software project (Birth Rate, BR%) are determined. Adding these two percentages yields a requirements volatility metric, as a percentage.

$$RV = BR\% + DR\% \tag{4.2}$$

QSM® contains a measure called Requirements Growth Percentage, which documents how much the project requirements changed from the original plan. The mapping between Requirements Growth Percentage and the Requirements Volatility occurs in three configurations. The first configuration uses a direct one-to-one mapping. The second configuration replaces any zero and non-entries[9] in the QSM® database with a baseline of 25%, leaving the values greater than zero unaltered[10]. The final configuration is a modification of the one-to-one mapping in which all Requirements Growth Percentage entries in the QSM® database are increased by a static value of 25% for the Requirements Volatility (intended to impose additional constraint on the SRM).

### 3. Complexity (CX)

The final primary element required for the risk assessment model is a measurement for the complexity of the software development. (Nogu00) derives a complexity measure from analyzing a Prototype System Description Language (PSDL) specification. This complexity measure is determined from the following equation:

---

[9] Seventy-two projects contained no values for *Requirements Growth Percentage*.

[10] The majority of projects (33%), contained requirements growth between zero and 25%. The intent is to replace the requirements growth non-entries with a similar distribution.

$$LGC = O + D + T \tag{4.3}$$

The equation accounts for all of the operators (O), data streams (D), and abstract data types (T) contained within a flattened PSDL source code and combines their frequency to determine a measure called the Large Granular Complexity (LGC). The LGC is predominately limited to analyzing PSDL code; however the SRM provides a conversion equation for lines of Ada code as follows.

$$KLOC = (40LGC + 150)/1000 \tag{4.4}$$

or roughly

$$KLOC = 40LGC/1000 \tag{4.5}$$

where

KLOC = thousands of lines of code

The QSM® database provides excellent information detailing the coding effort of software projects. Information is available that documents the percentage of new, used, and modified code as well as the total counts in lines of code. The capability exists to segment the project data by development language. The mapping strategy involves calculating the total amount of Ada code (new or modified) applied to the software project and then converting this value to LGC.

Preliminary results indicated that SRM conversion from KLOC to LGC is possibly too conservative. Therefore, additional scenarios were set up to test the conversion formula for

$$KLOC = 30LGC/1000 \tag{4.6}$$

and

$$KLOC = 20LGC/1000 \tag{4.7}$$

The result is three different configurations for each of the three primary elements required for the risk assessment model, for a total of $3^3 = 27$ possible combinations of

mapping scenarios. The life cycle phases considered span the completion of the specifications thru the delivery of the code. The maintenance phase is excluded from the validation experiments.

## C. VALIDATION RESULTS

### 1. SRM Performance

Figure IV-7 shows the performance of the SRM and requires additional discussion. Plotted are the 112 software projects on a scatter diagram with the effective source lines of code (E-SLOC)[11] serving as the horizontal axis and the project duration in months plotted on the vertical axis. Using the mapping scenario AAA[12], the projections of the SRM are plotted[13]. Visually one can deduce two facts: (1) The SRM is projecting the software project completion times very optimistically, and (2) Projections follow a bi-polar pattern.

(Nogu00) details that the closest mapping by definition, scenario AAA, is logically the best mapping for validation. However, scenario AAA is too optimistic. In 98.2% of the projects exercised, the risk assessment model predicts project completion of scenario AAA well before the actual completion time. The average projected completion time was 21% of the actual value (i.e. if a project required 36 months to complete, the SRM predicts on average 7 ½ months).

The closest match of the 27 scenarios, scenario BCC, still exhibits an average absolute error projecting 45% of the actual value. The realization is that on average the actual projects took a substantially longer time to complete than what the SRM predicts.

---

[11] As defined by (Putn92); A measure of the size or functionality of a software system. Counts executable source lines deliverable to customer/user, thus excluding environmental or scaffolding code. May include estimate of equivalent new lines in reused or modified modules. Also known as source statements.

[12] Effiency breakpoint is a productivity index of 10. Requirements are mapped one-to-one. Complexity is derived with the constant 40.

[13] The SRM projects the probability of project completion. To make this comparison, it was necessary to fix the completion percentage and determine the days to achieve this probability. The probabilities for every SRM conversion in this dissertation are fixed at 95%, as recommended by (Nogu00).

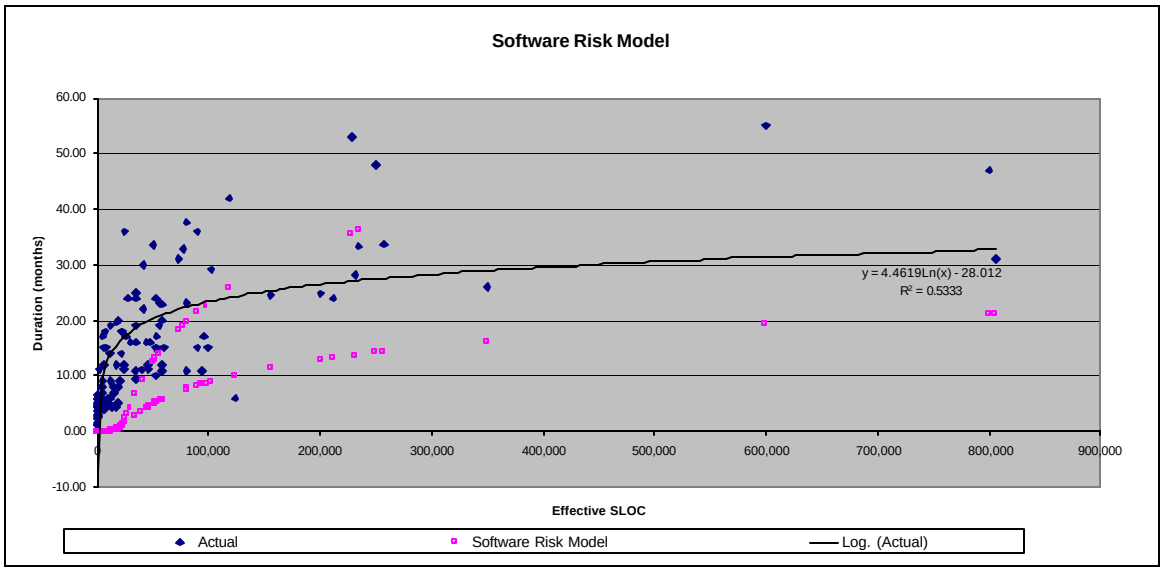



**Figure IV-7. SRM Projection of Actual Projects.**

In order to better understand the accuracy of the SRM, the model's performance is compared against two known industry standards, Simplified Software Equation (Putn92) and the Basic Constructive Cost Model (Boeh81). Three[14] forms of the Basic COCOMO and the Simplified Software Equation are tested against the actual project data. The implementation details of this comparison can be found in Appendix C.

## 2. Simplified Software Equation Performance

Figure IV-8 below illustrates the projection of the Simplified Software Equation with three different productivity indexes (16, 13, and 9)[15]. The horizontal and vertical axes remain consistent with the previous chart. Due to the small number of actual projects in some sectors, the projects are not segregated by their industry sectors. The

[14] The equations implemented were chosen because the original author provided these equations for quick and easy macro estimations. Both of these authors provide more detailed estimation techniques which are not addressed in the dissertation.

[15] These values were chosen after consultation with Lawrence Putnam, Sr. who was asked to correlate his productivity values with the three types of applications represented by Basic COCOMO.

Simplified Software Equation yields the smallest error when applied to the appropriate types of projects. However, for purposes of obtaining a better understanding of the model's performance, treating the projects as one of three application types will suffice. The Simplified Software Equation, with these three productivity indexes, projects on average between 47 and 65 percent accuracy (absolute error) and an average between 64 and 89 percent (actual error). For example, with a productivity index of 13, the Simplified Software Equation predicted all of the software project durations with an absolute error of 35 percent[16].

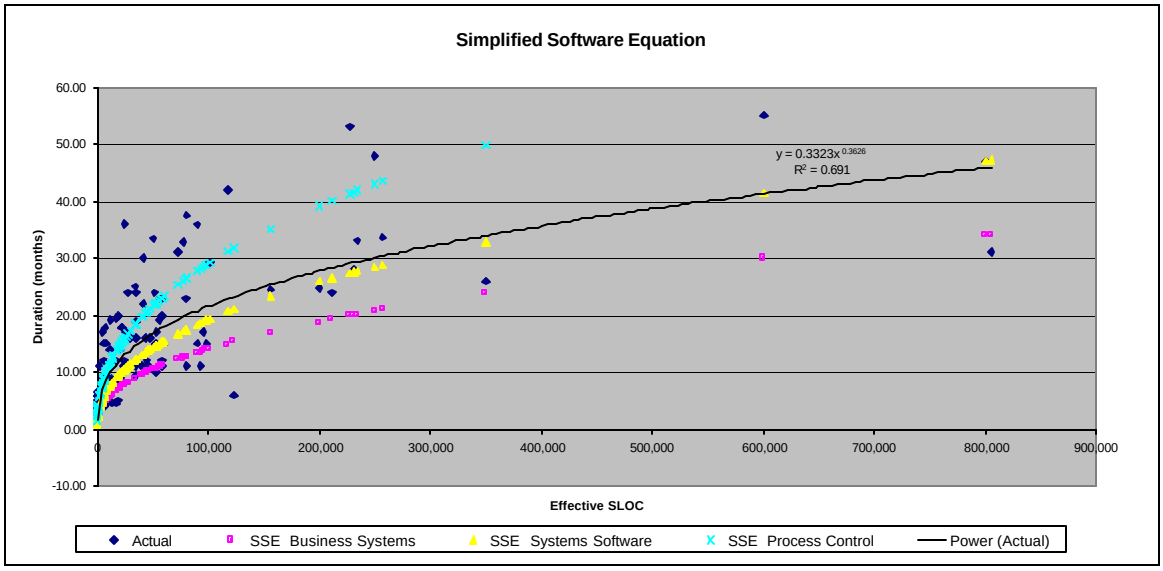



**Figure IV-8. SSE Projections of Actual Projects.**

### 3. COCOMO Performance

Figure IV-9 illustrates the behavior of Basic COCOMO against the actual project data. Basic COCOMO provides three macro models for estimation depending on the type of software development. As with the Simplified Software Equation, no distinction is made in the different application types of the 112 test projects. Interestingly, there

16 It remains important to reiterate that the Simplified Software Equation is not being treated in accordance with its original design. The Simplified Software Equation produces the least error when properly aligned with a productivity parameter. Additionally, several project projections were below the recommended minimum values (Putn92).

seems to be little variation in the three COCOMO Models. This is also evident in the small standard deviation in the degree of error between the three implementations of Basic COCOMO. The Basic COCOMO projected[17], on average, the actual project duration with an absolute error of about 35 percent[18].

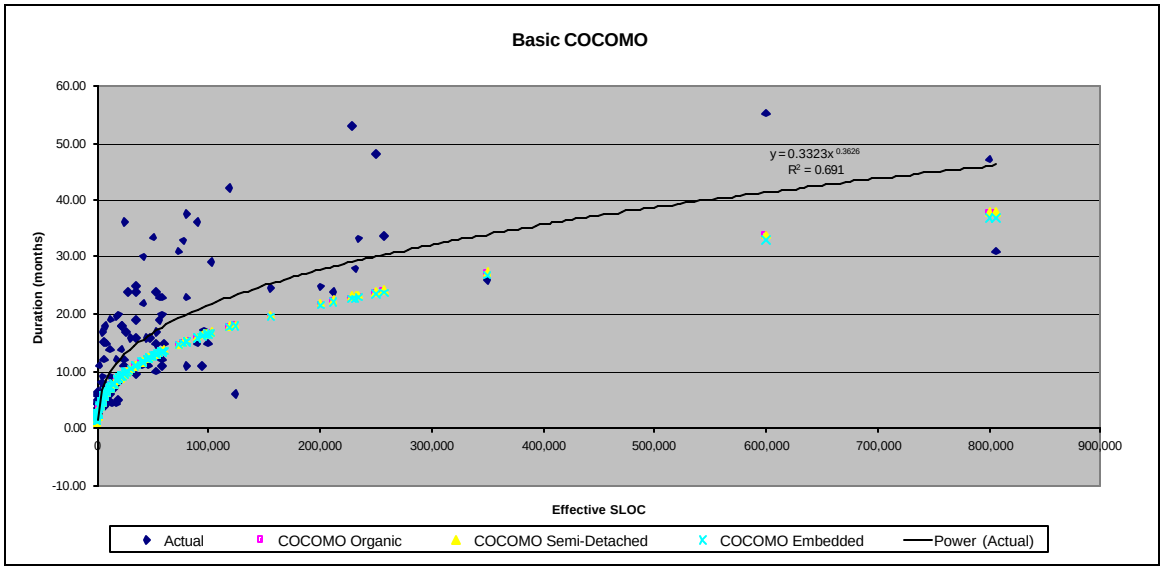



**Figure IV-9. Basic COCOMO Projections of Actual Projects.**

Boehm (Boeh81) provides this reasoning for the similarities in the duration projections:

- For software products of the same size, the estimated effort is considerably greater and the estimated productivity is considerably less for the embedded mode
- For larger products, the drop off in productivity (diseconomy of scale) is greater for the embedded mode
- *The estimated schedule as a function of product size is about the same for all three modes*

[17] (Boeh81) indicates that the $T_{dev}$ considers the Main Build and the time required to produce the software specifications. The SRM and the Simplified Software Equation consider the Main Build beginning after the completion of the software specifications. The model comparisons account for the difference. All COCOMO projections are adjusted according to (Putn92). $T_{dev\text{-}feas} = T_{dev\text{-}min} / 3$.

[18] The graphs of both the Software Equation and the COCOMO demonstrate the highly volatile nature of predicting software project durations. However, the error percentages were derived from the average error on the individual data points, not the data trend lines.

- For software projects requiring the same amount of development time, the embedded mode projects will consume more effort

### 4. Consolidated Results

Table IV-2 presents a comparison of the models utilized in this validation. For comparison purposes, the 27 SRM models were reduced down to six. These six model implementations bound the results produced by exhausting all 27 scenarios. With the reduction of the SRM models, a total of 12 model implementations are presented from the analysis.

To fully validate each of the model's projections, five rating criteria were established; weighted according to their relevance to the model's success. The following, in descending order of priority, is a complete list of the rating criteria. The implementation details are contained in Appendix C.

- Actual Error – a measure of the average error produced by each model calculated by $\text{actual\_error} = \frac{\text{projected} - \text{actual}}{\text{actual}}$. Negative values indicate an under-estimate.
- Absolute Error – a measure of the average error produced by each model calculated by $abs(\text{actual\_error})$. There is no distinction between whether the model projected high or low. Useful for providing a single measure of the relative error.
- Balance – A model that projects too optimistically can prove disastrous to a software project. The model is evaluated on how evenly it projects estimates (i.e. are the over-estimates proportional to the under-estimates). A model projecting with closer "balance" receives a higher rating.
- Under Estimation – The fourth consideration is designed to evaluate the actual error of an optimistic projection. This criterion only considers the individual under projections and computes their average.
- Over Estimation – The least weighted criteria is over estimation. In actuality, less damage can potentially occur from projecting too cautiously. This criterion only considers the individual over projections and computes their actual error.

| | | | Actual Error (wt 5) | | | | Absolute Error (wt 4) | | | | Balance (wt 3) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **N** | **CORREL** | **Mean** | **Std** | **Rank** | **Score** | **Mean** | **Std** | **Rank** | **Score** | **% Under** | **Rank** | **Score** |
| SRM - AAA | 111 | 0.8214 | -77% | 28% | 12 | 60 | 78% | 24% | 12 | 48 | 98% | 12 | 36 |
| SRM - ACC | 111 | 0.8485 | -55% | 41% | 9 | 45 | 59% | 35% | 8 | 32 | 95% | 10 | 30 |
| SRM - BAA | 111 | 0.8403 | -71% | 35% | 11 | 55 | 73% | 31% | 11 | 44 | 96% | 11 | 33 |
| SRM - BCC | 111 | 0.8686 | -43% | 51% | 8 | 40 | 55% | 38% | 7 | 28 | 80% | 7 | 21 |
| SRM - CAA | 111 | 0.7645 | -57% | 54% | 10 | 50 | 70% | 34% | 10 | 40 | 83% | 8 | 24 |
| SRM - CCC | 111 | 0.7952 | -18% | 78% | 2 | 10 | 69% | 41% | 9 | 36 | 62% | 1 | 3 |
| SSE - Business Systems | 111 | 0.7815 | -36% | 33% | 7 | 35 | 42% | 24% | 5 | 20 | 88% | 9 | 27 |
| SSE - Systems Software | 111 | 0.7815 | -11% | 45% | 1 | 5 | 36% | 30% | 3 | 12 | 68% | 3 | 9 |
| SSE - Process Control | 111 | 0.7815 | 34% | 69% | 6 | 30 | 53% | 55% | 6 | 24 | 37% | 2 | 6 |
| COCOMO - Organic | 111 | 0.785 | -20% | 40% | 5 | 25 | 36% | 26% | 3 | 12 | 75% | 6 | 18 |
| COCOMO - Semi-Detached | 111 | 0.7857 | -18% | 41% | 2 | 10 | 35% | 27% | 1 | 4 | 71% | 4 | 12 |
| COCOMO - Embedded | 111 | 0.7864 | -18% | 40% | 2 | 10 | 35% | 26% | 1 | 4 | 71% | 4 | 12 |

| | | | Under Estimation (wt 2) | | | | | Over Estimation (wt 1) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **N** | **CORREL** | **N-under** | **Mean** | **Std** | **Rank** | **Score** | **N-over** | **Mean** | **Std** | **Rank** | **Score** | **Total** |
| SRM - AAA | 111 | 0.8214 | 109 | -79% | 23% | 12 | 24 | 2 | 37% | 40% | 7 | 7 | 175 |
| SRM - ACC | 111 | 0.8485 | 105 | -61% | 34% | 8 | 16 | 6 | 38% | 48% | 8 | 8 | 131 |
| SRM - BAA | 111 | 0.8403 | 107 | -74% | 29% | 10 | 20 | 4 | 27% | 26% | 1 | 1 | 153 |
| SRM - BCC | 111 | 0.8686 | 89 | -61% | 37% | 7 | 14 | 22 | 31% | 30% | 5 | 5 | 108 |
| SRM - CAA | 111 | 0.7645 | 92 | -77% | 32% | 11 | 22 | 19 | 40% | 27% | 10 | 10 | 146 |
| SRM - CCC | 111 | 0.7952 | 69 | -70% | 37% | 9 | 18 | 42 | 67% | 47% | 11 | 11 | 78 |
| SSE - Business Systems | 111 | 0.7815 | 98 | -44% | 21% | 6 | 12 | 13 | 27% | 40% | 1 | 1 | 95 |
| SSE - Systems Software | 111 | 0.7815 | 76 | -34% | 21% | 2 | 4 | 35 | 39% | 44% | 9 | 9 | 39 |
| SSE - Process Control | 111 | 0.7815 | 41 | -26% | 16% | 1 | 2 | 70 | 70% | 63% | 12 | 12 | 74 |
| COCOMO - Organic | 111 | 0.785 | 83 | -37% | 21% | 5 | 10 | 28 | 30% | 39% | 3 | 3 | 68 |
| COCOMO - Semi-Detached | 111 | 0.7857 | 79 | -37% | 20% | 3 | 6 | 32 | 30% | 38% | 3 | 3 | 35 |
| COCOMO - Embedded | 111 | 0.7864 | 79 | -37% | 20% | 3 | 6 | 32 | 31% | 37% | 5 | 5 | 37 |

**Table IV-2. Validation Results.**

The columns in Table IV-2 represent the following:

- *Model* – identifies the model being recorded
- *N* – the total number of projects considered by the model
- *CORREL* – the correlation coefficient. The correlation coefficient is used to determine the relationship between two properties (MSEX02). The number is interpreted as the proportion of the total variation (SPSS99).
- *Actual Error (wt 5)* – determined by

$$\text{actual\_error} = \frac{\text{projected} - \text{actual}}{\text{actual}} \tag{4.8}$$

  - *Mean* – the average error
  - *Standard Deviation* – of the average
  - *Rank* – low is best, high is the worst
  - *Score* – $\text{Score}_{actual\_error} = \text{rank}_{actual\_error} * \text{weight}_{actual\_error}$

- *Absolute Error (wt 4)* – determined by *abs*(actual_error)
  - *Mean* – the average
  - *Standard Deviation* – of the average error

- *Rank* – low is best, high is the worst
- *Score* - $\text{Score}_{absolute\_error} = \text{rank}_{absolute\_error} * \text{weight}_{absolute\_error}$

- *Balance (wt 3)* – A measure of the dispersion below and above the actual value.
  - *% Under* – the percentage of N that is projected below actual values
  - *Rank* – Values closer to 50% receive the highest ranking. This indicates that approximately 50% fall below and 50% fall above the actual values.
  - *Score* - $\text{Score}_{balance} = \text{rank}_{balance} * \text{weight}_{balance}$

- *Under Estimation (wt 2)* – Of the conservative projections, what is their average error?
- *N-under* – the raw number of projections short of the actual value.
  - *Mean* – the average error
  - *Standard Deviation* – of the average error
  - *Rank* – low is best, high is the worst
  - *Score* - $\text{Score}_{under\_error} = \text{rank}_{under\_error} * \text{weight}_{under\_error}$

- *Over Estimation (wt 1)* – Of the over-optimistic projections, what is their average error?
- *N-over* – the raw number of projections over the actual value.
- *Mean* – the average error
- *Standard Deviation* – of the average error
- *Rank* – low is best, high is the worst
- *Score* - $\text{Score}_{over\_error} = \text{rank}_{over\_error} * \text{weight}_{over\_error}$
- *Total* – a smaller number is better.

$$total = Score_{actual\_error} + Score_{absolute\_error} + Score_{balance} + Score_{under\_error} + Score_{over\_error}$$

Appendix C contains the implementation details. However, Table IV-2 illustrates that the *embedded* configuration of the Basic COCOMO produces the best overall results

with the Simplified Software Equation (systems software) performing a close second. One interesting point is the correlation coefficient. The SRM model performed worst in practically every scenario, yet the series has a higher correlation coefficient. This is even more evident in Figure IV-7, Figure IV-8, and Figure IV-9.

The SRM projects one of two possible values for any situation; discussed in detail in *Section D. Issues with the SRM*. The Simplified Software Equation and Basic COCOMO model produces a static projection based upon a pre-defined equation appropriate to their unique situation. The COCOMO family provides three of these static models (*organic*, *semi-detached*, and *embedded*). All software development is categorized within these three choices. The Simplified Software Equation provides more fidelity allowing the user to adapt using a productivity parameter closer to the actual situation.

## D. ISSUES WITH THE SRM

The SRM exhibits unsettling results as seen in the two distinct projection lines in Figure IV-7. This leads to the suspicion that the SRM does not accurately account for development time because it only projects two possibilities independent of the software complexity. To test this hypothesis, an experiment was conducted to see how the SRM would predict software development risks under the following conditions:

Nine scenarios make up the test:

- 0.8 < High Efficiency = 0.99 (direct time). Test uses (0.9)
- 0.6 < Medium Efficiency = 0.8 (direct time). Test uses (0.7)
- 0.4 < Low Efficiency = 0.6 (direct time). Test uses (0.5)
- High Requirements > 40% (requirements volatility). Testing 50%
- 20% < Medium Requirement = 40% (requirements volatility) Testing 30%
- 0% < Low Requirements = 20% (requirements volatility). Testing 0%
- Complexity = $KLOC = (40LGC + 150)/1000$

Representing the software development with the nine different scenarios should yield nine different software risk projections from the SRM. Figure IV-10 illustrates a different result. The horizontal axis represents time in days[19], and the vertical axis the probability of project completion. Read Figure IV-10 as if asking a risk assessment question. In this experiment, a software product is built that contains approximately 160K of E-SLOC, which needs to be produced on or before 30 months (target value = 660 days).

The single vertical line is located at the 660 days mark on the horizontal axis. Now ask the question: If the software organization demonstrates these types of characteristics: high, medium, or low *efficiency*, and the project is experiencing high, medium, or low requirements volatility, what is the probability of completing the 160K E-SLOC project on or before day 660?

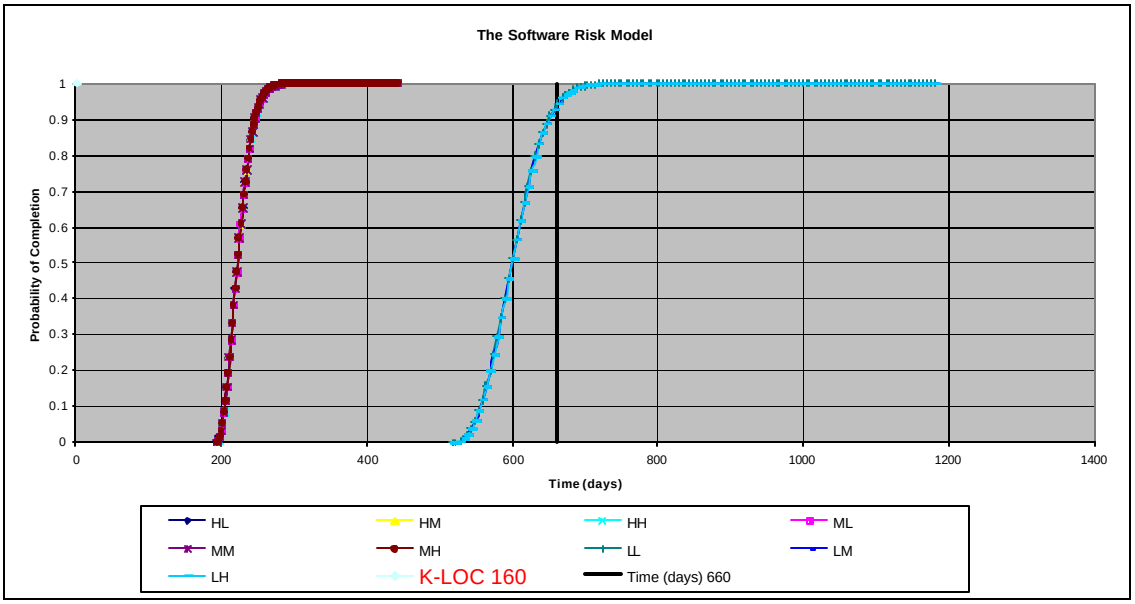



**Figure IV-10. SRM Project Completion Projections.**

[19] If the desire is to convert to months, assuming only 22 work days per month, divide the days by 22 (Nogu00).

The risk projection illustrates that for a *low efficiency* (test value 0.5 direct time) and *high* requirements volatility (test value 50% volatility) project[20], the probability of completion, on or before day 660, is about 92%. Additionally, for a *medium efficiency* and *high* requirements volatility project, the probability of completion, on or before day 660, is 100%. Intuitively this is correct; a low *efficiency* organization should experience a lower probability of success than a medium *efficiency* organization (everything else being equal).

However, where are the other seven projections: HL, HM, HH, ML, MH, LL, and LM? The graph properly represents all nine projections, showing them in one of two possible locations. If this is true, then the SRM can only deliver two possibilities[21] for a fixed E-SLOC, regardless of the *efficiency* or the requirements.

To trace the cause of this phenomenon, the mathematical implementation of the SRM (Figure IV-11) from (Nogu00) is examined below:

*Algorithm Model 4:*

```
// Inputs: EF, RV, CX, t
// t is given in days, we assume 22 days per month
// Output: p = P(x<=t)
If (EF > 2.0)then begin
        α = 1.95;
        γ = 22 * ln(1 + LGC^4.5/exp(28.5))*(1 + 0.0045 * RV);
        β = (γ/5.71);
        end
   else  begin
        α = 2.5;
        γ = 22 * ln(1 + LGC^12/exp(76))*(1 + 0.0045 * RV);
        β = (γ/5.47);
        end;
p = 1 - exp(-(((t - γ)/β)^α));
```

**Figure IV-11. SRM Algorithmic Model.**

[20] Indicated on the legend of Figure IV-10 as only LH. The first digit in the legend is *always* the efficiency and the second digit is *always* the requirements volatility. A third digit, if present, is the complexity.

[21] There exist actual differences in the plot when the requirements volatility is changed. However, the changes are statistically insignificant.

The SRM is designed around the *efficiency* of an organization. If an organization is considered high *efficiency* (EF > 2.0), then SRM implements the first curve, otherwise it implements the second curve. Recall that the breakpoint between a low and a high *efficiency* organization is 66.7%:

$$EF = \frac{direct\_time\%}{indirect\_time\%}$$

or

$$EF = 1.94 = \frac{66\%}{34\%}$$

If the organization is capable of achieving 66.7% *Direct Time*, it is considered a *high efficiency* organization. In our example:

$$EF = 2.003 = \frac{66.7\%}{33.3\%}$$

which translates into a difference of about 400 days in Figure IV-10. The implementation of the SRM limits the possible outcomes to two. Essentially, given the E-SLOC, all a manager needs to know is if the organization is *high* or *low efficiency*. All responses from the model will be statistically the same. A proper implementation of a model should be sensitive to changing input conditions, which the SRM does not account for in this case.

Recall that validation of the SRM is against simulation and that the SRM was constructed from observing the behavior of a simulator and then validated when a mathematical model (the SRM) replicated similar results. The model's accuracy was determined by how closely it could replicate the simulation. Without external validation, the SRM has a single point of failure. If there was a problem with the behavior of the simulation, then the construed mathematical representation would also faithfully reflect the same concerns. Chapter Five explores this notion further in more detail.

Two other concerns arise from Figure IV-10. The specific attributes contributing to Figure IV-10 are irrelevant. The figure would produce the same pattern regardless of the input parameters.

Consider the LH scenario in Figure IV-10. The plot demonstrates approximately a 10% chance of completing the project by approximately day 570. However, the project has 90% chance of successful completion by approximately day 630, or 60 days later. Additional time increases the probability of completion by 1.5 percent every day. No matter what this organization does, aside from becoming greater than an *efficiency* of two, there is a zero percent chance of completing the project before day 500.

Similar results occur when considering the scenario MH. The range of days between 10% and 90% is only about 40 days. The SRM claims an organization in this situation can increase the probability of completion by 2.25% each day. However, this does not begin until the project has been in existence for about 200 days.

In summary, the range of project success is not realistic. An organization going from 10% success to 90% success ranges from 21 days to about 200 days, depending on the *efficiency* and the E-SLOC. The second point is the notion that an organization cannot improve the probability of success rate regardless of what actions are taken. Applying additional people to work on the software project creates no change (either better or worse). If the personnel receive training or use a better tool suite, the probability of success does not change. Again, the only change in the probability of success is crossing the break point on *efficiency*. This break point causes the graph to shift around 50 days in the smallest projects, to around 400 days in the largest projects.

The intent of phases one, two, and three is to assess confidence in the risk assessment model. As the analysis shows, little confidence is warranted in the risk assessment model. No realistic scenario mapped during the experiments provides confidence in the risk assessment model's performance. The results presented in this chapter and Appendix C require a "fresh" approach to projecting software project risk, an approach which addresses the prevailing issues with the SRM, and develops a more robust software risk assessment model.

THIS PAGE INTENTIONALLY LEFT BLANK

# V. SIMULATION CALIBRATION

Phases one, two, and three were intended to establish confidence in the risk assessment model. Unfortunately, this cannot occur with the SRM in its current form. The research focus now shifts to determine the "root causes" for the SRM failure; challenging previous assumptions in the SRM. The most contentious assumption is the suitability of VitéProject to simulate the development of software projects.

This chapter presents the foundation for challenging this assumption and begins by documenting replication of the underlying data in the SRM. *Efficiency* is increased, when using the VitéProject simulation, through the development of an Application Program Interface (API) developed by (Alex01). A software development benchmark is established that is useful to gauge software development trends. And finally, a method is proposed to calibrate the VitéProject for software development. Ultimately, evidence is presented that questions VitéProject's usefulness for this purpose.

## A. DISSERTATION REPLICATION AND VITÉPROJECT API

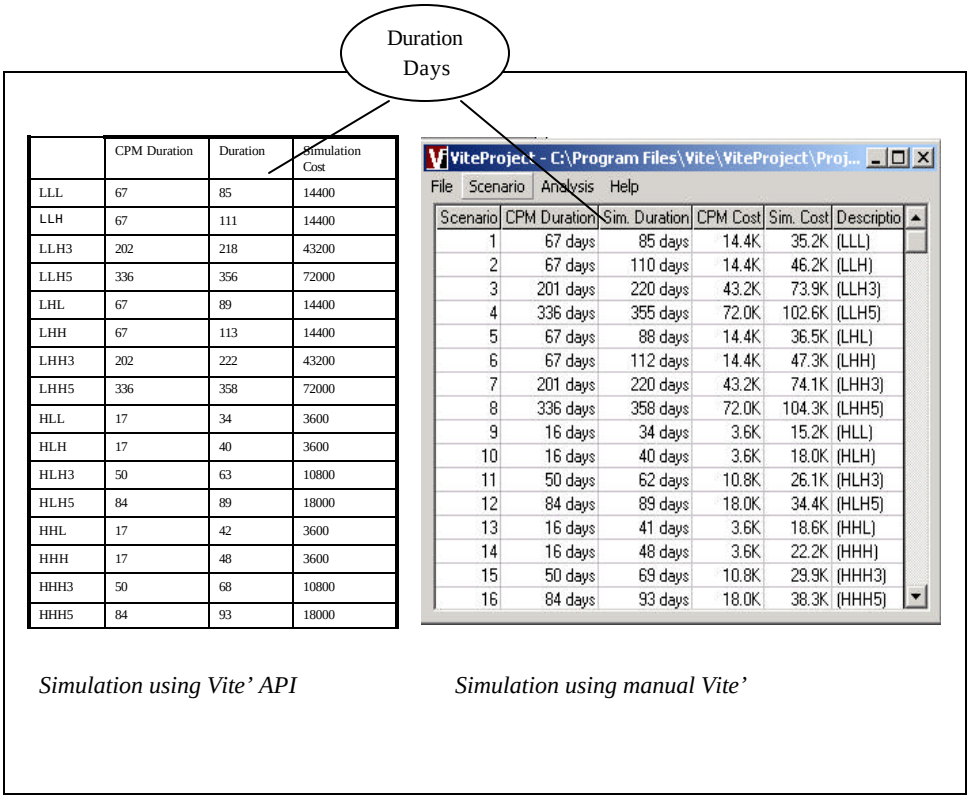
The first task in improving the SRM was to recreate the documented analysis in the development of the SRM. This should have been a straightforward process and the intent was to use the original dissertation (Nogu00) as a baseline to continue the research. The discovery however, was quite to the contrary.

Difficulties in duplicating the original SRM data surfaced while validating the VitéProject API (Alex01)[22]. To test the API, an experiment was established to replicate the simulation parameters in the SRM data (Nogu00). A correct API should produce results equivalent to simulating without the use of the API. After excessive attempts to validate the API against the SRM results, no results ever matched.

[22] The API increases the efficiency of VitéProject. The API automates the establishment of simulation scenarios and automatically records the simulation data in a Microsoft Excel spreadsheet.

Additional experiments were conducted without regard to the original SRM data; which provided conclusive evidence the API functioned correctly. First, simulations were conducted without the use of the API or using the SRM documentation. Second, the same experiments were conducted using the API (all configuration information remained fixed). Figure V-1 demonstrates that fixed simulation inputs produce results within a close approximation to one another. Simulations by definition are stochastic; thus, accounting for the slight variation in the durations.

After completing a working API, the fact remained that the results were not consistent with the foundation data of the SRM. Additional investigation ultimately determined the source of the inconsistencies.




| | CPM Duration | Duration | Simulation Cost |
|---|---|---|---|
| LLL | 67 | 85 | 14400 |
| LLH | 67 | 111 | 14400 |
| LLH3 | 202 | 218 | 43200 |
| LLH5 | 336 | 356 | 72000 |
| LHL | 67 | 89 | 14400 |
| LHH | 67 | 113 | 14400 |
| LHH3 | 202 | 222 | 43200 |
| LHH5 | 336 | 358 | 72000 |
| HLL | 17 | 34 | 3600 |
| HLH | 17 | 40 | 3600 |
| HLH3 | 50 | 63 | 10800 |
| HLH5 | 84 | 89 | 18000 |
| HHL | 17 | 42 | 3600 |
| HHH | 17 | 48 | 3600 |
| HHH3 | 50 | 68 | 10800 |
| HHH5 | 84 | 93 | 18000 |

| Scenario | CPM Duration | Sim. Duration | CPM Cost | Sim. Cost | Descriptio |
|---|---|---|---|---|---|
| 1 | 67 days | 85 days | 14.4K | 35.2K | (LLL) |
| 2 | 67 days | 110 days | 14.4K | 46.2K | (LLH) |
| 3 | 201 days | 220 days | 43.2K | 73.9K | (LLH3) |
| 4 | 336 days | 355 days | 72.0K | 102.6K | (LLH5) |
| 5 | 67 days | 88 days | 14.4K | 36.5K | (LHL) |
| 6 | 67 days | 112 days | 14.4K | 47.3K | (LHH) |
| 7 | 201 days | 220 days | 43.2K | 74.1K | (LHH3) |
| 8 | 336 days | 358 days | 72.0K | 104.3K | (LHH5) |
| 9 | 16 days | 34 days | 3.6K | 15.2K | (HLL) |
| 10 | 16 days | 40 days | 3.6K | 18.0K | (HLH) |
| 11 | 50 days | 62 days | 10.8K | 26.1K | (HLH3) |
| 12 | 84 days | 89 days | 18.0K | 34.4K | (HLH5) |
| 13 | 16 days | 41 days | 3.6K | 18.6K | (HHL) |
| 14 | 16 days | 48 days | 3.6K | 22.2K | (HHH) |
| 15 | 50 days | 69 days | 10.8K | 29.9K | (HHH3) |
| 16 | 84 days | 93 days | 18.0K | 38.3K | (HHH5) |

*Simulation using Vite' API* *Simulation using manual Vite'*

**Figure V-1. Comparison Between API & Manual Simulations.**

The development data of the SRM does not accurately document the parameters used in the underlying simulations that helped develop the risk assessment model. Experiments were conducted to find the extent of the error in the documentation. Recall

that the development of the SRM documents 30 simulation trials for 16 scenarios, for a total of 480 individual simulation executions (Nogu00).

Using the VitéProject API, 3000 simulation trials were conducted for 27[23] base scenarios and for nine[24] extended scenarios (xxH5), for a total of 108,000 individual simulation executions. The purpose of these simulation trials is to determine the actual simulation parameters used for the SRM. The possible error is narrowed down to the establishment of the simulation probabilities. To test this hypothesis, the probabilities were altered one at a time to seek convergence on the (Nogu00) results, each alteration exercised 108,000 times. Table V-1 illustrates the actual parameters that were tested in VitéProject.

| *Probability Code* | *Functional Error Rate* | *Project Error Rate* | *Informational Exchange* | *Noise* |
|---|---|---|---|---|
| **A** | 0.01 | 0.01 | 0.8 | 0.10 |
| **B** | 0.01 | 0.01 | 0.8 | 0.01 |
| **C** | 0.01 | 0.10 | 0.8 | 0.10 |
| **D** | 0.01 | 0.10 | 0.8 | 0.01 |
| **E** | 0.10 | 0.01 | 0.8 | 0.10 |
| **F** | 0.10 | 0.01 | 0.8 | 0.01 |
| **G** | 0.10 | 0.10 | 0.8 | 0.10 |
| **H** | 0.10 | 0.10 | 0.8 | 0.01 |

*Probability Code A is the documented probabilities from (Nogu00).*
*Probability Code G is the closest match for analysis.*

**Table V-1. Probability Codes.**

This table was developed by starting with the documented simulation settings, Probability Code A, and altering the values towards the default simulation settings. Appendix D documents the actual results. Due to the stochastic nature of the experiments, the results are not identical. Success is achieving any results that are within one standard deviation[25] of the original SRM results. As pointed out above, Probability Code A, is the documented probabilities from (Nogu00). However, Probability Code G

[23] This dissertation conducts 27 base scenarios. The original work (Nogu00) conducted eight base scenarios.

[24] This dissertation conducts nine extended scenarios. The original work (Nugu00) conducted four extended scenarios.

[25] Standard deviations were derived from (Nogu00), see Table V-2.

is in actuality the closest match. An interesting occurrence is even when executing the simulation according to Probability Code G, there is not a reasonable match when the complexity multiple is greater than one.

This dissertation documents which probability settings produced the documented results in (Nogu00). However, upon closer examination, it becomes evident that the simulations do not produce, even with the corrected probability settings, the expected results when the FTE[26] factor is greater than one (i.e. a simulation scenario of xxH5). The research ultimately reproduced the foundation data of the SRM. Appendix D documents how to derive the foundation data of the SRM using Microsoft Excel.

## B. SOFTWARE DEVELOPMENT BENCHMARKS

Another issue remains in the SRM documentation. When referencing Table 5.3 from (Nogu00), a rational determination of the origin of the LGC column is difficult to differentiate. Table 5.3 from the original research is provided below as Table V-2. The suspect correlation is the relationship between the software complexity and the actual LGC. An extract from (Nogu00) is provided below:

> The complexity levels L and H correspond to 781 LGC. However, applying the value "low" to the parameter Solution Complexity, the result is a decrease in the total complexity to 746 LGC. The complexity values for scenarios with complexity levels H2.5 and H5 correspond to the estimated development times for the project considering that the time for each activity is increased by a factor 2.5 and 5 respectively. These values correspond to 1334 LGC and 3230 LGC respectively. These four values of LGC are used in all the scenarios (see Table 5.3).

[26] A representation of the number of Full-time equivalent personnel assigned to conduct an activity.

| Scenario | EF | RV | CX | E(t) days | SD(t) days | CPM (days) | LGC | Months | days |
|---|---|---|---|---|---|---|---|---|---|
| LLL | L | L | L | 88 | 5 | 67 | 746 | 3.61 | 79 |
| LLH | L | L | H | 101 | 6 | 67 | 781 | 4.25 | 93 |
| LLH2.5 | L | L | H2.5 | 254 | 16 | 168 | 1334 | 11.74 | 258 |
| LLH5 | L | L | H5 | 507 | 31 | 335 | 3230 | 24.12 | 531 |
| LHL | L | H | L | 101 | 7 | 67 | 746 | 3.61 | 79 |
| LHH | L | H | H | 128 | 10 | 67 | 781 | 4.25 | 93 |
| LHH2.5 | L | H | H2.5 | 319 | 25 | 168 | 1334 | 11.74 | 258 |
| LHH5 | L | H | H5 | 638 | 49 | 335 | 3230 | 24.12 | 531 |
| HLL | H | L | L | 32 | 2 | 67 | 746 | 3.61 | 79 |
| HLH | H | L | H | 42 | 3 | 67 | 781 | 4.25 | 93 |
| HLH2.5 | H | L | H2.5 | 105 | 7 | 168 | 1334 | 11.74 | 258 |
| HLH5 | H | L | H5 | 209 | 14 | 335 | 3230 | 24.12 | 531 |
| HHL | H | H | L | 42 | 3 | 67 | 746 | 3.61 | 79 |
| HHH | H | H | H | 49 | 4 | 67 | 781 | 4.25 | 93 |
| HHH2.5 | H | H | H2.5 | 122 | 9 | 168 | 1334 | 11.74 | 258 |
| HHH5 | H | H | H5 | 244 | 18 | 335 | 3230 | 24.12 | 531 |

**Table V-2. Table 5.3 from "Nogu00".**

The question of how is the LGC column derived remains. There exists no logical evidence that establishing VitéProject parameters to represent xxH1 is any different than the parameters of xxH5. Thus, there is currently no way to determine, with confidence, what the VitéProject simulations projections actually represent. Research needs to explore ways to validate the simulated results and determine a way to benchmark software development.

Recall that all three variations of the Basic COCOMO produced very similar project durations; regardless of the type of software development (Boeh81) (Chapter IV). Expanding this experiment to determine the *average staff* and minimum *effort* projected by the Basic COCOMO provides a partial foundation of a software development benchmark. The remaining benchmark is derived by projecting the Simplified Software Equation over the same range of input criteria. As with Chapter IV, the productivity parameters of 9, 13, and 16[27] remain for the Simplified Software Equation. The experiment produced Figure V-2.

[27] Lawrence Putnam recommended these numbers. The original intent was to reasonably correlate the productivity index with the three implementations of the Basic COCOMO.

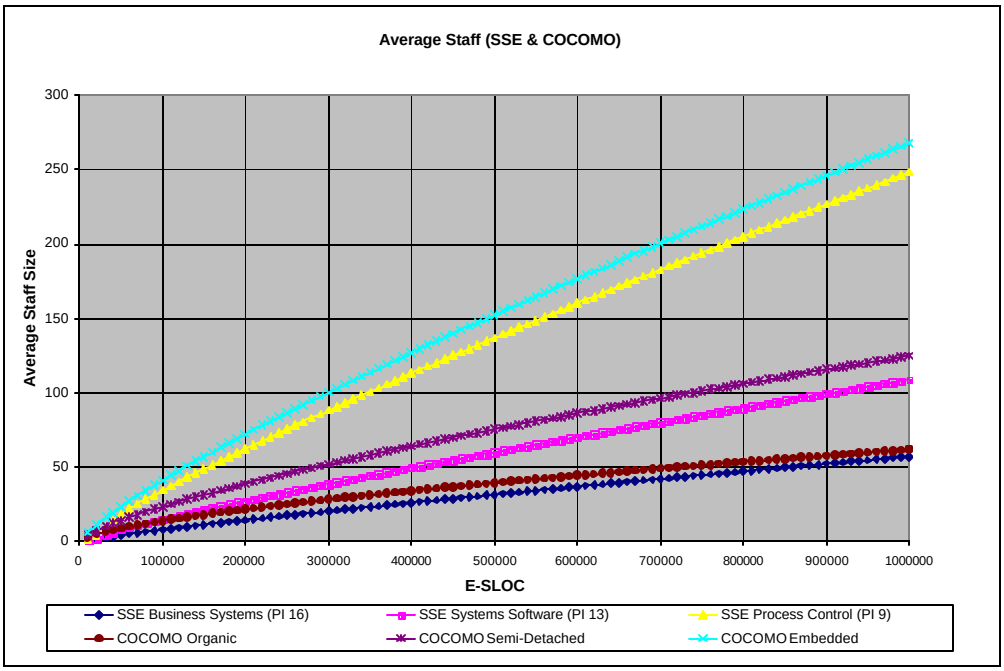



**Figure V-2. SSE and COCOMO Average Staff Projections.**

Figure V-2 is a scatter plot with the Effective SLOC as the horizontal axis. The vertical axis is the *average staffing*[28]. It is important to realize that Figure V-2 is only experimental data, not actual project data. The experiment consisted of projecting the average staffing required to complete a software development of 10K to 1000K E-SLOC. The implementation is contained in Appendix D.

The idea that the Basic COCOMO produces results with little variation is not evident in the projections of the average staff. Figure V-2 demonstrates that a very reasonable match exists between the Simplified Software Equation and Basic COCOMO, when using these productivity parameters. All six projections strongly correlate with a power equation[29].

---

[28] Computed by calculating the equivalent number of people working on the project at a given time.

[29] Basic COCOMO uses a power equation with $R^2$ of 1.0. The Simplified Software Equation uses a power equation with $R^2$=0.9875.

Figure V-3 documents the results of projecting the required *effort* to complete a software project. The experimental parameters are the same as Figure V-2, however, the vertical axis is scaled to represent the required development *effort*[30,31] in man-months.

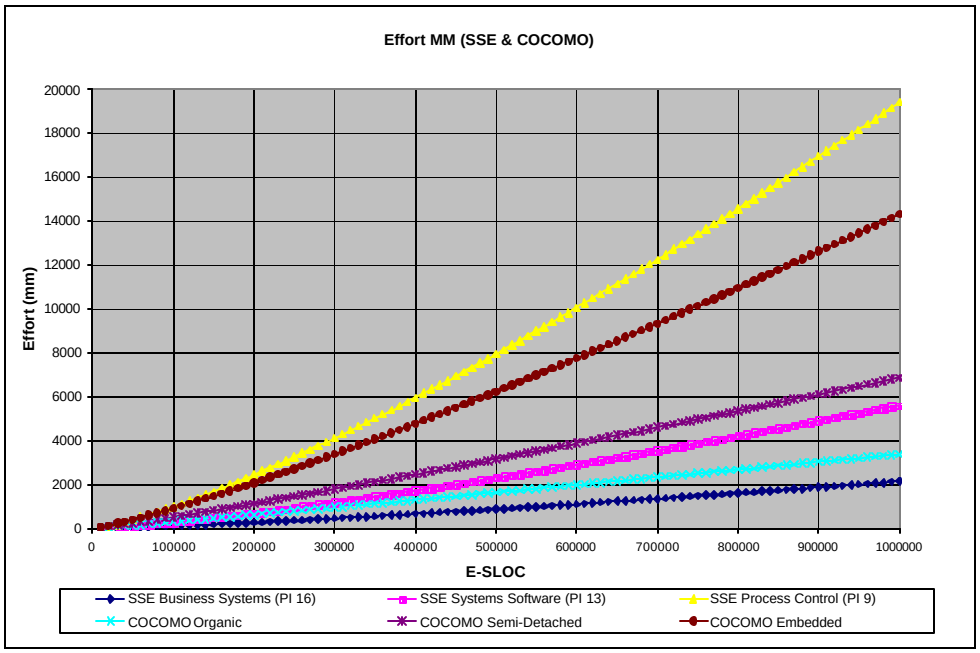



**Figure V-3. SSE and COCOMO Effort Projections.**

While studying the power equations projecting the development time, average staffing, and the minimum effort, it became very evident that these two models are predicting reasonably consistent with each other. If a technique can be developed to bound the VitéProject projections within the trend lines produced from these two models, a reasonable benchmark would exist for the research. The benchmark could be used to validate the VitéProject results.

30 (Boeh81) states the *effort* (mm) is the number of man-months estimated for the specifications and main build of the life-cycle. The Simplified Software Equation (Putn92) states *effort* (mm) is for the main build only.

31 Due to the difference in Footnote 30, the COCOMO effort projections in theory should be larger than the Simplified Software Equation effort projection.

## C. CALIBRATION OF VITÉPROJECT

The original intent of this research was to use the (Nogu00) documentation of the SRM, along with the VitéProject API, to expand the usefulness of the VitéProject. However, (Nogu00) does not accurately demonstrate the appropriate simulation parameters. Furthermore, a close simulation parameter match is documented in Appendix D; however, inconsistencies discredit the accuracy of the SRM foundation data. The research required a new approach to using the VitéProject simulation. This approach involved using the software development benchmark to tune simulation parameters.

Using the API, millions of simulation executions were conducted to try and tune the VitéProject. Figure V-4 illustrates the projection of the closest match. This instance of the simulation configuration was developed from 729,000 simulation executions. Twenty-seven hundred simulations support each vertical bar[32].

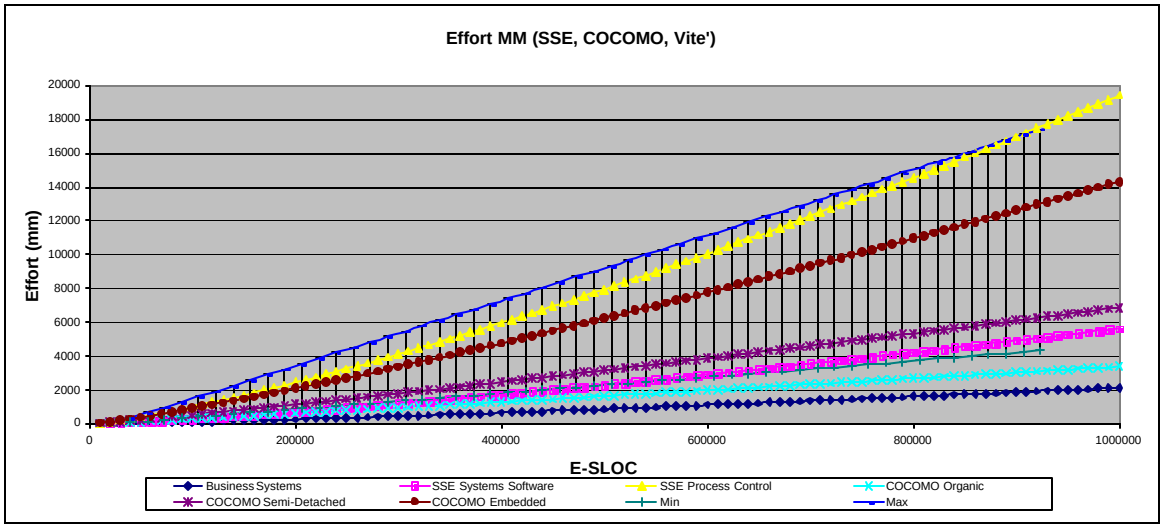



**Figure V-4. VitéProject Projections.**

[32] For illustration clarity, every 5th simulation is illustrated in Figure V-4.

Figure V-4 requires additional explanation. Figure V-4 is a reproduction of Figure V-3, the benchmark trend lines. The independent variable remains the *E-SLOC* with the dependent variable as *Effort (mm).* Each vertical bar on the graph bounds the lower and upper effort projections from the simulator. For example, when developing a 600K E-SLOC software project, a HLL[33] software development can produce the product in about 2700 man-months. The same size project would require an LHH software development of about 11,000 man-months. The implementation details are contained in Appendix D.

This is the closest the simulation can be tuned. The simulation trends generate a fairly close match to about 600K. Bounding the calibration between 10K and 600K provides projections with less error. Figure V-5 illustrates that when bounded, VitéProject reasonably projects the required effort to develop software.

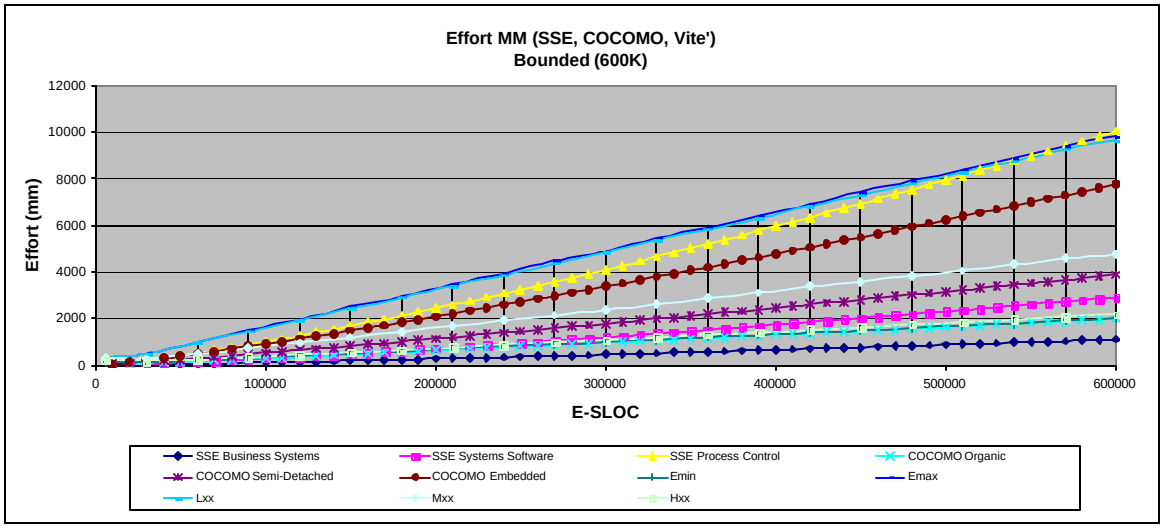



**Figure V-5. Bounded VitéProject Calibration.**

Figure V-5 only illustrates the Lxx[34], Mxx, Hxx trend lines. It is necessary to ensure that the simulator accounts for all of the potential development effort. Figure V-6

[33] HLL nomenclature is consistent with Chapter IV. Digit 1 is *efficiency*, digit 2 is *requirements*, and digit 3 is *complexity*. HLL is interpreted by the simulator to represent Efficiency $_{high}$, Requirements $_{low}$, Complexity $_{low}$.

[34] Lxx represents a low efficiency development independent of the requirements and complexity values.

illustrates all possible simulation values. For example, for any size E-SLOC, there are $27^{35}$ possible ways to configure the simulation.

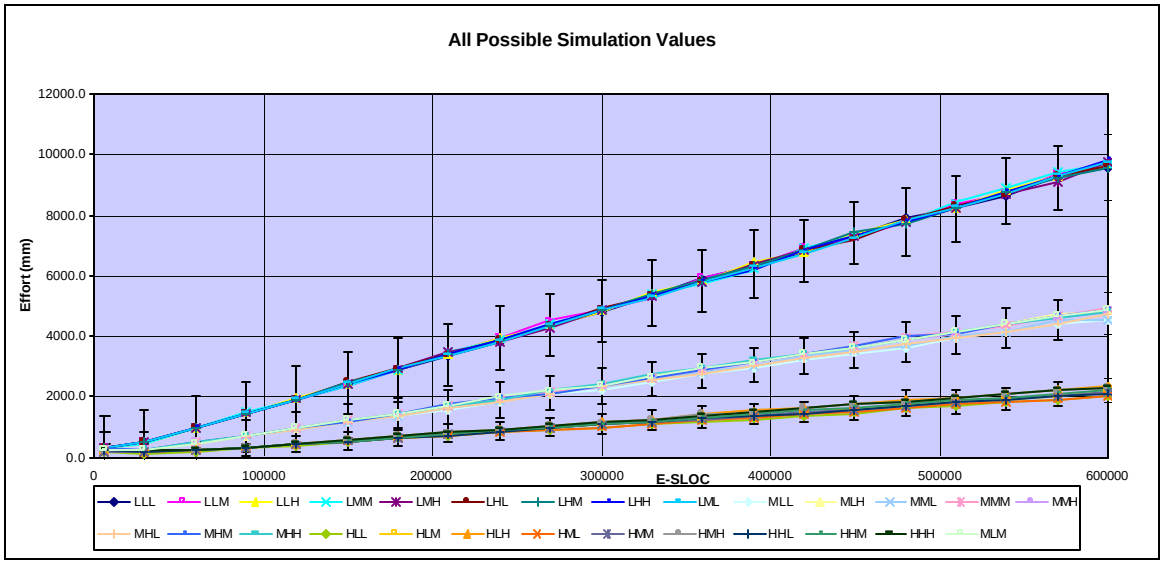



**Figure V-6. All Possible Simulation Values.**

Figure V-6 includes error bars on the trend lines. The distance of the error bars is the standard deviation for that instance of the plot. From 10K up thru approximately 180K the simulator will provide a continuous account for the require effort. However, beyond 180K, voids begin to appear in the simulation. Voids in the simulation indicate that the projections are not continuous. Essentially, a simulation of *Low efficiency* has a finite range of possible values, and the same applies for *Medium* and *High efficiency*. Problems occur because there is no overlap between the three different *efficiency* ranges.

## D. ISSUES WITH VITÉPROJECT

VitéProject has provided invaluable insights to the behavior of software development. The simulations clearly demonstrate the benefits of increasing the

[35] LLL, LLM, LLH, … HHL, HHM, HHH produces 27 different combinations.

*efficiency* of the work force. Changes in requirements also effect the development of software. VitéProject demonstrates that complexity cannot be represented by software size alone; a point discussed further in Chapter VI. However, use of VitéProject requires attention to some specific issues.

### 1. Linear Trend Lines

Figure V-7 presents another view of the simulation projections. In this instance, the 27 projections of Figure V-6 have been reduced down to nine. These nine projections account for all of the deviation in Figure V-6. For instance, the LHH projection (the highest one), represents the original LHH line offset by the standard deviation. This is repeated for all of the projections. In essence, the entire spectrum of *effort* possibilities produced by the VitéProject simulation is represented.

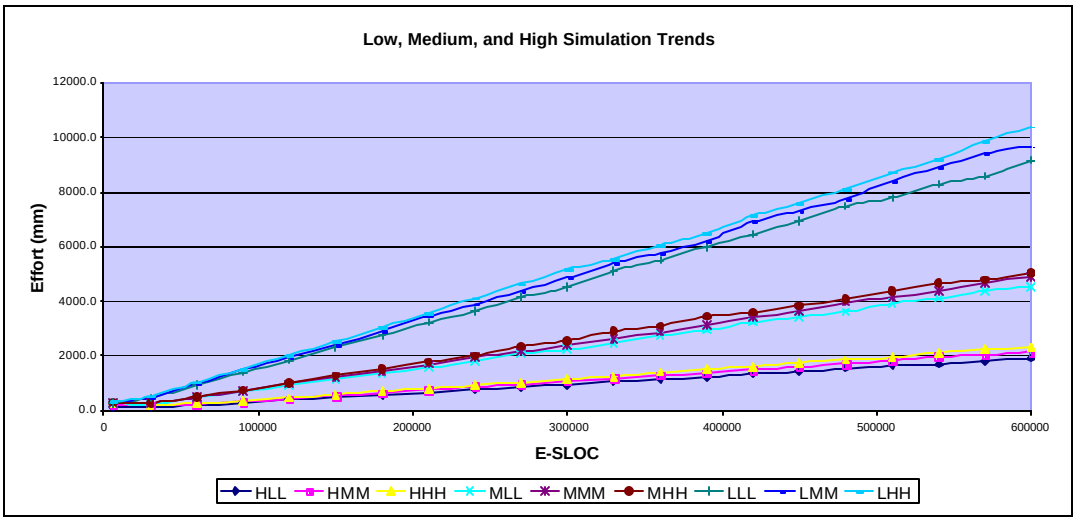



**Figure V-7. Low, Medium, and High Efficiency Projections.**

Observing the simulation projections, one soon realizes that the simulation projections appear linear. This hypothesis is tested and presented in Figure V-8.

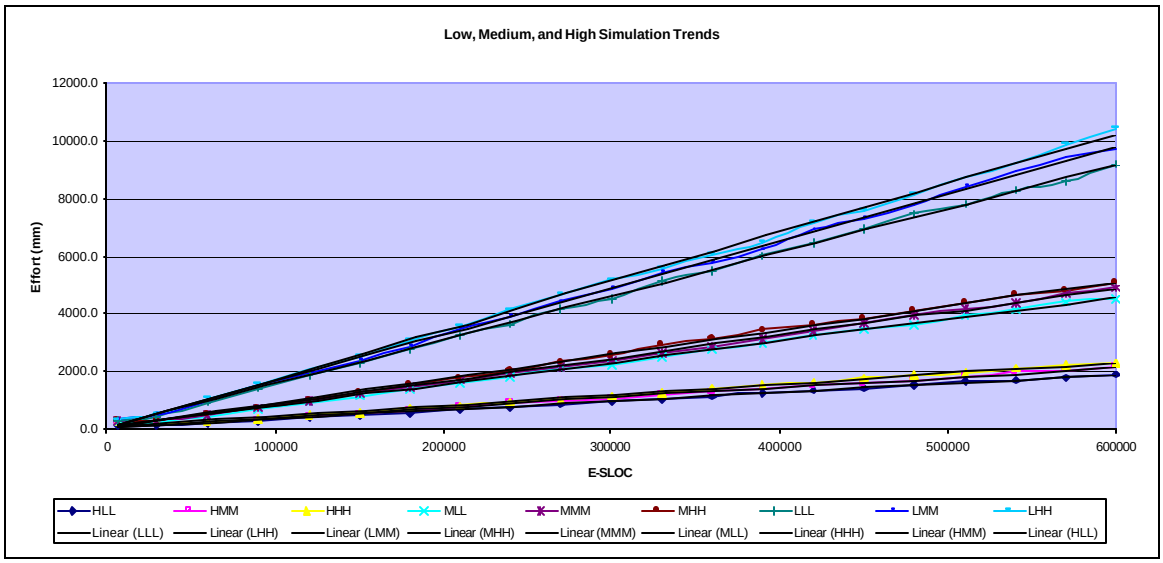



**Figure V-8. Simulation Projection Trend Lines.**

Table V-3 presents the equations and the $R^2$ values for all of the trend line data in Figure V-8. In an effort to calibrate the VitéProject simulation, the stochastic behavior in the simulator was maximized (Appendix D); thus, increasing the standard deviation of the projections. However, Table V-3 illustrates that the simulation produces development times with linear characteristics. Essentially, the simulation results can be obtained without the use of the simulator.

| Scenario | Equation | $R^2$ |
|---|---|---|
| LHH | y = 0.017x + 14.783 | $R^2$ = 0.9992 |
| LMM | y = 0.0163x + 15.033 | $R^2$ = 0.9992 |
| LLL | y = 0.0152x + 50.783 | $R^2$ = 0.9994 |
| MHH | y = 0.0084x + 55.395 | $R^2$ = 0.9981 |
| MMM | y = 0.008x + 34.175 | $R^2$ = 0.9985 |
| MLL | y = 0.0075x + 35.045 | $R^2$ = 0.9987 |
| LHH | y = 0.0037x + 58.994 | $R^2$ = 0.9948 |
| LMM | y = 0.0035x + 47.845 | $R^2$ = 0.9962 |
| LLL | y = 0.0031x + 27.477 | $R^2$ = 0.9984 |

**Table V-3. Equations and $R^2$ values from Simulations.**

## 2. Voids in Simulation

Figure V-7 and Figure V-8 clearly demonstrate an extremely small range of possible values produced by the simulator. Chapter IV, Section D, presents issues with the validation of the SRM. Figure V-9 is reproduced from Chapter IV. The validation attempts of the SRM demonstrated only two possible values can be obtained from the SRM; one for *Low* and *High efficiency*. An explanation is required for the three possible ranges of values presented in Figure V-8.

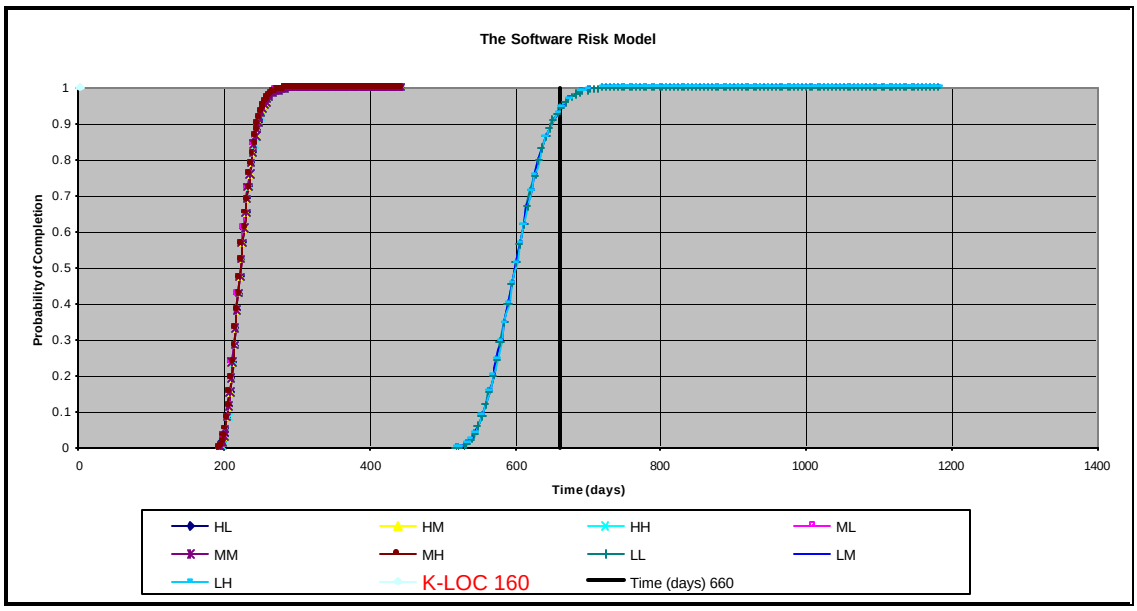



**Figure V-9. SRM Project Completion Projections.**

The VitéProject API affords the opportunity to utilize the "medium" setting in the simulator. Because of the manual techniques, the development of the SRM only used the "Low" and "High" settings in the simulator. This dissertation surmises that even if the SRM used the "Medium" setting (as in Figure V-8) in the simulator, the results would have been very similar. Thus, the third range of values result from the "Medium" setting.

It was not until the completion of the VitéProject calibration that the research revealed VitéProject does not produce continuous projections; the true source of the SRM

error. The SRM was developed by the simulation and determined valid when the SRM could reproduce the results of the simulation. The SRM faithfully projects two possible values, exactly what the simulator would have projected[36].

All of the issues surfaced with the SRM during the validation in Chapter IV, Section D, can be traced back to the VitéProject Simulation. Developing the SRM using the VitéProject and then validating the SRM against the VitéProject created a single point of failure. The simulation results do have merit. They accurately demonstrate the behavior of software development organizations; however, the development projections are scaled incorrectly to fill the voids produced in the executions.

[36] As indicated the simulation would actually projected three possible values. The development of the SRM did not consider the "Medium" parameter.

# VI. MODIFIED RISK MODEL DEVELOPMENT

The discontinuity produced by VitéProject and documented throughout Chapter V and Appendix D could potentially disrupt the development of a software risk assessment model founded on the simulation. The issues with (Nogu00, Alex01, Murr02) implementation of VitéProject are substantial enough that current resources cannot provide a mitigation. Due to these reasons, successful enhancement of the SRM cannot utilize VitéProject, with the documented parameters, as the primary source of its development and validation.

Without the use of VitéProject, the research enhanced the SRM through empirical evidence derived from actual software projects. However, there are not enough projects identified in Chapter IV and Appendix B to account for the lack of simulation data; additional software projects are necessary. The specifics of the request to QSM® for additional projects are contained in Appendix E. The calibration parameters described in Chapter V bounded the request for additional projects. QSM® provided approximately 2,000 software projects for analysis.

VitéProject, as mentioned in Chapter V, provided invaluable insights to the behavior of software. One key insight, or confirmation, is that SLOC alone cannot fully describe software complexity; also identified by (Boeh82, Putn92, Zuse97, and Dupo02). This limiting issue was recognized during calibration of VitéProject. The SRM (Nogu00) treats software complexity as a logarithmic transformation of E-SLOC; thus, having one dependent of the other. A meaningful extension of the SRM should separate the two concepts of the software complexity[37] and software size[38].

Additionally, any SRM extension designed to work with PSDL would need to develop a suitable technique to determine the inherent functional complexity of a PSDL prototype. Supporting research threads have developed such a complexity measure.

[37] Referred to as the Functional Complexity.

[38] Referred to as the Functional Size.

(Dupo02) provides the full details of this measure with necessary extracts provided within this dissertation.

This chapter concludes by delivering a formal, enhanced, risk assessment model developed from extending the SRM. The newly developed model is coined the Modified Risk Model, or MRM. The Modified Risk Model is a macro model developed to aid program managers in effectively planning the required *effort* to deliver software products. The model projects the probability of completing a software project, subject to the available resources supplied by management. Additionally, the Modified Risk Model is versatile enough to be adapted to practically any software development activity.

## A. EXPANDED PROJECT BASE

### 1. Project Selection Criteria

Figure VI-1 is reproduced from Chapter V and illustrates baseline trends of the Simplified Software and Basic COCOMO Equations. As detailed earlier, the VitéProject simulation cannot account for the total development effort of the software project. A separate model is required to provide *effort* projections and to account for the gaps in VitéProject coverage. Establishing a continuous model would allow the research to proceed with developing a mathematical representation. Essentially, the continuous model will replace the data expected from a suitable simulation.

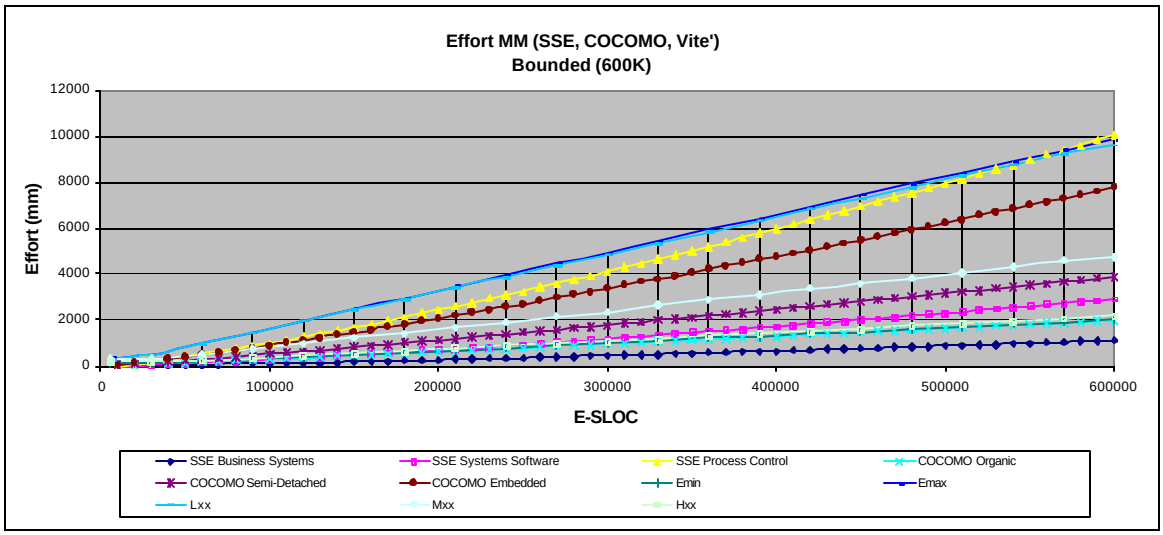



**Figure VI-1. Bounded Effort Projections: SSE & COCOMO Equations.**

The SRM requires the *efficiency*, *requirement volatility*, and *complexity* to complete a projection. Chapter IV demonstrates a mapping of these measures to the QSM® database. The simulations provided no evidence to dispute the viability of using *efficiency* and *requirement volatility* to help model software development risks. The enhancement to the SRM will still contain these parameters.

However, developing the VitéProject calibration demonstrated that an accurate simulation projection must consider separating the SRM notion of complexity into two independent measures; the dependencies of the three SRM parameters should be independent of the functional size (E-SLOC) of the software. For example, consider a low efficiency organization, developing a stable system of minimum or routine complexity, developing a 100K E-SLOC software system. A high efficiency organization developing an unstable complex system of 100K E-SLOC could potentially require more effort than the low efficiency organization. The SRM, or the VitéProject, could never project more required effort for the high efficiency organization.

If the simulation behavior is truly representative of software development, then research should be able to extend the SRM on the premise that four parameters can provide the essential information required to successfully project the software

development risks: *efficiency*, *requirement volatility*, *functional complexity*[39], and *functional size*[40]. Two of the parameters, *efficiency* and *requirement volatility*, remain identical to those of (Nogu00) and the other two decompose the original notion of complexity. Additionally, the extension to the SRM, the MRM, is based upon projecting the required development effort[41].

Bounding these four measures to reflect the baseline projections of Figure VI-1, maximized the chance of accounting for the discontinuity and successful modeling of a continuous representation. Lawrence Putnam, Sr. of QSM®, was consulted on this approach:

> …What I need: I need all the projects that I can get information on in the PI range (8 - 16) for all available application types. I'm particularly concerned with 10K - 600K SLOC, but outside of this range will be good for testing outliers.
>
> What I'm going to experiment with: I am going to fix the PI and observe the differences in E for the different application types at fixed SLOC. Then for the same set of data, I am going to fix the application type and observe the differences in E for the different PI's at fixed SLOC. I'm doing this to document the sensitivity and impact that application type and PI have on the required E. Ultimately, this will help me prove or disprove results that I am receiving from the simulator. …

QSM® graciously responded with developmental data on about 2000 software projects with the following guidance:

> …Have you considered this: With the selection you propose you will find a spread in efforts at a fixed PI and a fixed size which will be the schedule-effort trade off? I have done bunches of those searches trying to prove the 4th power trade off relationship. It is not easy to do because the matches on size are few and far between. **You will have to pick ranges.** Same with PIs. …

---

[39] The *functional complexity* is on a ratio scale tested from 0 to 5. The minimum value calibrated is zero (routine complexity). The maximum value calibrated is five (real time development).

[40] *Functional size* can be considered any measure that can be backfired to E-SLOC. As with the SRM, E-SLOC is a fundamental parameter in the model performance.

[41] This is a fundamental difference between the two models SRM and MRM. The SRM projected the minimum number of ***days*** to develop a product. The MRM projects the minimum ***effort*** in person/man months required to develop a software product.

## 2. Project Subset

Figure VI-2 is an overview of the subset of 2000 projects[42] received from QSM®. Appendix E provides complete details of the project subset. This overview details the different application types represented in the project subset. Additionally, it is easy to observe the total number of software applications produced by the different application types. For example, there exist only one *microcode* application and over 1400 *business systems.*

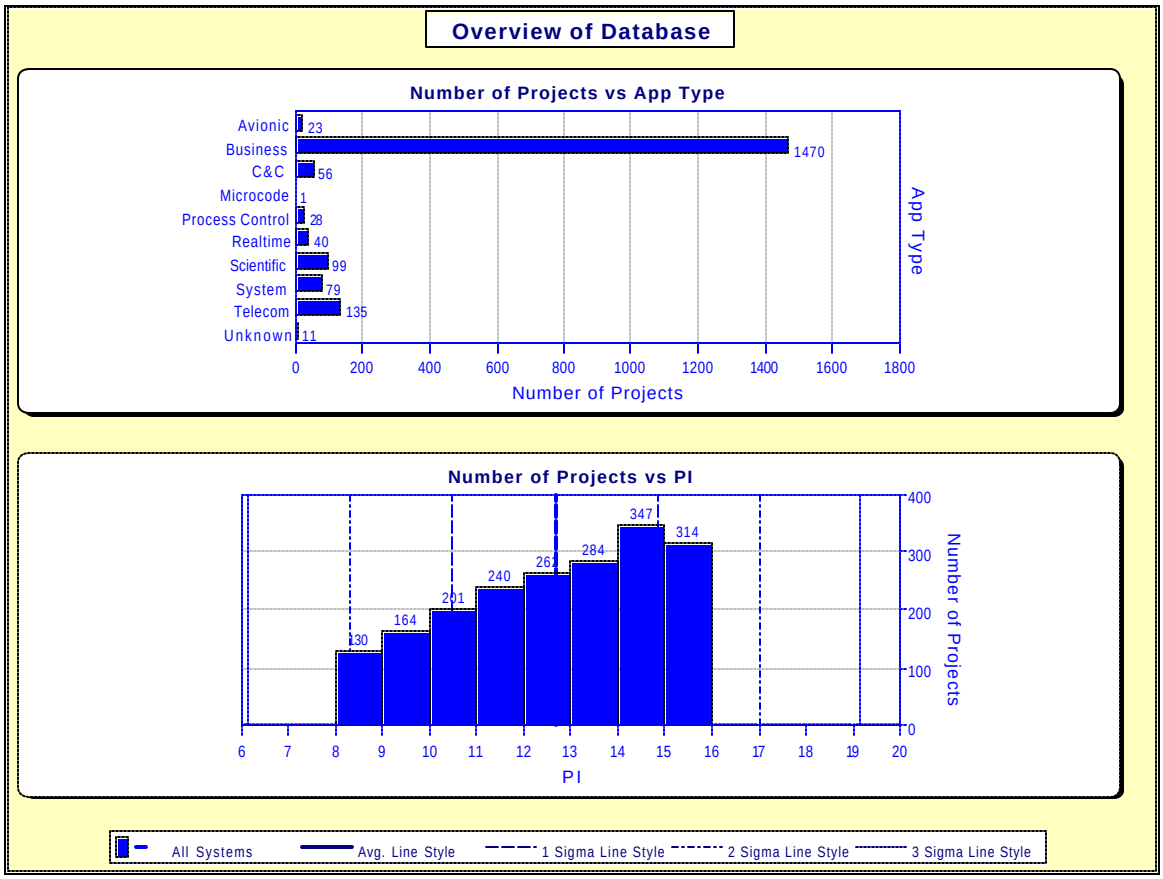



**Figure VI-2. Overview of 2,000 Projects**

[42] The actual number of projects was 1942. This dissertation refers to the subset of 2000 for clarity.

The base trend lines, evident in Figure VI-1, used *productivity index (PI)* values of 9, 13, and 16. To project within these limitations, software development data is requested from organizations whose *productivity index* ranged from 8 – 16 as indicated in **Error! Reference source not found.**. **Error! Reference source not found.** also provides the specific number of projects for each of the development organizations. The average PI is 12.65 with a 2.17 standard deviation.

Now the task is to segment the data as suggested by Lawrence Putnam, Sr. VitéProject confirmed intuition that if sensitivity analysis is conducted on each variable in isolation, the following should be demonstrated:

- Higher efficiency organizations could produce software projects with less *effort* than low efficiency organizations
- A software project experiencing volatile requirements takes more effort than a stable development
- An increase in *functional complexity* increases the required development effort

The hypothesis is that if sensitivity analysis is conducted on a large enough set of real projects, similar behavior would be evident.

### 3. Project Isolation

A suitable technique is developed to segment the project subset. Four parameters need to be examined; *efficiency*, *requirements volatility*, *functional complexity*, and *functional size*. Due to state space explosion, it becomes imperative, to examine the most sensitive of the four parameters. The data is segmented according to the effects of *efficiency*, *functional complexity*, and *functional size*. The strategy is as follows:

- *Efficiency*. Segment the project subset into four efficiency groupings. The groupings are determined by the combination of projects of different

*productivity index*. For example, the project subset contains a total of eight indexes (**Error! Reference source not found.**). The research segments the subset of projects into four efficiency groups {(8/9), (10/11), (12/13), (14/15)}. Efficiency group (8/9) is the lowest efficiency of the four groups with (14/15) representing the organization with the most efficiency.

- *Functional Complexity*. The project subset is comprised of ten different application types. (Putn92) indicates that these application types have an inherent complexity associated with each. Ten application types are segmented into four groups, each decreasing in complexity:
  - Type A: {Microcode, Avionic, and Real Time Systems}
  - Type B: {Command & Control and Process Control Systems}
  - Type C: {Telecommunications, Systems Software, and Scientific Systems}
  - Type D: {Business and Miscellaneous Systems}
- *Functional Size*. The basic unit of work is E-SLOC.

### 4. Requirements Volatility

The strategy to segment the project subset kept the state space at a manageable level. Not considering the effects of changing requirements on the project subset has proven to be negligible. Figure VI-3 demonstrates approximately a forty percent reduction in the project subset if a consideration is extended to the requirement volatility[43]. Maintaining the maximum data points for the analysis is paramount to the research.

[43] Only 1177 projects in the subset contained values in the Requirements Growth Percentage field out of 1942 total projects.

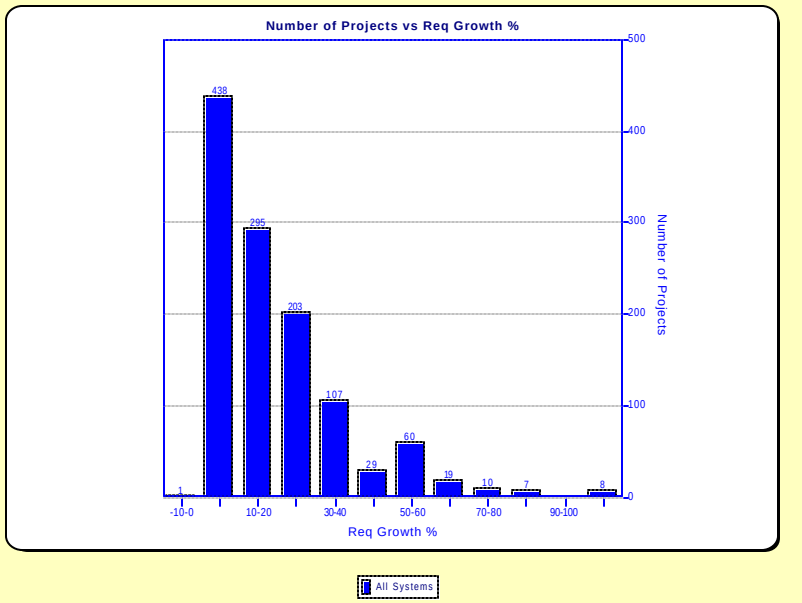



**Figure VI-3. Requirements Growth Percentage.**

As indicated above, the analysis considers four segments of the *efficiency* and four segments of the *functional complexity* for a total of $4^2 = 16$ possible scenarios. Resources simply prevent sampling the requirements in four segments as well $4^3 = 64$. Additionally, the SRM validation indicates that the requirements volatility is the least sensitive parameter in the model. Accounting for the requirements volatility is preserved in the MRM and Chapter VII demonstrates the effects.

## 5. Efficiency

The QSM® database provides ample information to conduct analysis on the *efficiency* and *complexity.* Chapter IV and Appendix C establish that *efficiency* relates to the productivity parameter. Figure VI-4 demonstrates the effects of segmenting the project subset into four efficiency groups. By fixing the *efficiency*, the effects of altering the *functional complexity* of a system can be studied. If so, Figure VI-4 should clearly indicate that with a constant efficiency, more complex system developments will require more development effort.

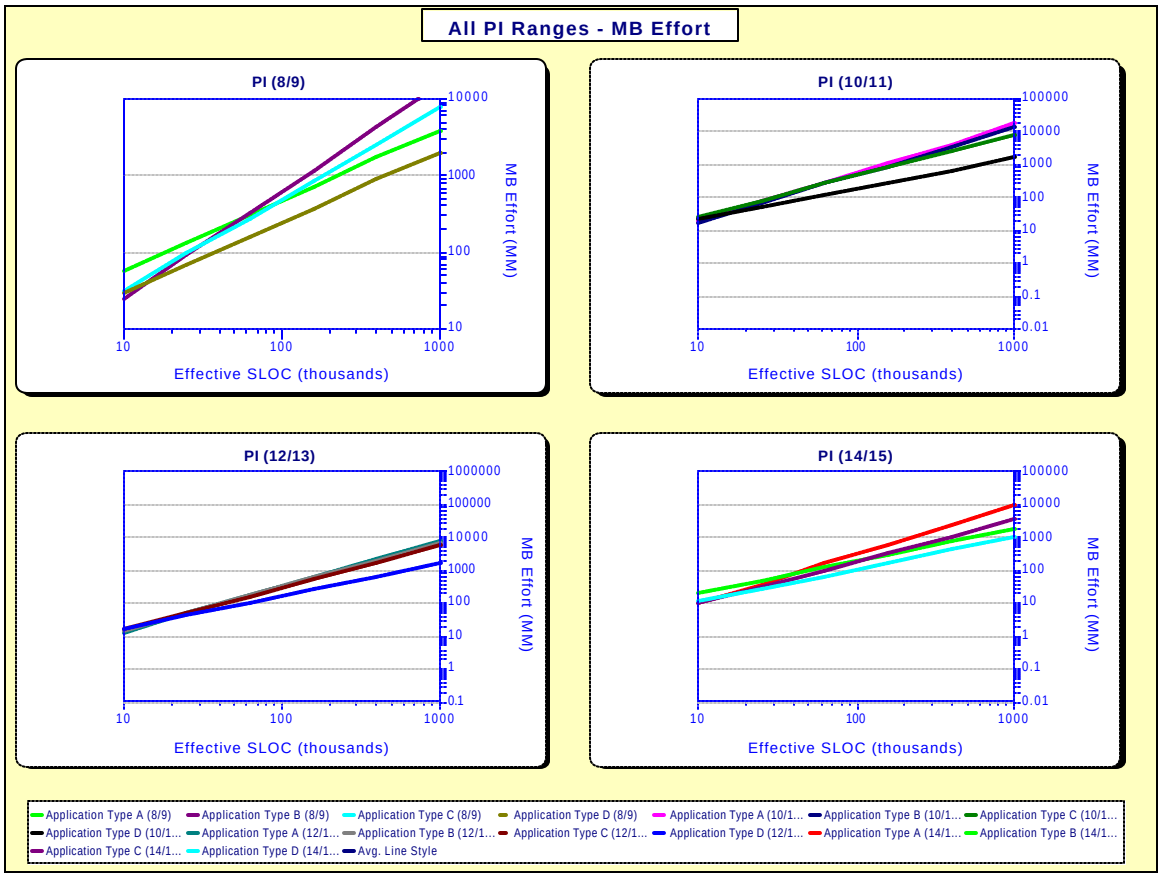



**Figure VI-4. Project Segregation on Productivity Index.**

Figure VI-4 represents the four efficiency groupings. Additionally, each group contains the four trends of *functional complexity*[44]. Each graph is presented on a double-logarithmic scale. The size of the application (E-SLOC) serves as the independent axis. The dependent axis is the required *effort* (man-months) for the main build. Appendix E contains the implementation details.

- PI fixed at 8/9. The top-left graph represents a relatively low-efficiency organization. The four trend lines demonstrate the *effort* required and decreases from application Type B, Type C, Type A, and Type D

[44] Figure VI-4 and Figure VI-5 use a power equation to represent the actual data plots. Appendix E contains the actual data plot details.

respectively. The hypothesis indicated that Type A should require the most *effort*. However, in this case, Type A demonstrates outlier behavior; the trend Type A crosses the other projection lines.

- PI fixed at 10/11. The top-right graph represents a software organization performing with slightly more efficiency. The four trend lines provide support for the hypothesis. Application Type A, Type B, Type C, and Type D represent a decreasing required *effort*, as expected.
- PI fixed at 12/13. The bottom-left graph represents an efficient organization. The four trend lines again provide support for the hypothesis: Type A, Type B, Type C, and Type D represents a decreasing required *effort*, as expected.
- PI fixed at 14/15. The remaining graph on the bottom-right represents organizations with the most efficiency in the project subset. The complexity trend lines follow the pattern Type A, Type C, Type B, and Type D; indicating the Type C software projects perform less efficiently than the Type B projects. As with the chart PI (8/9), the significance of the trend lines crossing is an indication of the extent of the outlier behavior.

The trend line data clearly demonstrate that an increase in software system *complexity* requires an increase in development *effort*. The empirical evidence presented in Figure VI-4 provides a critical foundation to the development of the MRM.

### 6. Functional Complexity

Using the same technique demonstrated with the *efficiency*, the project subset is segmented according to the *complexity* of the application type. Figure VI-5 illustrates another view of the sixteen trend lines projected in Figure VI-4. This view provides support for the following hypothesis; increasing the *efficiency* of an organization decreases the required *effort* to develop software projects. If this holds, then Figure VI-5 should demonstrate a reduction in the required *effort* for the higher efficiency software development organizations.

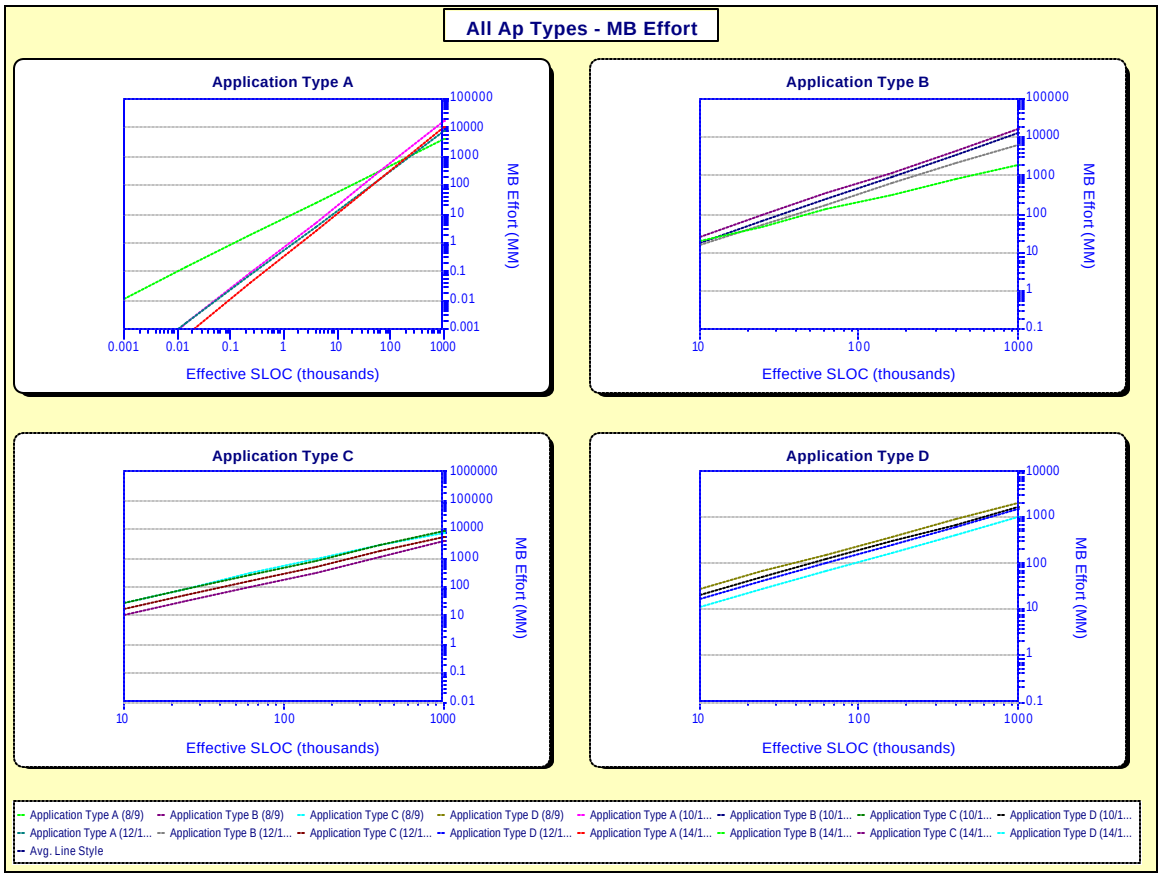



**Figure VI-5. Project Segregation on Functional Complexity.**

Figure VI-5 represents the four complexity groupings. Additionally, each group contains the four trends of *efficiency*. Consistent with Figure VI-4, each graph is illustrated on a double-logarithmic scale. The size of the application (E-SLOC) serves as the independent axis. The dependent axis is the required *effort* (man-months) for the main build. Appendix E contains the implementation details.

- Application Type A (Microcode, Avionic, and Real Time Systems). The top-left graph represents the most complex type of software development. Up through 100K E-SLOC, the four trend lines demonstrate the *effort* required decreases as the *efficiency* of the organization increases: PI (8/9), PI (10/11), PI (12/13), and PI (14/15) respectively. The hypothesis indicated that a low efficiency organization should require the most *effort*; this is evident in the illustration. As with Figure VI-4, trend line Type A (8/9) behaves as an outlier.

- Application Type B (Command & Control and Process Control Systems). The top-right graph represents a software organization developing a slightly less complex software system. The Type B application is further removed from the *real time* development. The four trend lines again provide support for the hypothesis; PI (8/9), PI (10/11), PI (12/13), and PI (14/15) respectively.
- Application Type C (Telecommunications, Systems Software, and Scientific Systems). The bottom-left graph represents relatively routine software development. A slight outlier is contained in this graph. The trend line progression is PI (10/11), PI (8/9), PI (12/13), and PI (14/15) respectively. As indicated in the illustration, the transposing of PI (10/11) and PI (8/9) is negligible.
- Application Type D (Business and Miscellaneous Systems). The remaining graph on the bottom-right represents software applications that are the most routine and least complex to develop. The *efficiency* trend lines follow the hypothesis pattern PI (8/9), PI (10/11), PI (12/13), and PI (14/15) respectively.

As with the trend line data from Figure VI-4, fixing the *functional complexity* clearly demonstrates that an increase in organizational *efficiency* reduced the development effort. The empirical evidence presented in Figure VI-5 provides another critical foundation to the development of the MRM.

### 7. Results of Trend Analysis

With the completion of the sensitivity analysis on the *efficiency*, *functional complexity*, and *functional size*, the trend lines can be consolidated to help formulate the model. It is necessary to verify if the consolidation would reveal the same effects that flawed the VitéProject and SRM. Recall the beginning of this chapter explains that the VitéProject and the SRM cannot, under any circumstance, project a low efficiency organization with less *effort* than a high efficiency organization[45]. For the model to avoid this pitfall, it should be able to demonstrate an overlapping in the trend lines.

[45] This is assuming both artifacts have the same E-SLOC or the same activity durations in VitéProject.

Figure VI-6 illustrates the results of merging all sixteen trends into a single model. The independent axis contains the E-SLOC on a bounded scale from zero to 600,000. The dependent scale represents the effort in man-months. The trend lines provide continuous representation of the required effort to develop software.

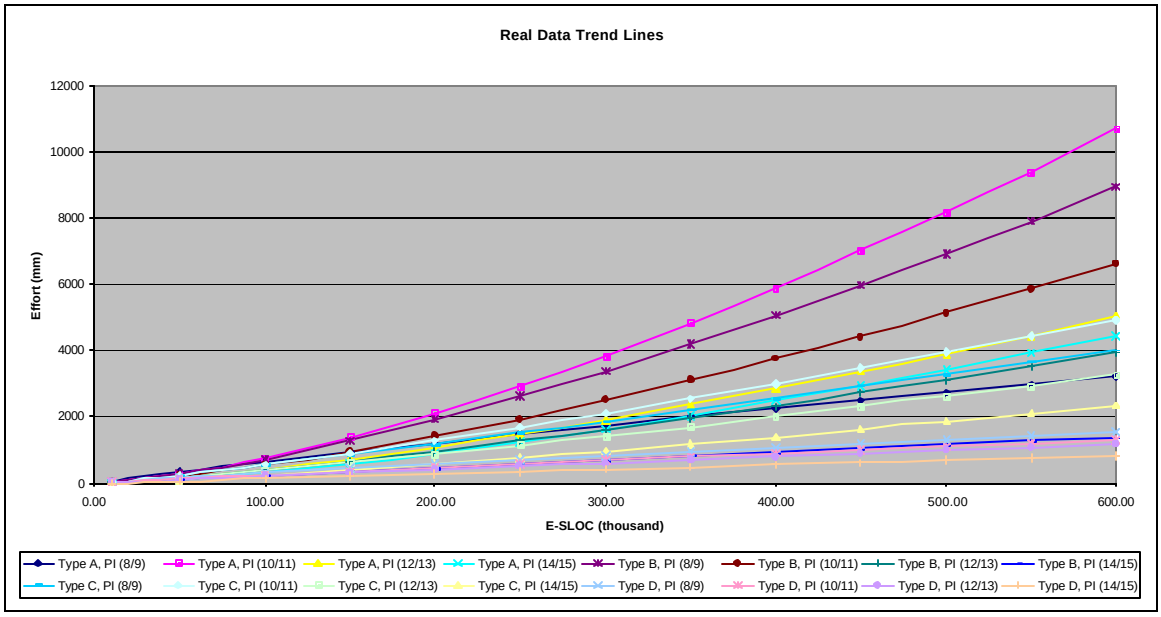



**Figure VI-6. Trend Line Data from Real Projects.**

Table VI-1 provides a clear legend for Figure VI-6 and Appendix E provides additional information. The table orders the scenarios based upon the maximum effort the trend required. Also provided is the $R^2$ value. The trend line data clearly demonstrates that an overlap exists in the projected effort; avoiding the discontinuity problems. The remainder of the table demonstrates how *functional complexity* and organizational *efficiency* impact software development.

| Effort Required[46] | Trend Line Scenario | $R^2$ |
|---|---|---|
| 16 | Type A (10/11) | 0.8167 |
| 15 | Type B (8/9) | 0.8767 |
| 14 | Type B (10/11) | 0.8918 |
| 13 | Type A (12/13) | 0.9946 |
| 12 | Type C (10/11) | 0.9039 |
| 11 | Type A (14/15) | 0.9115 |
| 10 | Type C (8/9) | 0.9167 |
| 9 | Type B (12/13) | 0.7959 |
| 8 | Type C (12/13) | 0.8511 |
| 7 | Type A (8/9) | 0.8311 |
| 6 | Type C (14/15) | 0.8697 |
| 5 | Type D (8/9) | 0.8978 |
| 4 | Type B (14/15) | 0.8102 |
| 3 | Type D (10/11) | 0.7426 |
| 2 | Type D (12/13) | 0.7777 |
| 1 | Type D (14/15) | 0.7668 |

**Table VI-1. Ordered Projection of Trend Line Data.**

The trend line scenarios are consistent with the individual projections in Figure VI-4 and Figure VI-5. Typically, a less complex system (Type D) requires the least effort regardless of the *efficiency* of the developing organization. However, among the Type D's the more efficient organization produces with less *effort*.

The largest outlier, Type A (8/9), is the most inconsistent in the representation and required additional research. Appendix E documents that trend line Type A (8/9) is comprised of 27 software projects: six avionic, one microcode, and 20 real time. The one microcode project distorts the trend line data (Appendix E).

[46] Effort required ranges from high to low. The trend line Type A (10/11), required the most *effort* of all of the trends at 600K E-SLOC.

## B. COMPLEXITY IN THE MODIFIED RISK MODEL

VitéProject demonstrated that E-SLOC can not alone account for the software complexity. Several software analysts also provide support (Dupo02, Zuse97). During the validation of the SRM, evidence mounted to suggest that any enhancement to the risk assessment model would require an improved method to determine complexity of PSDL. Additionally, modifying the SRM to work with "real" project data created a desire to develop an abstract model to easily apply to multiple development domains. The intent was to design the enhancement with the ability to interface with any calibrated complexity measure.

The VitéProject demonstrated that the *functional size* is independent of the *functional complexity.* So, it becomes impractical to try and represent the software complexity in a single "all encompassing" measure. A full complexity description of the software must include the *functional size* and the *functional complexity.* Since the SRM derived its complexity from PSDL in the form of Large Granular Complexity, a new complexity measure is required to extend the previous LGC implementation.

### 1. The Dupont Scale

(Dupo02) developed a complexity measure specifically adapted for PSDL. This complexity measure provides a suitable interface to the MRM. The Dupont Scale calculates the complexity of PSDL using a hybrid complexity measure that properly accounts for data flow and the properties associated with each operator and data stream. The hypothesis is that the more data that is generated and flows between operators, the higher the complexity. Moreover, each property represents a different level of complexity. Operators and data streams become more complex as more properties are associated with them. Minimizing data flow and associated properties, increases the understandability of the prototype; hence, reduces the complexity.

(Dupo02) derives a ranking of the properties using a set of weights. Each operator and data stream is assessed a total weight based on the sum of its weighted properties. This weight is added to one, to represent each operator and data stream as something greater than itself.

The technique of ranking the process with a set of weights is combined with the theory of information flow which takes the product of the total number of *fan_in* and *fan_out* data streams as a function of each module (i.e. operator), representing the total possible number of combinations of *fan_in* data streams to *fan_out* data streams for the module. This product is multiplied by the weighted value of its functional operator, providing a complexity for that operator. Finally, the total system complexity is calculated as the sum of operator complexities. The following demonstrates a partial calculation of the *functional complexity* using the Dupont Scale (Dupo02):

$$DS = \sum_{i=1}^{n} o_i \left[ dsi\left(o_i\right) * dso\left(o_i\right) \right]$$

where:

- DS is complexity of PSDL under the Dupont Scale
- $o_i$ is each individual operator
- dsi is data streams in of operator oi and dsi = max(data_stream_in, 1)
- dso is data steams out of operator oi and dso = max(data_stream_out, 1)
- n is the total number of operators

### 2. Software Volume

Any measure of the software size or functionality can be utilized for the software volume; as long as this value can be backfired[47] to E-SLOC, ((Putn92) uses a term called *gearing factor*). The MRM is calibrated for E-SLOC. (Putn92) defines E-SLOC as a measure of the size or functionality of a software system. When determining the E-

[47] The process of converting a sizing measure into an equivalent measure of E-SLOC.

SLOC, count executable source lines deliverable to customer/user, which will exclude environmental or scaffolding code. The count may include an estimate of equivalent new lines in reused or modified modules; also known as source statements.

The concept of the (Putn92) gearing factor is important and expands the use of the model to other software domains. Essentially, the gearing factor is a constant multiplied by any specified sizing metric, to derive an equivalent value in E-SLOC. (QSM® Data Manager) defines the use of a *gearing factor*:

> The gearing factor is the average number of basic units of work in your chosen Function Unit (Basic Work Units/Function Units). Originally, it was designed to be used as a common reference point for comparing different sizing metrics by mapping them to the smallest sizing unit common to all software projects: lines of code.
>
> In today's GUI environments, there are other basic function units that are equivalent to a line of code (and may be more meaningful in visual environments). Some examples might be setting a property, constructing a simple macro element, updating a value in a table, etc. …
>
> …If sizing in lines of code (or any basic unit of work), the gearing factor will be 1. If you choose a function unit other than lines of code (or any equivalent size unit), estimate the average number of basic work units contained in each function unit. …
>
> …The gearing factor is best determined by running an automated code counter on the finished product and dividing the LOC count by the number of function units in the final product (LOC/Function Units).

All of the MRM calibrations utilize E-SLOC as the base volume measure. Chapter IV demonstrated that (Nogu00) provides a conversion equation to derive the equivalent E-SLOC from LGC; ($KLOC = 40LGC/1000$). The MRM recommends utilizing the LGC conversion equation to estimate the volume of the software size. However, the validation in Chapter IV also indicated that this equation could be too conservative. In the absence of additional data points to establish the *gearing factor* for PSDL, this research supports the original equation presented in the development of the SRM.

## C. THE MODIFIED RISK MODEL DEVELOPMENT

Sufficient information is available to derive a mathematical representation of the behavior demonstrated in Figure VI-6. This dissertation demonstrates the model would require four primary inputs: *efficiency*, *requirement volatility*, *functional complexity*, and *functional size*. The SRM requires three primary input parameters and utilizes the three-variable form of the Weibull Cumulative Distribution Function to project the probability of project completion, equation (6.1).

$$\text{cdf}: F(x;\gamma,\alpha,\beta) = \begin{cases} 0, & x < \gamma \\ 1 - \exp(-((x-\gamma)/\beta)^{\alpha}), & x \geq \gamma \end{cases} \tag{6.1}$$

where

$\alpha$ = shape parameter of the pdf
$\beta$ = scale parameter of the pdf
$\gamma$ = location parameter of the pdf
$x$ = the random variable under study

The MRM is intended to utilize four primary input parameters; however, does the three-variable Weibull equation continue to provide the best distribution for the MRM? A series of experiments are conducted to determine the most appropriate distribution for extending the risk assessment model. Using ReliaSoft's Weibull ++ 5.0 32 Bit (Pro), the most probable distribution is computed for the sixteen trend lines in Figure VI-6. The Weibull ++ software conducts analysis comparing six different distributions: exponential 1&2 variable, Weibull 2&3 variable, normal, and lognormal. Appendix E contains the details but Table VI-2 provides an overview of the derived information.

| Scenario | Effort Required | Best Fit | Alpha | Beta | Gamma |
|---|---|---|---|---|---|
| Type A, PI (10/11) | 16 | Weibull 3 | 1.1600 | 4601.9700 | -2.5900 |
| Type B, PI (8/9) | 15 | Weibull 3 | 1.2500 | 4065.5300 | -26.6600 |
| Type B, PI (10/11) | 14 | Weibull 3 | 1.2500 | 3010.8900 | -19.1600 |
| Type A, PI (12/13) | 13 | Weibull 3 | 1.2500 | 2302.5700 | -15.9400 |
| | | | | | |
| **Type C, PI (10/11)** | **12** | **Weibull 2[48]** | **1.0300** | **2244.7700** | **33.4200** |
| Type A, PI (14/15) | 11 | Weibull 3 | 1.1800 | 1950.0400 | -3.4500 |
| Type C, PI (8/9) | 10 | Weibull 3 | 1.8000 | 2367.8000 | -180.5000 |
| Type B, PI (12/13) | 9 | Weibull 3 | 1.4100 | 1951.8400 | -42.8900 |
| | | | | | |
| Type C, PI (12/13) | 8 | Weibull 3 | 1.5800 | 1752.9000 | -78.6500 |
| Type A, PI (8/9) | 7 | Weibull 3 | 2.9000 | 2639.9900 | -623.3800 |
| **Type C, PI (14/15)** | **6** | **Weibull 2[49]** | **0.9900** | **1031.5300** | **12.6600** |
| Type D, PI (8/9) | 5 | Weibull 3 | 2.8600 | 1246.6300 | -287.4100 |
| | | | | | |
| Type B, PI (14/15) | 4 | Weibull 3 | 2.2700 | 961.6700 | -142.2900 |
| Type D, PI (10/11) | 3 | Weibull 3 | 2.6300 | 1002.8000 | -200.0700 |
| Type D, PI (12/13) | 2 | Weibull 3 | 2.3900 | 844.5000 | -140.1000 |
| Type D, PI (14/15) | 1 | Weibull 3 | 2.5300 | 604.6300 | -112.1300 |

**Table VI-2. Alpha, Beta, & Gamma Values for Trend Data.**

Following the conventions of Table VI-1, the trend lines are ordered by the required effort (most to least). For every trend line except two, the three-variable Weibull function provides the best fit to the distribution. Included in Table VI-2 are the corresponding values for the *alpha*, *beta*, and *gamma*. The next logical step is to derive an interface between the real-world inputs and the derived ranges of the *alpha*, *beta*, and *gamma* variables.

### 1. Alpha

Alpha is the shape parameter. The skew of the function is altered thru changes in this value. When *alpha* = 1, the Weibull distribution reduces to the exponential

[48] Three-variable Weibull was the third closest match.

[49] Three-variable Weibull was the fifth closest match

distribution with scale parameter *beta.* The special case, alpha = 2, is called the Rayleigh Distribution with scale parameter *beta,* named after William Strutt, Lord Rayleigh.

Table VI-2 indicates the range for the *alpha* variables is between 0.99 and 2.90. Recall from Chapter IV that EF is derived by

$$EF = Direct\%/Idle\% \tag{6.2}$$

The universe of possible *efficiency* values is demonstrated in Table VI-3. The subset used for calibrating the mathematical model is 45% <= Direct Time <= 99%. The final step was to derive a curve fit between the two ranges.

| Direct Time % | Idle Time % | EF |
|---|---|---|
| 0.00% | 100.00% | 0.00 |
| 5.00% | 95.00% | 0.05 |
| 10.00% | 90.00% | 0.11 |
| 15.00% | 85.00% | 0.18 |
| 20.00% | 80.00% | 0.25 |
| 25.00% | 75.00% | 0.33 |
| 30.00% | 70.00% | 0.43 |
| 35.00% | 65.00% | 0.54 |
| 40.00% | 60.00% | 0.67 |
| **45.00%** | **55.00%** | **0.82** |
| **50.00%** | **50.00%** | **1.00** |
| **55.00%** | **45.00%** | **1.22** |
| **60.00%** | **40.00%** | **1.50** |
| **65.00%** | **35.00%** | **1.86** |
| **70.00%** | **30.00%** | **2.33** |
| **75.00%** | **25.00%** | **3.00** |
| **80.00%** | **20.00%** | **4.00** |
| **85.00%** | **15.00%** | **5.67** |
| **90.00%** | **10.00%** | **9.00** |
| **95.00%** | **5.00%** | **19.00** |
| 100.00% | 0.00% | #DIV/0! |

**Table VI-3. EF Ranges.**

(SPSS99) was utilized to derive Equation (6.3). The inputs values supplied to the curve fit regression were derived from the *efficiency* (Table VI-3) and the *alpha* variable from Table VI-2.

$$a = 1.0968 * EF^{0.3970} \tag{6.3}$$

where

EF = the organizational efficiency

**2. Beta**

The beta is the scale parameter responsible for stretching or compressing the graph or the horizontal axis. The effect of a scale parameter greater than one is to stretch the probability distribution; the greater the scale value, the greater the stretching. The effect of a scale parameter less than one is to compress the probability distribution. The compression approaches a spike as the scale parameter goes to zero. A scale parameter of one leaves the probability distribution unchanged.

Both the *requirements volatility* (RV) and the *functional complexity* (FC) have a combined effect on the scale parameter. Chapter IV details how to calculate the *requirements volatility* and is reproduced in Equation (6.4) for clarity.

$$RV = BR\% + DR\% \tag{6.4}$$

The intent of this dissertation is to develop a mathematical model that is generic enough to have applicability in any software development domain. Deriving the *efficiency* of an organization and determining the *requirements volatility* on a software project can be determined on any software project. Interfacing with a *complexity* measure must ensure abstraction remains.

Working in parallel, (Dupo02) agreed to an interface between the Dupont Scale for PSDL and the MRM resulting in a bounded complexity representation. Bounding the complexity provides a baseline for expansion to additional complexity measures. This

concept is similar to the backfiring concept for the software size. The *functional complexity* is bound on a real scale as 0.0 <= FC <= 5.0, where zero represents a software development of low complexity and five represents a real-time software development. Figure VI-7 illustrates the validated[50] complexity ranges represented in the MRM.

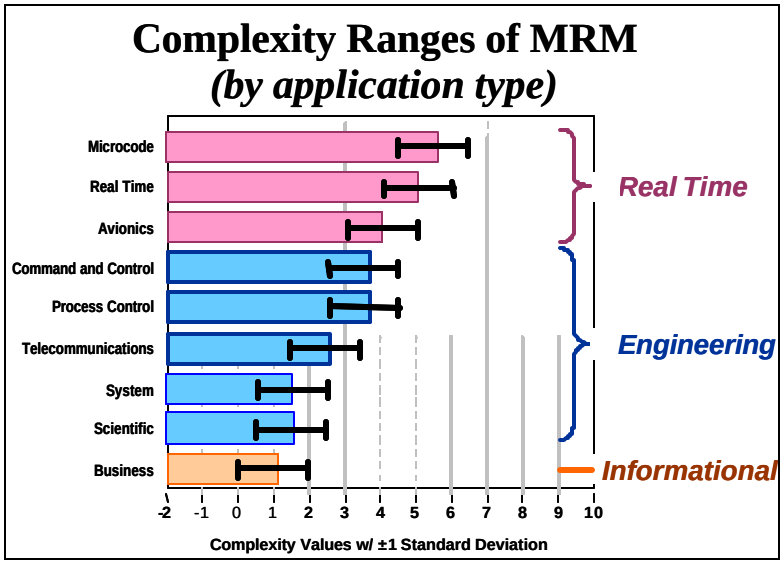



**Figure VI-7. Functional Complexity Ranges of the MRM.**

(SPSS99) is utilized again to compute the equations for the *beta*. Recall that Table VI-2 did not utilize any input from the *requirements volatility* in the real project subset. For this reason, the impact of changing requirements was included as the least sensitive parameter in the model; however, Chapter VII validates against changing requirements. Equation (6.5) demonstrates how to determine the required *beta* for the MRM.

$$b = g * \exp(FC * 0.1202)^{\left(1+\frac{RV}{10}\right)} \quad (6.5)$$

where

[50] Chapter VII presents the MRM validation.

FC is the functional complexity
RV is the requirements volatility

### 3. Gamma

The three-variable Weibull distribution provides for a location parameter; referred to as *gamma* in this dissertation. The location parameter is utilized to shift the distribution to a different starting point. The absence of the location parameter produces distribution curves originating at zero.

(Dupo02) explains that the SRM concept of LGC is actually a sizing measure. From Chapter IV, (Nogu00) recommends calculating the E-SLOC of the software project thru Equation (6.6).

$$KLOC = (40\text{LGC} + 150)/1000 \tag{6.6}$$

where

LGC = Operator + Data Streams + Edges

The MRM implements a slight variation of Equation (6.6) that is suitable to provide the *functional size* when determining the size of a PSDL graph, Equation (6.7).

$$\text{FS} = (40\text{LGC} + 150) \tag{6.7}$$

If the analyst is not evaluating a PSDL graph then equation (6.8) will suffice or another suitable sizing measure that has been backfired to E-SLOC.

$$\text{FS} = \text{E-SLOC} \tag{6.8}$$

The location parameter in the three-variable Weibull function implements the Functional Size in equation (6.9).

$$g = 0.5408 * \mathrm{FS}^{1.2092} - h \tag{6.9}$$

where

FS = the *functional size*

$h = -5.8\mathrm{E}-10FS^3 + 1.038\mathrm{E}-03FS^2 + 0.7724FS$

## D. MODIFIED RISK MODEL CHARACTERISTICS

The Modified Risk Model requires four primary input parameters, all of which are automatically collectable and derived extremely early in the software lifecycle.

- Organization. The MRM implements a measure to capture the *efficiency* of a software development organization.
- Complexity. The MRM architecture accommodates interface with the Computer Aided Prototyping System developed at the Naval Postgraduate School. (Dupo02) and this research are capable of deriving key complexity measures from the machine generated specification code. The MRM is capable of using different complexity measures as a "plug-ins"; thus allowing the model to interface with organizations not equipped with CAPS.
- Requirements. – A software project can be viewed as a finite set of issues that require resolution prior to project completion. These issues are not fully revealed in the beginning of the process. The MRM captures the stability of the known issues and adjusts projections based on the introduction or deletion of additional issues. As with the other model parameters, *requirements volatility* is completely adaptable to unique software development situations. A risk analyst can choose to monitor the change in the project's risk or implement static projections.
- Management Trade-offs – To successfully development software, a balance must exist between the Organization *(efficiency)*, Product Attributes *(complexity)*, and Project Stability (*requirements volatility*). In reality, this is not always the case. It becomes the responsibility of management to balance the equation. Management applies resources (time and people) to achieve a successful balance.

The Modified Risk Model informs management how well balanced is the software development. The risk analyst also has the ability to derive the management trade-offs within a confidence interval. With this information, management can implement any suitable staffing profile to achieve the model's projection.

Figure VI-8 revisits a primary issue with previous implementations of the Software Risk Model. The horizontal axis represents time, in days[51], and the vertical axis represents the probability of completion. Figure VI-8 illustrates the projection of nine different software development scenarios; Table VI-4 provides the details of the legend.

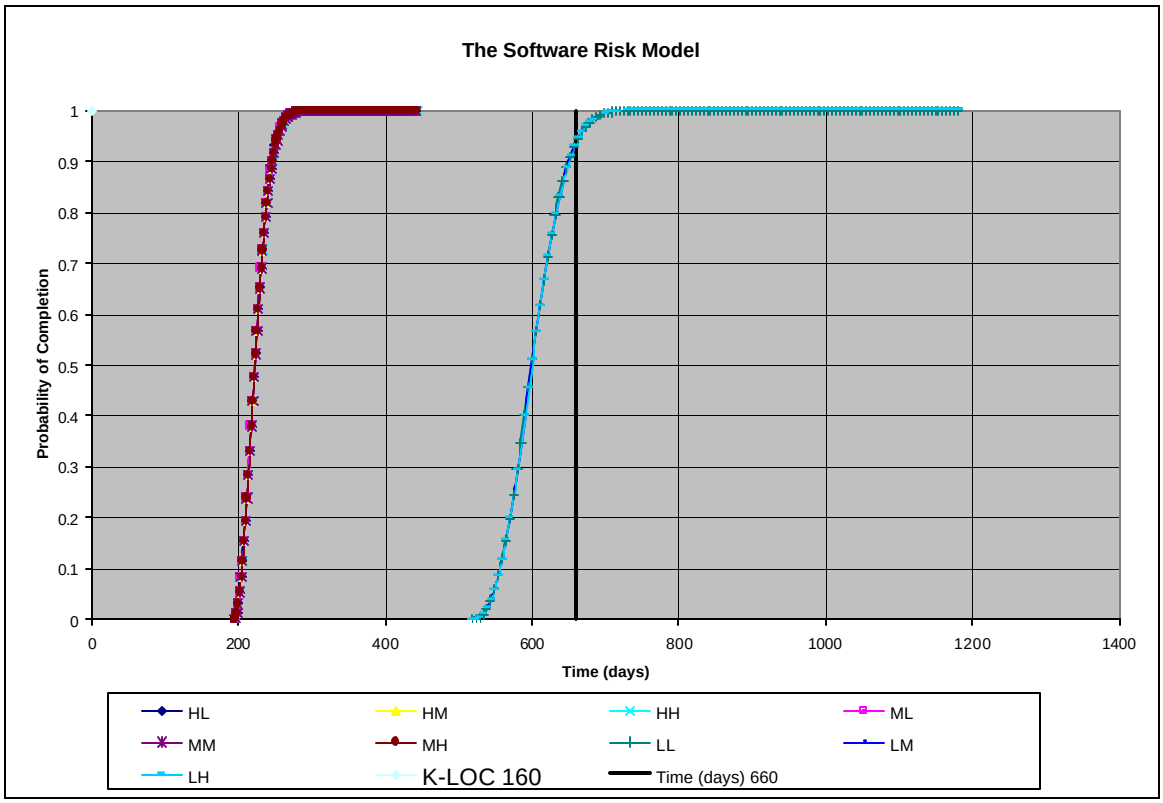



**Figure VI-8. Characteristic of the SRM.**

In this specific example, a value of 660 days is projected to determine the probability of completing a software project with a functional size of 160K E-SLOC.

[51] The MRM and SRM are fundamentally different; however, this example is concerned with the behavior of the models, not the actual scale values.

Due to the discrete nature of the SRM (explained in Chapters IV and V), the model is incapable of representing nine unique projections for each of the development scenarios. Six scenarios are represented in the curve on the left (all medium and high efficiency scenarios) and three are represented in the curve on the right (low efficiency scenarios).

| EF | RV | CX |
|---|---|---|
| L = 0.50 | L = 0.00 | n/a |
| M = 0.70 | M = 0.30 | n/a |
| H = 0.90 | H = 0.50 | n/a |

**Table VI-4. Legend for Figure VI-8.**

A major goal of MRM, when enhancing the SRM, is to remove the effects of discrete behavior from the model. Figure VI-9 demonstrates the same nine software development scenarios projected with the MRM[52]. The horizontal axis is scaled different than Figure VI-8; the MRM utilizes effort. However, the behavior of the model is clear. Nine unique scenarios will produce nine unique probability projections. Table VI-5 extends Table VI-4 with the addition of the *functional complexity.*

| EF | RV | FC |
|---|---|---|
| L = 0.50 | L = 0.00 | L = 0.0 |
| M = 0.70 | M = 0.30 | M = 3.0 |
| H = 0.90 | H = 0.50 | H = 5.0 |

**Table VI-5. Legend for Figure VI-9.**

Another important realization from Figure VI-9 is the evidence of the projections overlapping. This is important because of the following example revisited from Chapters IV & V. A low efficiency organization can develop a software product expending less effort than a high efficiency organization if the functional complexity and requirements volatility impacted the high efficiency's project more severely; even if the functional size is the same for each development.

[52] The projection line represents 360 man-months (30 months with a staff size of 12 personnel).

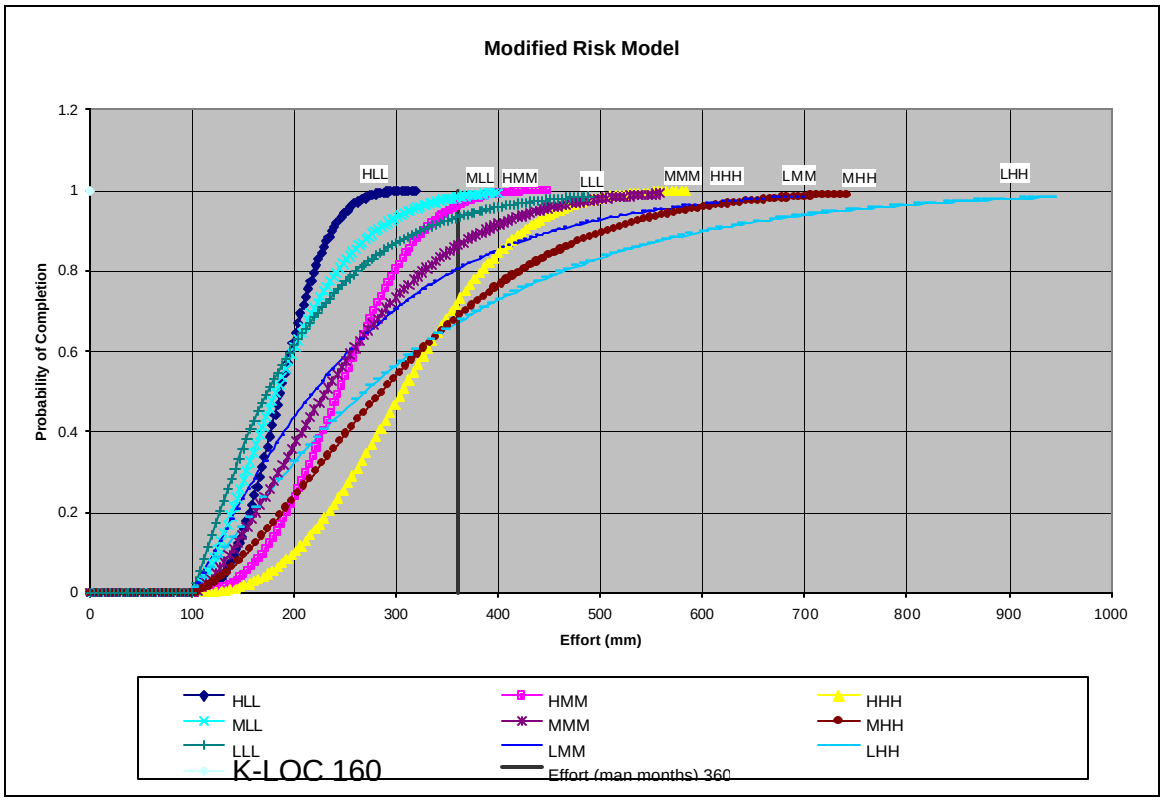



**Figure VI-9. Characteristic of the MRM.**

Figure VI-9 demonstrates six scenarios have above an 80% chance of completing the project by applying 360 man-months of effort. However, three scenarios demonstrate additional effort is required to deliver the software project. Trade-offs become evident in Figure VI-9. Table VI-6 extracts the calculated values from Figure VI-9.

| Scenario | Effort | %Ef decrease | Prob | %Prob increase |
|---|---|---|---|---|
| LLL | 384.8020 | | 0.9335 | |
| MLL | 314.0191 | -18.39% | 0.9824 | 5.23% |
| HLL | 251.9534 | -19.76% | 1.0000 | 1.79% |
| | | | | |
| LMM | 552.7310 | | 0.8042 | |
| MMM | 440.2097 | -20.36% | 0.8623 | 7.23% |
| HMM | 352.9292 | -19.83% | 0.9601 | 11.34% |
| | | | | |
| LHH | 743.5670 | | 0.6700 | |
| MHH | 583.6138 | -21.51% | 0.6850 | 2.24% |
| HHH | 459.5414 | -21.26% | 0.7219 | 5.38% |

Efficiency Changes

| Scenario | Effort | %Ef increase | Prob | %Prob decrease |
|---|---|---|---|---|
| LLL | 384.8020 | | 0.9335 | |
| LMM | 552.7310 | 43.64% | 0.8042 | -13.86% |
| LHH | 743.5670 | 34.53% | 0.6700 | -16.69% |
| | | | | |
| MLL | 314.0191 | | 0.9824 | |
| MMM | 440.2097 | 40.19% | 0.8623 | -12.23% |
| MHH | 583.6138 | 32.58% | 0.6850 | -20.56% |
| | | | | |
| HLL | 251.9534 | | 1.0000 | |
| HMM | 352.9292 | 40.08% | 0.9601 | -3.99% |
| HHH | 459.5414 | 30.21% | 0.7219 | -24.81% |

Requirements, Complexity Changes

**Table VI-6. Effort & Probability Sensitivity.**

Table VI-6 demonstrates that the required effort is decreased an average of 20.8% when the organization increases *efficiency* from a low (50% direct time) to a medium (70% direct time) efficiency organization. Similar results occur if the organization increases *efficiency* from a medium efficiency organization to a high (90% direct time) efficiency organization. Overall, the software project required an average decrease in effort of 36% by performing the software production with a high efficiency organization instead of a low efficiency organization. Also, demonstrated is probability of success increases with the increase of an organization's efficiency.

The effects of increasing the requirements volatility and complexity[53] reside in the lower portion of the table. The average increase in the required effort is 41% when the requirements and complexity increase from low (0.0%) to medium (30%); and then

[53] To maintain consistency between illustrations (Figure VI-8 and Figure VI-9), requirements volatility and functional complexity are considered in tandem. This is not a requirement in the MRM. Validations in this chapter and Appendix F consider the effects of requirements and complexity separately.

another 32% when the requirements and complexity increase from medium (30%) to high (50%). Overall, a low efficiency organization is the least prepared to adapt to additional complexity and volatile requirements.

THIS PAGE INTENTIONALLY LEFT BLANK

# VII. MODIFIED RISK MODEL VALIDATION

Chapter VI presents a software risk assessment model capable of projecting the probability of successfully delivering a software project. The Modified Risk Model is a dynamic macro model developed to aid program managers effectively gauge and plan the rough order of magnitude *effort* to deliver software solutions. The model projects the *probability* of completing a software project or the required *effort*, subject to the available resources supplied by management. This approach to software project risk management is unique because the model's input parameters can be derived. Subjective variables are not part of the model. Different program managers would derive the same projections on the same software project.

Validation of the Modified Risk Model extends to approximately 2,000 software projects. As demonstrated in Chapter VI and Appendix E, these projects originate from several application domains. The validation on the post-mortem projects is a variation from the original intent of the risk assessment model; operational tests have not been conducted on the model[54].

The validation strategy extends the approach implemented in Chapters IV and Appendix C. Three different levels of abstraction are considered in the application subset. First, the overall performance of the MRM is evaluated on the entire project subset. Comparisons are provided to evaluate the performance of the Basic COCOMO and Simplified Software Equation. Second, the level of detail is increased by decomposing the project subset into the three application domains introduced in Chapter VI: *Real Time, Engineering*, and *Informational*. Finally, at the atomic decomposition, analysis is presented on eight[55] individual project applications.

The chapter begins by discussing the interface with the project subset. The evaluation criteria are reviewed followed by a presentation of the actual results; Chapter

[54] The MRM and the SRM were designed to provide early projection in the software lifecycle. No validation has been conducted on on-going software development.

[55] The project subset contains ten application categories. Twelve applications are subjectively removed: one from the Microcode, 11 from Miscellaneous (unknown).

VII presents the first two levels of validation detail and Appendix F provides the atomic level of detail. Appendix F also contains the interface details and mitigation for the differences between the Basic COCOMO and Simplified Software Equation.

## A. INTERFACE WITH PROJECT DATABASE

Chapter IV , the validation of the SRM, required a specific mapping to extract values from the QSM® database. The MRM requires a mapping as well. Because the MRM requires four primary input parameters (*efficiency*, *requirements volatility*, *functional complexity*, and *functional size*), four satisfactory mapping techniques are required.

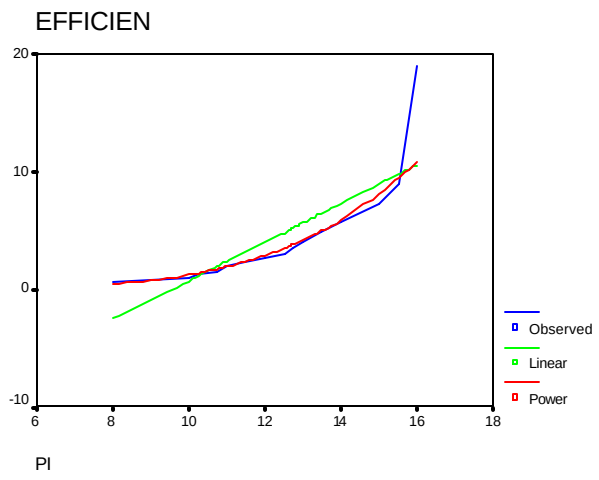
**Efficiency (EF).** During the validation attempts of the SRM, the *efficiency* of the organization is mapped to either a low or high value by establishing a dividing line on the productivity index. This served the SRM well since the model could only utilize an organization on a bi-polar scale. However, the MRM requires a different approach.

The MRM is a continuous model, so careful consideration went into the implementation of a mapping for the *efficiency*. The validation continues to utilize the productivity index of the developing organization in the database. However, a technique is required to allow for continuous representation. (Nogu00) considered a low efficiency organization one to implement only about 45% direct time on project development. A high efficiency organization can approach as high as 95% direct time. This research supports this interpretation.

The mapping strategy, as demonstrated in Figure VII-1, considered the bounded range of projects with *productivity indexes* of eight through 16. Efficiency ranges were then considered after bounding the *direct time* available on a software project (40% - 95%). Next, a power function is derived using (SPSS99) using the *productivity index* as the independent variable and computes a dependent variable to represents the developing unit's *efficiency*. Equation (7.1) translates all of the productivity indexes for all of the validation.

$$\text{Efficiency} = PI^{4.6191} * e^{-10.4} \qquad (7.1)$$

| Efficiency | Productivity Index |
|---|---|
| 0.67 | 8.0 |
| 0.82 | 9.0 |
| 1.00 | 10.0 |
| 1.22 | 10.3 |
| 1.50 | 10.8 |
| 1.86 | 11.0 |
| 2.33 | 11.5 |
| 2.70 | 12.0 |
| 3.00 | 12.5 |
| 3.55 | 12.8 |
| 4.00 | 13.0 |
| 4.88 | 13.5 |
| 5.67 | 14.0 |
| 7.33 | 15.0 |
| 9.00 | 15.5 |
| 19.00 | 16.0 |




Independent: PI

| Dependent | Mth | Rsq | d.f. | F | Sigf | b0 | b1 |
|---|---|---|---|---|---|---|---|
| EFFICIEN | LIN | .671 | 14 | 28.54 | .000 | -15.625 | 1.6356 |
| EFFICIEN | POW | .952 | 14 | 278.94 | .000 | 3.0E-05 | 4.6191 |

**Figure VII-1. Mapping Productivity Index with Efficiency.**

**Requirements Volatility (RV).** The validation strategy is implemented identical to the strategy presented in Chapter IV and Appendix C. In the database from QSM®, there exist a measure that is collected called *Requirements Growth Percentage*. This measure documents how much the project requirements changed from the original plan. The *Requirements Growth Percentage* is mapped one-to-one for all of the MRM validation.

**Functional Complexity (FC).** The CAPS prototyping environment provides the primary analysis for the *functional complexity.* Since the project subset for validation was not developed utilizing CAPS, interfacing with the subset requires an alternative method to determine the *functional complexity.* The validation of the MRM utilizes Figure VII-2 reproduced from Chapter VI.

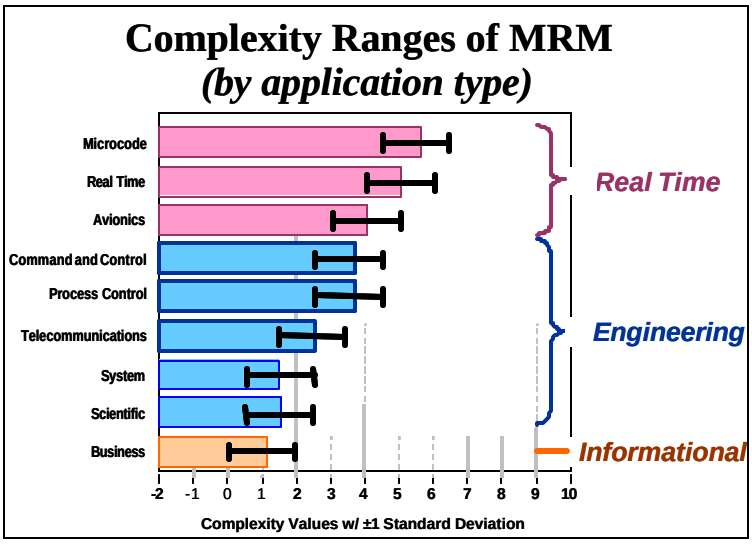



**Figure VII-2. Complexity Ranges of the MRM.**

Figure VII-2, developed from the trend line analysis of Chapter VI, segregates the nine different application sub-types into appropriate *functional complexity.* The application sub-types are abstracted into three application types: *real time, engineering,* and *informational*.

For example, a software project developing a telecommunication application may not have the functional complexity automatically derived in the CAPS environment. In this case, an analyst can refer to Figure VII-2. The chart indicates that a telecommunication development is part of the engineering family of application types. The appropriate functional complexity value is in the range of 1.5 to 3.5.

**Functional Size.** As with the *functional complexity,* the *functional size* should be obtained from the PSDL source code. The procedure for doing so is detailed in Chapter VI. However, the *functional size* in the MRM is calibrated with E-SLOC as the base-sizing unit. This convenient feature affords the validation of the MRM to implement a direct one-to-one mapping between the MRM's *functional size* and the E-SLOC of the project subset.

Table VII-1 presents a summary of the mapping interface. This interface is used for all of the validation of the MRM.

| MRM Parameter | Project Database[56] |
|---|---|
| Efficiency (EF) | EF = 3.0E-5*Productivity Index$^{4.6191}$ |
| Requirements Volatility (RV) | RV = Requirements Growth Percentage |
| Functional Complexity (FC) | FC = Figure VII-2 |
| Functional Size (FS) | FS = Effective SLOC |

**Table VII-1. Consolidated Mapping Parameters.**

## B. VALIDATION SETUP

The validation of the Modified Risk Model constitutes the same rigor administered to the SRM validation documented in Chapter IV. However, there exist some primary differences listed in Table VII-2:

| SRM Validation (Chapter IV) | MRM Validation (Chapter VII) |
|---|---|
| Evaluates how well the three models[57] project the required development time. | Evaluates how well the three models[58] project the required effort. |
| Evaluates 112 software projects, predominantly real time & engineering. | Evaluates over 1900 software projects from real time, engineering, and informational. |
| Evaluate three versions of the COCOMO and Simplified Software Equation (i.e. does not consider the application type of the projects). | Evaluates the specific version of the COCOMO and Simplified Software Equation tailored to application type. |

**Table VII-2. Validation Distinctions.**

The remainder of this chapter documents the performance of three software estimation models against five evaluation criteria; projecting the required effort of over 1900 software projects. As indicated, the validation is documented at three levels of abstraction. The first and second levels are contained within this chapter; however, Appendix F contains the third level of abstraction.

The highest level of abstraction is a composite of all the models performed against the entire subset of projects. There is no consideration for the application families or the application sub-types. Figure VII-2 provides some insights to the

56 Additional information regarding the project database is contained in Appendix A.

57 Software Risk Model, Basic COCOMO, and the Simplified Software Equation.

58 Modified Risk Model, Basic COCOMO, and the Simplified Software Equation.

development of levels two and three. Level two decomposes the software project subset into the three application families contained in the figure: *real time*, *engineering*, *informational*. The projects on the left of the figure explain the application sub-types contained within. Essentially, level two abstracts the analysis to three tables. Finally, the atomic level considers each of the eight application sub-types and documents how each of the three models performed (Appendix F).

### 1. Evaluation Criteria

The following evaluation criteria were utilized in the SRM validation and are applicable to the MRM validation. In order to fully validate each of the model's projections, the five rating criterion are revisited. The criteria are weighted based on their relevance to the model's success. The following, in the order of precedent, is a complete list of the rating criteria. The implementation details are addressed in Appendix F.

- Actual Error. A measure of the average error produced by each model calculated by $\text{actual\_error} = \frac{\text{projected} - \text{actual}}{\text{actual}}$. Negative values indicate an under-estimate.
- Absolute Error. A measure of the average error produced by each model calculated by $abs(\text{actual\_error})$. There is no distinction between whether the model projected high or low. Useful for providing a single measure of the relative error.
- Balance. A model that projects too optimistically can prove disastrous to a software project. Evaluates how evenly the model projects estimates (i.e. are the over-estimates proportional to the under-estimates). A model projecting with closer "balance" receives a higher rating.
- Under Estimation. The fourth consideration is designed to evaluate the actual error of an optimistic projection. This criterion only considers the individual under projections and computes the average.
- Over Estimation. The least weighted criteria is over estimation. In actuality, less damage can potentially occur from projecting too cautiously. This criterion only considers the individual over projections and computes the actual error.

## 2. Table Properties

Each table presented throughout this validation chapter and in Appendix F follows the same format. Summary information is contained on the left side of the table. Then, in order of precedence, the results from each of the validation criteria are presented. The columns in Table VII-3 and all validation tables represent the following:

- *Model* – describes the model being recorded (MRM, COCOMO, SSE)
- *N* – the total number of projected considered by the model
- *CORREL* – the correlation coefficient. The correlation coefficient is used to determine the relationship between two properties (MSEX02). The number is interpreted as the proportion of the total variation (SPSS99).
- *Actual Error (wt 5)* – determined by

$$\text{actual\_error} = \frac{\text{projected} - \text{actual}}{\text{actual}} \tag{7.2}$$

  - *Mean* – the average error
  - *Standard Deviation* – of the average
  - *Rank* – low is best, high is the worst
  - *Score* – $\text{Score}_{actual\_error} = \text{rank}_{actual\_error} * \text{weight}_{actual\_error}$

- *Absolute Error (wt 4)* – determined by *abs*(actual_error)
  - *Mean* – the average
  - *Standard Deviation* – of the average error
  - *Rank* – low is best, high is the worst
  - *Score* - $\text{Score}_{absolute\_error} = \text{rank}_{absolute\_error} * \text{weight}_{absolute\_error}$

- *Balance (wt 3)* – a measure of the dispersion below and above the actual value.
  - *% Under* – the percentage of N that are projected below actual values
  - *Rank* – values closer to 50% receive the highest ranking. This would indicate that approximately 50% fall below and 50% fall above the actual values.

- *Score* - $\text{Score}_{balance} = \text{rank}_{balance} * \text{weight}_{balance}$

- *Under Estimation (wt 2)* – of the conservative projections, what is the average error?
- *N-under* – the raw number of projections short of the actual value.
  - *Mean* – the average error
  - *Standard Deviation* – of the average error
  - *Rank* – low is best, high is the worst
  - *Score* - $\text{Score}_{under\_error} = \text{rank}_{under\_error} * \text{weight}_{under\_error}$

- *Over Estimation (wt 1)* – of the over-optimistic projections, what is the average error?
- *N-over* – the raw number of projections over the actual value.
- *Mean* – the average error
- *Standard Deviation* – of the average error
- *Rank* – low is best, high is the worst
- *Score* - $\text{Score}_{over\_error} = \text{rank}_{over\_error} * \text{weight}_{over\_error}$
- *Total* – a smaller number is better

$$total = Score_{actual\_error} + Score_{absolute\_error} + Score_{balance} + Score_{under\_error} + Score_{over\_error}$$

## C. MRM VALIDATION - LEVEL ONE (ALL APPLICATIONS)

Table VII-3, the most abstract view, provides the crudest level of detail regarding the performance of the three models. The table is a consolidation of each of the eight project sub-types. A validation for the ninth project sub-type, *microcode*, is not provided because sufficient microcode applications do not exist in the project database.

Additionally, a quad chart is provided for each of the three models. The quad chart illustrates four views to support four out of the five[59] evaluation criteria. The four views presented, beginning in the top left quadrant and continuing left to right, are:

[59] The fifth view, balance, does not require a figure to elaborate.

- Actual Error
- Absolute Error
- Under Estimation
- Over Estimation

# All Applications

| All Application Types | | | | | Actual Error (wt 5) | | | | Absolute Error (wt 4) | | | | Balance (wt 3) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Consolidated Average | N | 5% | Sigma | Correl | Mean | Std | Rank | Score | Mean | Std | Rank | Score | % Under | Rank | Score |
| MRM | 1930 | 97 | 722 | 0.8513 | 63.00% | 121.00% | 3 | 15 | 98.00% | 96.00% | 3 | 12 | 32.73% | 2 | 6 |
| COCOMO | 1930 | 97 | 722 | 0.8456 | 42.00% | 96.00% | 2 | 10 | 73.00% | 76.00% | 1 | 4 | 41.57% | 1 | 3 |
| SSE | 1930 | 97 | 722 | 0.8422 | -18.00% | 139.00% | 1 | 5 | 83.00% | 113.00% | 2 | 8 | 78.27% | 3 | 9 |
| | | | | | Under Estimation (wt 2) | | | | | Over Estimation (wt 1) | | | | | |
| SSE required | 287 | | | | N-Under | Mean | Std | Rank | Score | N-Over | Mean | Std | Rank | Score | Total |
| projects removed. | | | | MRM | 600 | -52.00% | 56.00% | 2 | 4 | 1233 | 120.00% | 103.00% | 2 | 2 | 39 |
| | | | | COCOMO | 762 | -37.00% | 23.00% | 1 | 2 | 1071 | 99.00% | 88.00% | 1 | 1 | 20 |
| | | | | SSE | 1210 | -65.00% | 25.00% | 3 | 6 | 335 | 150.00% | 225.00% | 3 | 3 | 31 |

**Table VII-3. Overall Summary of Validation Results.**

There exist a total of 1930 software applications in the project subset[60]. The five percent column represents that each model had a total of 97 projects (5%) removed[61] from the validation reducing the number considered to 1833. The average standard deviation of the required effort is 722 man-months. Each model has a correlation coefficient of about 85%. An additional 287 projects were removed from consideration for the Simplified Software Equation[62].

Several entries in Table VII-3 provide valuable insight. The reader is reminded that Table VII-3 is a consolidation of the entire project subset. On the surface, it would appear that the Basic COCOMO out performed the MRM and SSE. In reality, as the remainder of the chapter demonstrates, each model provides its own strengths and weaknesses.

60 This total includes the five percent noted in footnote 61 below.

61 The worst five percent, for each model, were removed to minimize outliers. The same five percent were not necessarily removed from each model.

62 (Putn92) bounds the conditions that the SSE is effective. Projecting applications below this set of minimums can provide erroneous results. If an application exceeded these minimums, it was removed from the analysis.

## 1. MRM Performance on All Applications

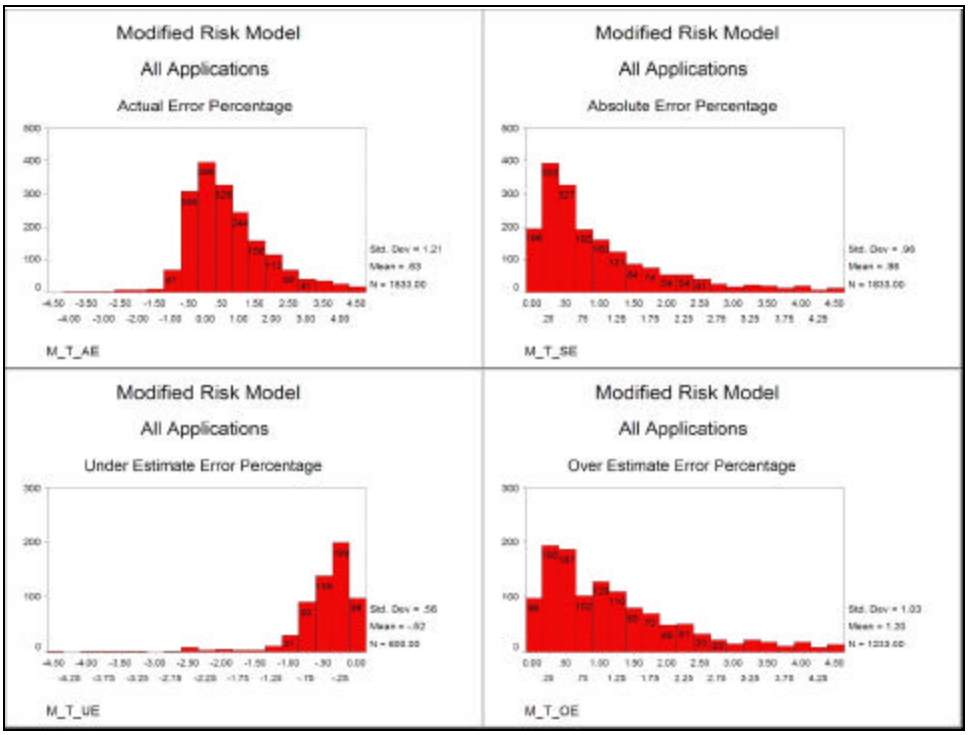



**Figure VII-3. MRM Results on All Applications.**

With the goal of projecting 25% of all projects within 25% of the actual value, the MRM achieved 27.4% for all projects.

- Actual Error – The mean error is 0.63 or an average of 63% from the actual value. The average error has a standard deviation of 121%. A total of 589 projects, or 32%, are projected within 25% of the actual value. The MRM ranked 3 of 3 for average actual error.
- Absolute Error – The absolute error considers all errors (either over or under) as equal and provides a single value to represent the error. The MRM projected the actual project performance with an absolute error of 98% ± 96%. The MRM ranked 3 of 3 for average absolute error.
- Under Estimate – The MRM projected 600 out of 1833 projects under the actual value, 38%. When the MRM projects short, it does so with an average error of 52%. The majority (73%) of the under estimates occur between zero and 50 percent. The MRM ranked 2 of 3 for balance and 2 of 3 for average under-estimation error.

- Over Estimation – A model projecting beyond the required effort can prevent successful bidding on project proposals. However, the effects of over estimating a project are far fewer than the potential effects of under estimation. The MRM over estimated 1233 projects. Forty percent of the over-estimates were between zero and 50 percent. The MRM ranked 2 of 3 for average over-estimation error.

## 2. COCOMO Performance on All Applications

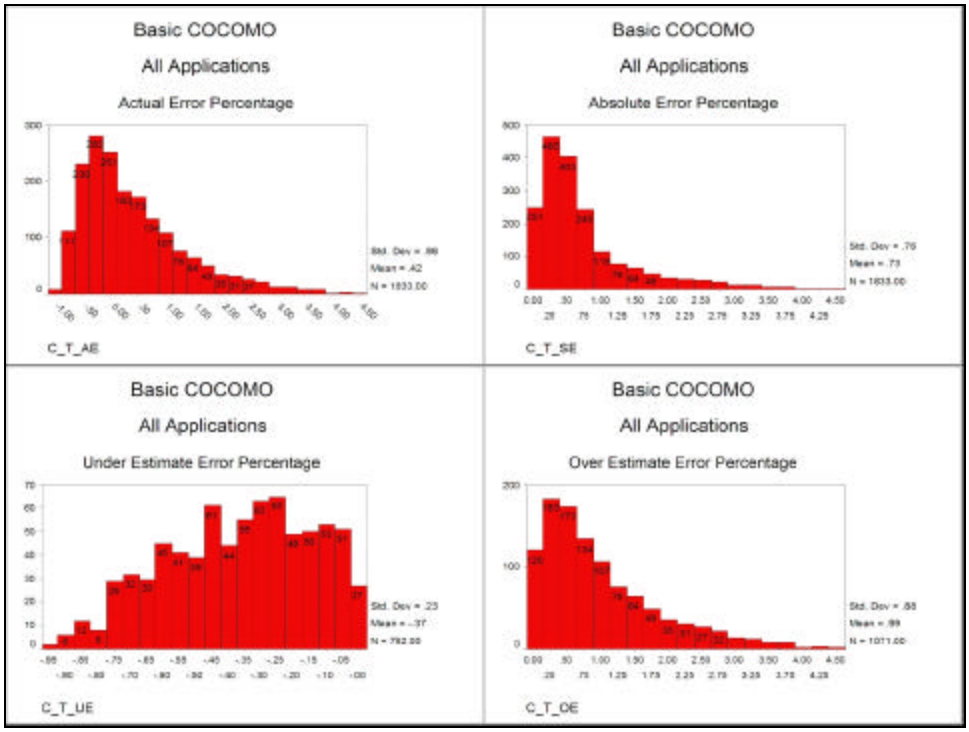



**Figure VII-4. COCOMO Results on All Applications.**

- Actual Error – The mean error is 0.42 or an average of 42% from the actual value. The average error has a standard deviation of 96%. A total of 716 projects, or 39%, are projected within 25% of the actual value. The Basic COCOMO ranked 2 of 3 for average actual error.
- Absolute Error – The Basic COCOMO projected the actual project performance with an absolute error of 73% ± 76%. The Basic COCOMO ranked 1 of 3 for average absolute error.
- Under Estimate – The Basic COCOMO projected 763 out of 1833 projects under the actual value, 42%. When the Basic COCOMO projects short, it does so with an average error of 37%. The majority (62%) of the under

estimates occur between zero and 50 percent. The Basic COCOMO ranked 1 of 3 for balance and 1 of 3 for average under-estimation error.

- Over Estimation – The Basic COCOMO over estimated 1071 projects. Twenty-eight percent of the over-estimates were between zero and 50 percent. The Basic COCOMO ranked 1 of 3 for average over-estimation error.

### 3. SSE Performance on All Applications

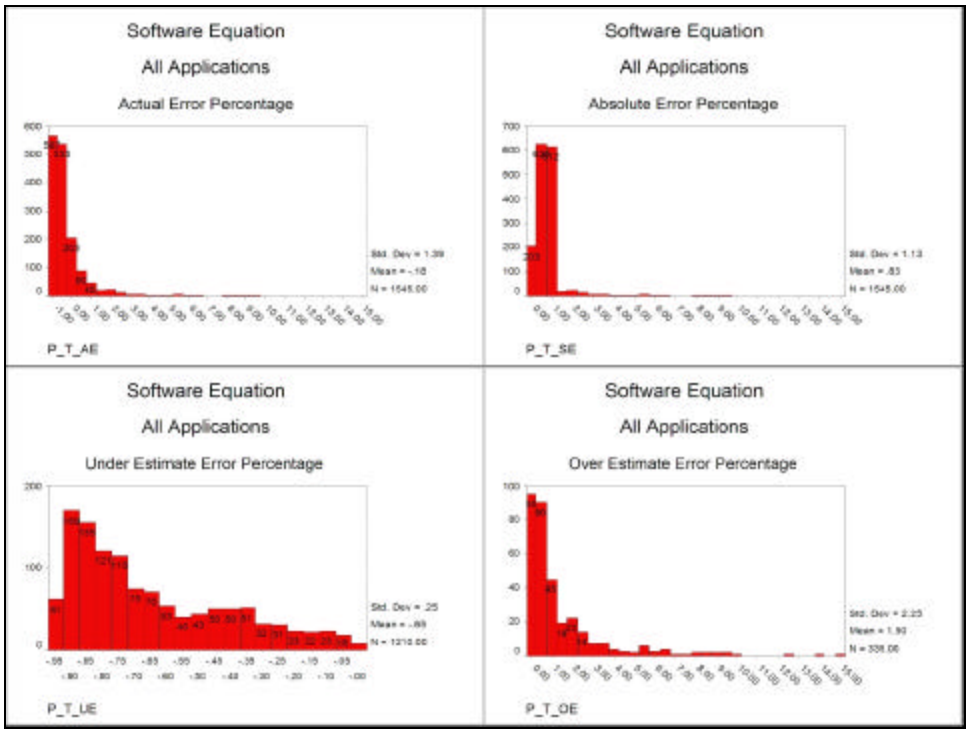



**Figure VII-5. SSE Results on All Applications.**

- Actual Error – The mean error is -0.18 or an average of 18% below the actual value. The average error has a standard deviation of 139%. A total of 308 projects, or 20%, are projected within 25% of the actual value. The SSE ranked 1 of 3 for average actual error.
- Absolute Error – The SSE Equation projected the actual project performance with an absolute error of 83% ± 113%. The SSE ranked 2 of 3 for average absolute error.
- Under Estimate – The SSE projected 1210 out of 1545 projects under the actual value, 78%. When the SSE projects short, it does so with an average error of 65%. Twenty-eight percent of the under estimates occur

between zero and 50 percent. The SSE ranked 3 of 3 for balance and 3 of 3 for average under-estimation error.

- Over Estimation – The SSE over estimated 335 projects. Fifty-five percent of the over-estimates were between zero and 50 percent. The SSE ranked 3 of 3 for average over-estimation error.

Overall the MRM projected with the most absolute and actual error and placed second for balance, over and under error projections. The five rating criteria provide a simple holistic analysis of each model's performance. However, at the consolidated level of detail presented in Table VII-3, several questions remain unanswered that require a more detailed analysis.

## D. MRM VALIDATION - LEVEL TWO

Attention now focuses on the specifics that comprise the data presented in the previous validation. This portion of the validation considers all of the software projects according to the application family: *real time, engineer*, or *informational*. Figure VII-2 above illustrates that the *real time* family is comprised of microcode[63], real time systems, and avionic systems. The *engineer* family is comprised of five application sub-types: command & control, process control, telecommunications, systems software, and scientific applications. The *informational* family is comprised of only the business applications.

### 1. Real Time Applications

Real Time applications (real time and avionic) comprise less than five percent of the total project population in the project subset, but are the most important for the primary application of the MRM; support to the Computer Aided Prototyping System. Table VII-4 summarizes the performance of the three models followed by quad charts in Figure VII-6, Figure VII-7, and Figure VII-8.

63 There exists only one microcode application in the project subset. This project is not considered during the validation.

Figure VII-2 details that a *functional complexity* in the range of three to six should be implemented for *real time* applications; the validation implements a value of four for real time and three for avionic. The *real time* family of applications is comparable to the COCOMO notion of an *embedded* mode. For all of the Basic COCOMO projections the embedded mode is utilized. The Simplified Software Equation implements its specialized productivity parameter for each application sub-type: 1947 for real time and avionic.

# Real Time Applications

| Real Time Applications | | | | | Actual Error (wt 5) | | | | Absolute Error (wt 4) | | | | Balance (wt 3) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Consolidated | N | 5% | Sigma | Correl | Mean | Std | Rank | Score | Mean | Std | Rank | Score | % Under | Rank | Score |
| MRM | 63 | 3 | 713 | 0.9287 | 48.26% | 101.82% | 1 | 5 | 71.78% | 86.59% | 1 | 4 | 35.09% | 1 | 3 |
| COCOMO | 63 | 3 | 713 | 0.9363 | 75.66% | 118.78% | 2 | 10 | 90.98% | 107.30% | 2 | 8 | 25.06% | 2 | 6 |
| SSE | 63 | 3 | 713 | 0.9310 | 452.27% | 346.89% | 3 | 15 | 453.13% | 345.74% | 3 | 12 | 1.67% | 3 | 9 |
| | | | | | Under Estimation (wt 2) | | | | | Over Estimation (wt 1) | | | | | |
| *SSE required* | 5 | | | | N-Under | Mean | Std | Rank | Score | N-Over | Mean | Std | Rank | Score | Total |
| *projects removed.* | | | | MRM | 21 | -33.60% | 30.87% | 3 | 6 | 39 | 92.34% | 99.49% | 1 | 1 | **19** |
| | | | | COCOMO | 15 | -30.63% | 22.20% | 2 | 4 | 45 | 111.09% | 116.73% | 2 | 2 | **30** |
| | | | | SSE | 1 | -23.86% | n/a | 1 | 2 | 54 | 461.08% | 343.87% | 3 | 3 | **41** |

**Table VII-4. Real Time Application Data.**

The *real time* application family is comprised of 63 projects. Each model had a total of three projects removed from the validation. The average standard deviation is 713 man-months. Each model has a correlation coefficient above 90%. A total of five projects were removed from consideration for the Simplified Software Equation.

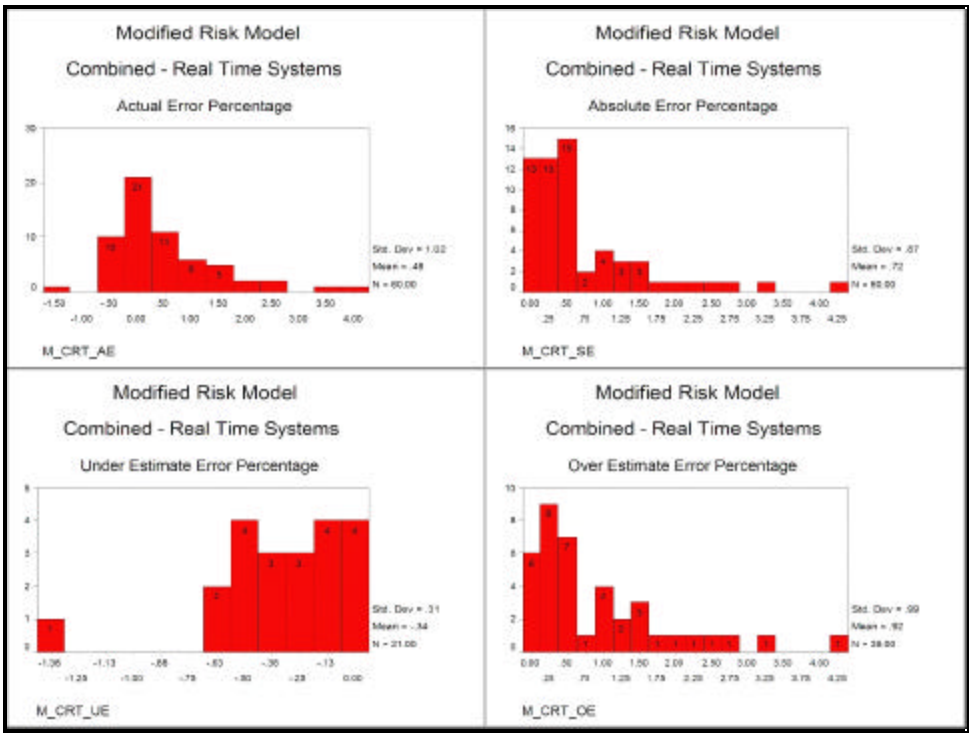



**Figure VII-6. MRM - Real Time Results.**

The Modified Risk Model projected 36.24% of all of the *real time* applications within 25% of the actual value. The MRM ranked 1st in overall model performance for *real time* applications:

- Actual Error – The mean error is 0.48 or an average of 48% from the actual value. The average error has a standard deviation of 102%. Twenty-one projects, or 35%, are projected within 25% of the actual value. The MRM ranked 1 of 3 for average actual error.
- Absolute Error – The MRM projected the actual project performance with an absolute error of 72% ± 87%. The MRM ranked 1 of 3 for average absolute error.
- Under Estimate – The MRM projected 21 out of 60 projects under their actual value, 35%. When the MRM projects short, it does so with an average error of 34%. The majority (85%) of the under estimates occur between zero and 50 percent. The MRM ranked 1 of 3 for balance and 3 of 3 for average under-estimation error.

- Over Estimation – The MRM over estimated 39 projects. Fifty-six percent of the over-estimates were between zero and 50 percent. The MRM ranked 1 of 3 for average over-estimation error.

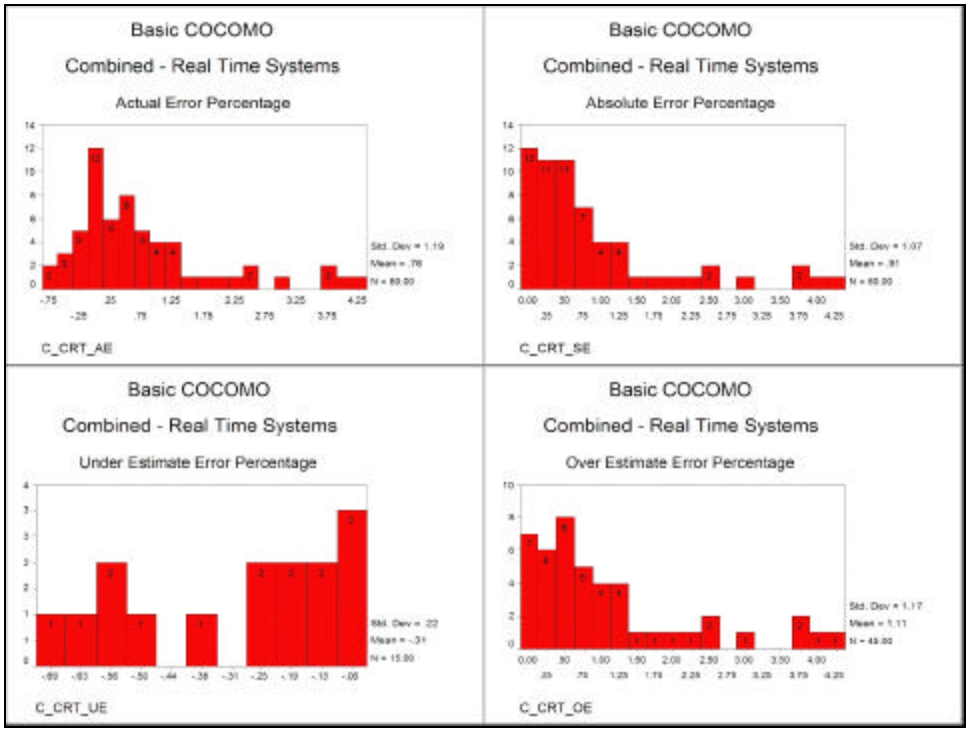



**Figure VII-7. COCOMO - Real Time Results.**

The Basic COCOMO ranked 2nd in overall model performance for *real time* applications:

- Actual Error – The mean error is 0.76 or an average of 76% from the actual value. The average error has a standard deviation of 119%. Twelve projects, or 20%, are projected within 25% of the actual value. The Basic COCOMO ranked 2 of 3 for average actual error.
- Absolute Error – The Basic COCOMO projected the actual project performance with an absolute error of 91% ± 107%. The Basic COCOMO ranked 2 of 3 for average absolute error.
- Under Estimate – The Basic COCOMO projected 15 out of 60 projects under the actual value, 25%. When the Basic COCOMO projects short, it does so with an average error of 31%. The majority (73%) of the under estimates occur between zero and 50 percent. The Basic COCOMO ranked 2 of 3 for balance and 2 of 3 for average under-estimation error.

- Over Estimation – The Basic COCOMO over estimated 45 projects. Forty-six percent of the over-estimates were between zero and 50 percent. The Basic COCOMO ranked 2 of 3 for average over-estimation error.

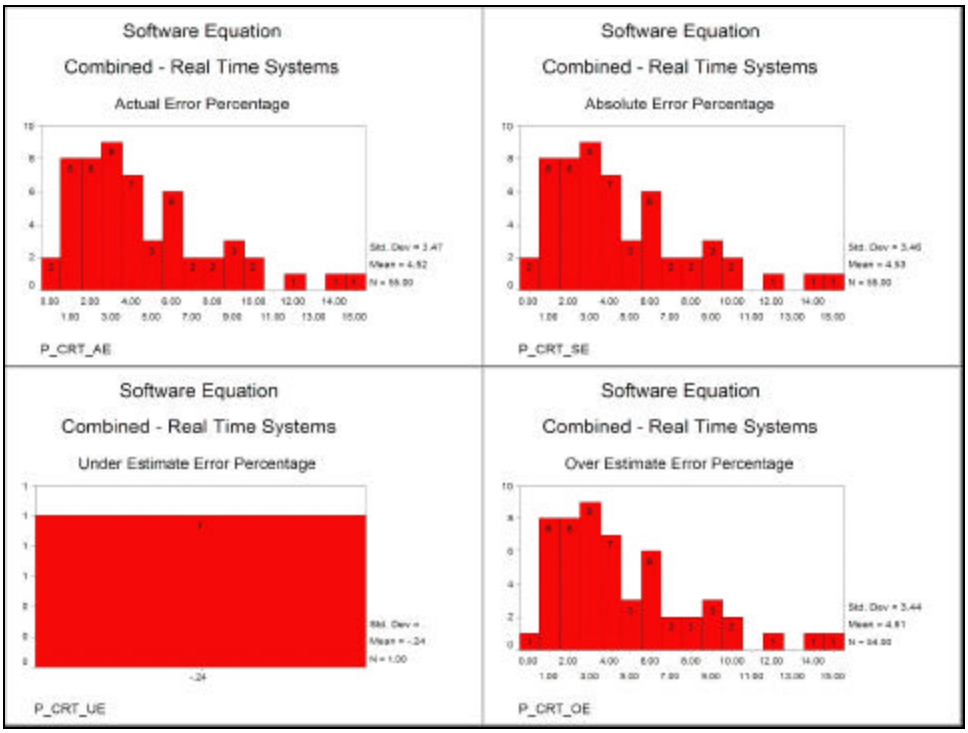



**Figure VII-8. SSE - Real Time Results.**

The Simplified Software Equation ranked 3rd in overall model performance for real time applications:

- Actual Error – The mean error is 4.52 or an average of 452% from the actual value. The average error has a standard deviation of 347%. Two projects are projected within 25% of the actual value. The SSE ranked 3 of 3 for average actual error.
- Absolute Error – The SSE projected the actual project performance with an absolute error of 453% ± 346%. The SSE ranked 3 of 3 for average absolute error.
- Under Estimate – The SSE projected one project under the actual value. When the SSE projects short, it does so with an average error of 24%. The SSE ranked 3 of 3 for balance and 1 of 3 for average under-estimation error.

- Over Estimation – The SSE over estimated 54 projects. One project was over-estimated within 50 percent. The SSE ranked 3 of 3 for average over-estimation error.

### 2. Engineer Application

*Engineer* application is the family of applications comprised of: command & control, process control, telecommunications, systems software, and scientific applications. *Engineer* applications comprise over twenty percent of the total project population in the project subset. Table VII-5 summarizes the performance of the three models followed by quad charts in Figure VII-9, Figure VII-10, and Figure VII-11.

Figure VII-2 details that a functional complexity in the range of 0.5 to 4.5 should be implemented for engineer applications. The MRM validation implements these functional complexity values:

- Command & control (FC of 2.5)
- Process control (FC of 2.5)
- Telecommunications (FC of 1.5)
- Systems software (FC of 0.5)
- Scientific applications (FC of 0.5)

The engineer family of applications is comparable to the COCOMO notion of a *semi-detached* mode. For all of the Basic COCOMO projections the semi-detached mode is utilized. The Simplified Software Equation implements its specialized productivity parameters for each application sub-type:

- Command & control (PP of 4181)
- Process control (PP of 5186)
- Telecommunications (PP of 8362)
- Systems software (PP of 13530)

- Scientific applications (PP of 13530)

# Engineering Applications

| Engineering Applications | | | | | Actual Error (wt 5) | | | | Absolute Error (wt 4) | | | | Balance (wt 3) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Consolidated | N | 5% | Sigma | Correl | Mean | Std | Rank | Score | Mean | Std | Rank | Score | % Under | Rank | Score |
| MRM | 397 | 20 | 805 | 0.8633 | 44.82% | 96.61% | 2 | 10 | 76.41% | 74.12% | 1 | 4 | 38.98% | 3 | 9 |
| COCOMO | 397 | 20 | 805 | 0.8508 | 50.57% | 105.41% | 3 | 15 | 80.16% | 85.04% | 2 | 8 | 40.30% | 2 | 6 |
| SSE | 397 | 20 | 805 | 0.8447 | 38.70% | 140.19% | 1 | 5 | 88.46% | 115.36% | 3 | 12 | 46.67% | 1 | 3 |
| | | | | | Under Estimation (wt 2) | | | | Over Estimation (wt 1) | | | | | | |
| SSE required | 42 | | | | N-Under | Mean | Std | Rank | Score | N-Over | Mean | Std | Rank | Score | Total |
| projects removed. | | | | MRM | 147 | -40.51% | 31.85% | 2 | 4 | 230 | 99.36% | 83.76% | 1 | 1 | 28 |
| | | | | COCOMO | 152 | -36.69% | 21.99% | 1 | 2 | 225 | 109.53% | 98.31% | 2 | 2 | 33 |
| | | | | SSE | 176 | -47.21% | 26.18% | 3 | 6 | 158 | 134.40% | 153.07% | 3 | 3 | 29 |

**Table VII-5. Engineer Application Data.**

*Engineer* applications are comprised of 397 projects. Each model had a total of 20 projects removed from the validation. The average standard deviation is 805 man-months. Each model has a correlation coefficient of about 85%. A total of 42 projects were removed from consideration for the Simplified Software Equation.

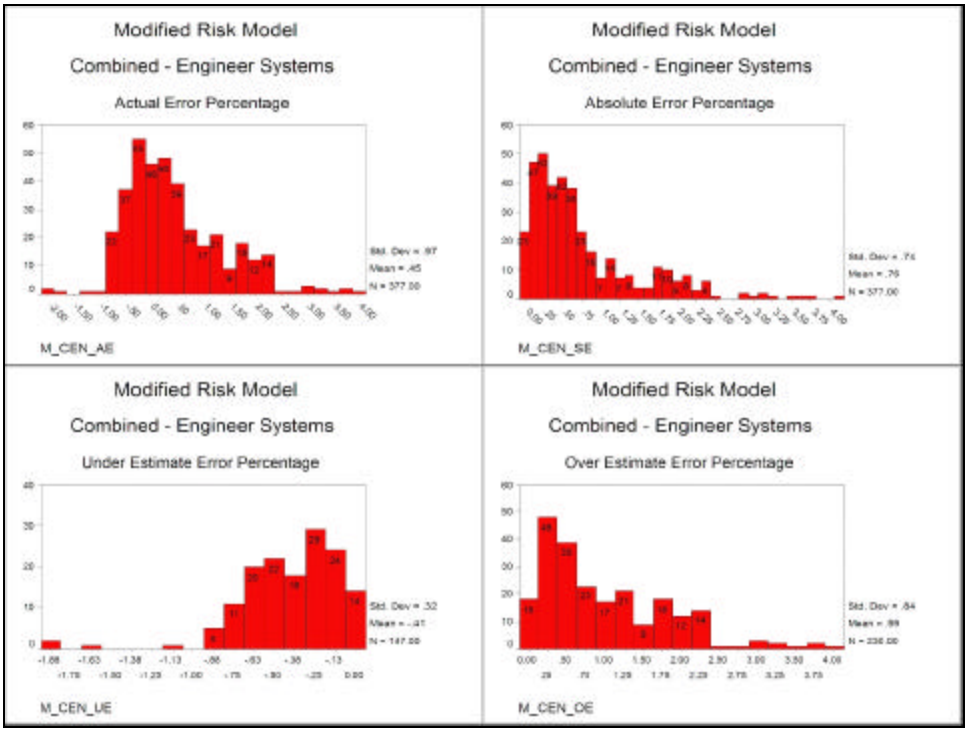



**Figure VII-9. MRM - Engineer Results.**

The Modified Risk Model projected 25.35% of all of the *engineer* applications within 25% of the actual value. The MRM ranked 1st in overall model performance for engineer applications with the Simplified Software Equation in a close second:

- Actual Error – The mean error is 0.45 or an average of 45% from the actual value. The average error has a standard deviation of 97%. One hundred twenty projects, or 32%, are projected within 25% of the actual value. The MRM ranked 2 of 3 for average actual error.
- Absolute Error – The MRM projected the actual project performance with an absolute error of 76% ± 74%. The MRM ranked 1 of 3 for average absolute error.
- Under Estimate – The MRM projected 147 out of 377 projects under their actual value, 39%. When the MRM projects short, it does so with an average error of 41%. The majority (73%) of the under estimates occur between zero and 50 percent. The MRM ranked 3 of 3 for balance and 2 of 3 for average under-estimation error.
- Over Estimation – The MRM over estimated 230 projects. Forty-six percent of the over-estimates were between zero and 50 percent. The MRM ranked 1 of 3 for average over-estimation error.

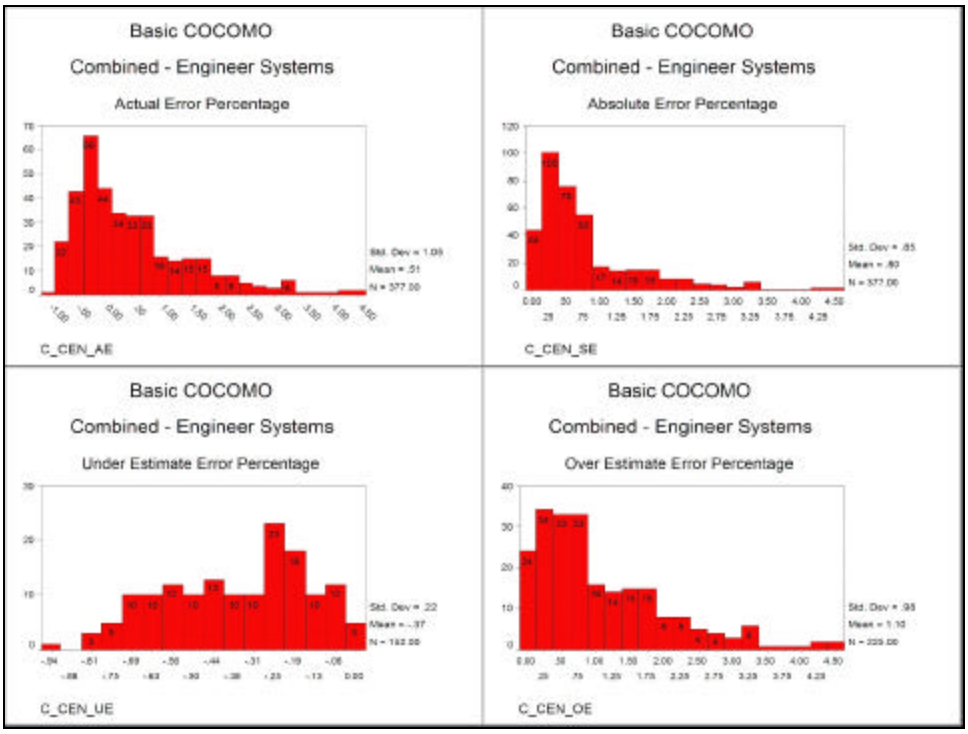



**Figure VII-10. COCOMO - Engineer Results.**

The Basic COCOMO ranked 3rd in overall model performance for *engineer* applications:

- Actual Error – The mean error is 0.51 or an average of 51% fom the actual value. The average error has a standard deviation of 105%. One hundred forty-four projects, or 38%, are projected within 25% of the actual value. The Basic COCOMO ranked 3 of 3 for average actual error.
- Absolute Error – The Basic COCOMO projected the actual project performance with an absolute error of 80% ± 85%. The Basic COCOMO ranked 2 of 3 for average absolute error.
- Under Estimate – The Basic COCOMO projected 152 out of 377 projects under the actual value, 40%. When the Basic COCOMO projects short, it does so with an average error of 37%. The majority (73%) of the under estimates occur between zero and 50 percent. The Basic COCOMO ranked 2 of 3 for balance and 1 of 3 for average under-estimation error.
- Over Estimation – The Basic COCOMO over estimated 225 projects. Forty percent of the over-estimates were between zero and 50 percent. The Basic COCOMO ranked 2 of 3 for average over-estimation error.

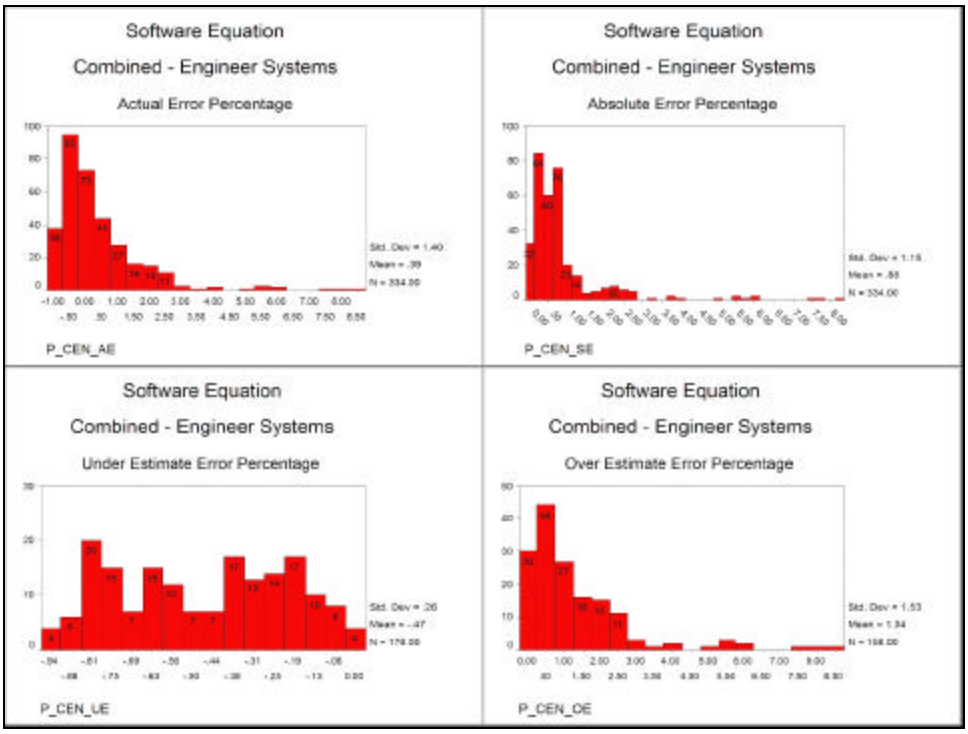



**Figure VII-11. SSE - Engineer Results.**

The Simplified Software Equation ranked 2nd in overall model performance for *engineer* applications:

- Actual Error – The mean error is 0.39 or an average of 39% from the actual value. The average error has a standard deviation of 140%. One hundred sixteen projects, or 35%, are projected within 25% of the actual value. The SSE ranked 1 of 3 for average actual error.
- Absolute Error – The SSE projected the actual project performance with an absolute error of 88% ± 115%. The SSE ranked 3 of 3 for average absolute error.
- Under Estimate – The SSE projected 176 out of 374 projects under their actual value, 47%. When the SSE projects short, it does so with an average error of 47%. The majority (55%) of the under estimates occur between zero and 50 percent. The SSE ranked 1 of 3 for balance and 3 of 3 for average under-estimation error.
- Over Estimation – The SSE over estimated 158 projects. Forty-six percent of the over-estimates were between zero and 50 percent. The SSE ranked third for average over-estimation error.

### 3. Informational Applications

*Informational* application is the family of applications comprised of business systems. Business systems comprise over 75% of the total project population in the project subset. Table VII-6 summarizes the performance of the three models followed by quad charts in Figure VII-12, Figure VII-13, and Figure VII-14.

Figure VII-2 details that a functional complexity in the range of 0.0 to 2.0 should be implemented for informational applications. The MRM validation implements a functional complexity value of zero for all of the informational applications (FC of 0.0). The informational family of applications is comparable to the COCOMO notion of an *organic* mode. For all of the Basic COCOMO projections the organic mode is utilized. The Simplified Software Equation implements its specialized productivity parameters for business systems (PP of 28657).

# Informational Applications

| Informational Applications | | | | | Actual Error (wt 5) | | | | Absolute Error (wt 4) | | | | Balance (wt 3) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Consolidated | N | 5% | Sigma | Correl | Mean | Std | Rank | Score | Mean | Std | Rank | Score | % Under | Rank | Score |
| MRM | 1470 | 74 | 329 | 0.6360 | 69.07% | 127.41% | 3 | 15 | 104.31% | 100.60% | 3 | 12 | 30.95% | 2 | 6 |
| COCOMO | 1470 | 74 | 329 | 0.6381 | 38.52% | 91.98% | 1 | 5 | 70.18% | 70.83% | 2 | 8 | 42.62% | 1 | 3 |
| SSE | 1470 | 74 | 329 | 0.6525 | -57.28% | 39.18% | 2 | 10 | 64.43% | 25.78% | 1 | 4 | 89.36% | 3 | 9 |
| | | | | | Under Estimation (wt 2) | | | | | Over Estimation (wt 1) | | | | | |
| SSE required | 240 | | | | N-Under | Mean | Std | Rank | Score | N-Over | Mean | Std | Rank | Score | Total |
| projects removed. | | | | MRM | 432 | -56.93% | 62.57% | 2 | 4 | 964 | 125.54% | 107.00% | 3 | 3 | 40 |
| | | | | COCOMO | 595 | -37.15% | 22.79% | 1 | 2 | 801 | 94.72% | 83.35% | 2 | 2 | 20 |
| | | | | SSE | 1033 | -68.10% | 22.98% | 3 | 6 | 123 | 33.58% | 27.37% | 1 | 1 | 30 |

**Table VII-6. Informational Application Data.**

*Informational* applications are comprised of 1470 projects; all from business applications. Each model had a total of 74 projects removed from the validation. The average standard deviation is 329 man-months. Each model has a correlation coefficient of around 60%. A total of 240 projects were removed from consideration for the Simplified Software Equation.

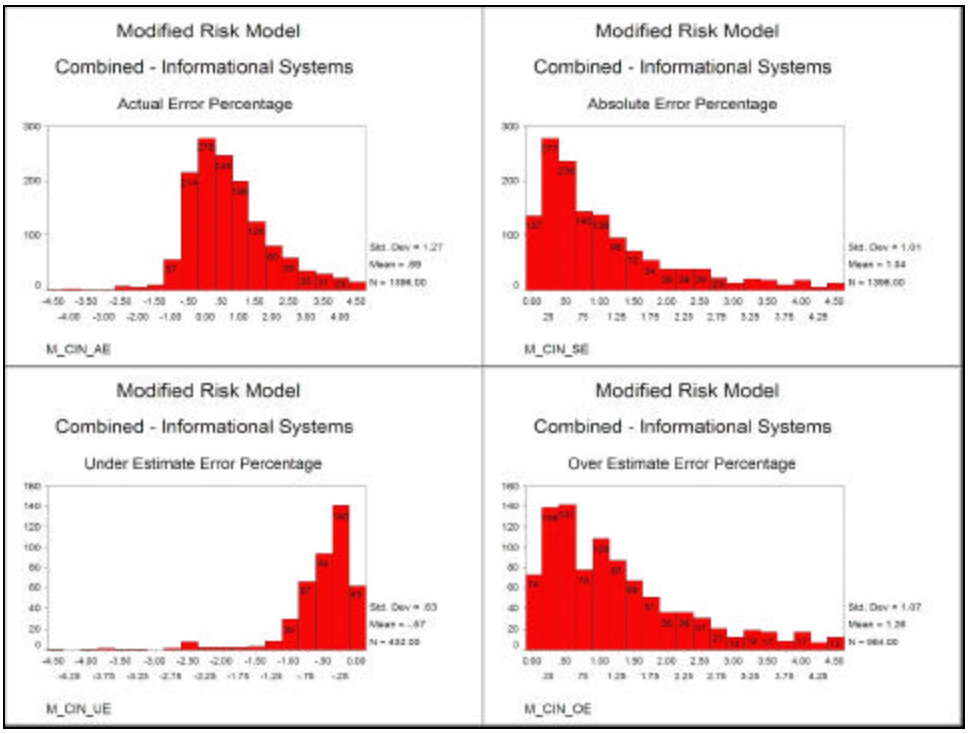



**Figure VII-12. MRM - Informational Results.**

The Modified Risk Model projected 19.13% of all of the *informational* applications within 25% of the actual value. The MRM ranked 3rd in overall model performance for *informational* applications:

- Actual Error – The mean error is 0.69 or an average of 69% from the actual value. The average error has a standard deviation of 127%. Four hundred fifteen projects, or 30%, are projected within 25% of the actual value. The MRM ranked 3 of 3 for average actual error.
- Absolute Error – The MRM projected the actual project performance with an absolute error of 104% ± 101%. The MRM ranked 3 of 3 for average absolute error.
- Under Estimate – The MRM projected 432 out of 1396 projects under the actual value, 31%. When the MRM projects short, it does so with an average error of 57%. The majority (69%) of the under estimates occur between zero and 50 percent. The MRM ranked 2 of 3 for balance and 2 of 3 for average under-estimation error.

- Over Estimation – The MRM over estimated 964 projects. Thirty-six percent of the over-estimates were between zero and 50 percent. The MRM ranked 3 of 3 for average over-estimation error.

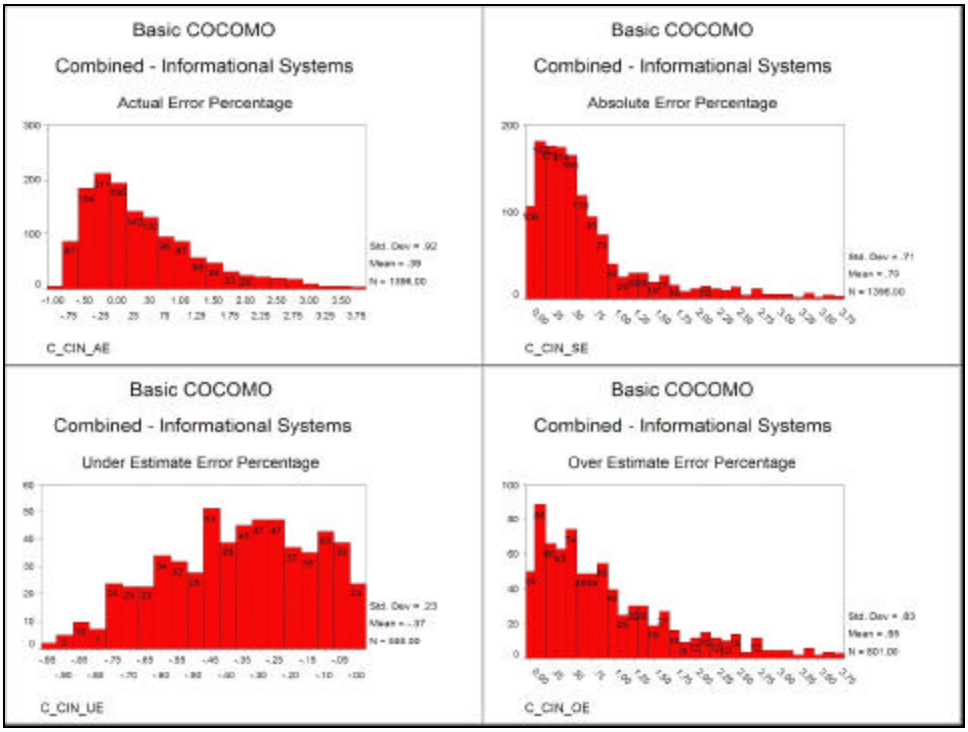



**Figure VII-13. COCOMO - Informational Results.**

The Basic COCOMO ranked 1st in overall model performance for *informational* applications:

- Actual Error – The mean error is 0.39 or an average of 39% from the actual value. The average error has a standard deviation of 92%. Four hundred twenty-nine projects, or 31%, are projected within 25% of the actual value. The Basic COCOMO ranked 1 of 3 for average actual error.
- Absolute Error – The Basic COCOMO projected the actual project performance with an absolute error of 70% ± 71%. The Basic COCOMO ranked 2 of 3 for average absolute error.
- Under Estimate – The Basic COCOMO projected 595 out of 1396 projects under their actual value, 43%. When the Basic COCOMO projects short, it does so with an average error of 37%. The majority (66%) of the under estimates occur between zero and 50 percent. The Basic COCOMO ranked 1 of 3 for balance and 1 of 3 for average under-estimation error.

- Over Estimation – The Basic COCOMO over estimated 801 projects. Forty-three percent of the over-estimates were between zero and 50 percent. The Basic COCOMO ranked 2 of 3 for average over-estimation error.

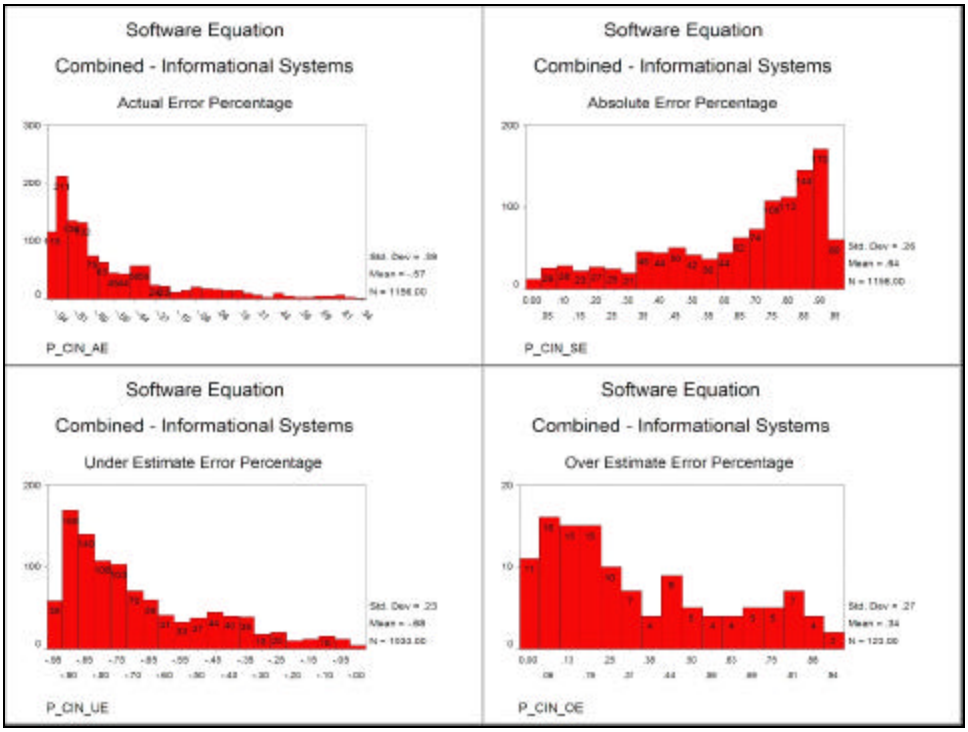



**Figure VII-14. SSE - Informational Results.**

The Simplified Software Equation ranked 2nd in overall model performance for *informational* applications:

- Actual Error – The mean error is 0.57 or an average of 57% from the actual value. The average error has a standard deviation of 39%. One hundred thirty-nine projects, or 12%, are projected within 25% of the actual value. The SSE ranked 2 of 3 for average actual error.
- Absolute Error – The SSE projected the actual project performance with an absolute error of 64% ± 26%. The SSE ranked 1 of 3 for average absolute error.
- Under Estimate – The SSE projected 1033 out of 1156 projects under the actual value, 89%. When the SSE projects short, it does so with an average error of 68%. A little under 25% of the under estimates occur

between zero and 50 percent. The SSE ranked 3 of 3 for balance and tied at 3 of 3 for average under-estimation error.

- Over Estimation – The SSE over estimated 123 projects. Seventy-five percent of the over-estimates were between zero and 50 percent. The SSE ranked 1 of 3 for average over-estimation error.

## E. VALIDATION CONCLUSION

The Modified Risk Model projected the required effort required to develop software projects with less error than Basic COCOMO and the Simplified Software Equation for *Real Time* and *Engineer* applications. The MRM did not perform the strongest on *Informational* (business) type applications. Table VII-7 demonstrates the overall accuracy of the MRM for each application type, Appendix F provides additional detail.

| | | Modified Risk Model | | | |
|---|---|---|---|---|---|
| | Application Type | # projected within 25% | total projected | accuracy | |
| Real Time | Real Time | 12 | 38 | 31.58% | 36.24% |
| | Avionic | 9 | 22 | 40.91% | |
| Engineering | Command & Control | 13 | 53 | 24.53% | 25.35% |
| | Process Control | 8 | 27 | 29.63% | |
| | Telecom | 36 | 128 | 28.13% | |
| | Systems Software | 19 | 75 | 25.33% | |
| | Scientific Software | 18 | 94 | 19.15% | |
| Informational | Business Systems | 267 | 1396 | 19.13% | 19.13% |

**Table VII-7. Accuracy of the MRM**

Table VII-7 provides a summary of how many applications were projected within 25% of the actual required effort. The validation summary of the MRM is established accordingly. The "# projected within 25%" column indicates the number of projects that were estimated within 25% of the actual value. Following the "total projected" column, the accuracy (i.e. percent of projects estimated within 25%) is provided.

The MRM projects within 25% of the project actuals 36% of the time for *real time* applications and 25% of project actual for *engineering* applications. As previously indicated in Table VII-6, the performance on *informational* applications trails slightly at 19%. The MRM is capable of projecting the required effort of software projects with greater accuracy than documented in (Boeh81) for the Basic COCOMO model.

The Basic COCOMO performs well on the low complexity systems, but not the higher complexity systems. The following is a quote from Lawrence Putnam, Sr. that provides some insights to help explain the reasons why:

> Basic COCOMO is very old. It has no provisions for update so it becomes further and further away from current practice as time goes on and productivity within the community improves. We did something like you are attempting using the data from Boehm's book. We put it in SLIM (or Metrics today) and then compared it with our trend lines, which are based on current data with only very modest time lag (a couple of years at most). We compared it with trends and data from the time of Boehm's book (circa 1980) and found that the effort was approximately the same as we were finding, but that the schedules were dangerously short. Later, when we used trend lines that showed the productivity improvement we found Boehm's data (and the COCOMO algorithms based on that data) were showing much greater effort than the industry trends but the schedules were coming into line with the trend lines. The last one we did (circa 1995) showed approximately 6x cost penalty at the proper schedule by using Basic COCOMO. Lesson: Any estimating system to be useful has to able to calibrated and tuned to current practice. SLIM can; COCOMO can't (or is hard with COCOMO II).

The Simplified Software Equation performed modestly for <u>all</u> application domains and exceptionally well for the *engineer* systems. For this analysis, the productivity parameters seemed too high for the business applications. This seems to be due to the overwhelming number of under estimates. Conversely, the productivity parameters for the high complexity systems seem to be too low; this is evident in the excessive shift in the power equation detailed in Appendix F.

Due to the SSE projecting the required effort with the most error, QSM® was consulted about the values utilized from the tables (Putn92) for the productivity parameters. QSM® suggested four key points that must be understood about utilizing the SSE.

- conservative / minimum time estimates were used in the simple equations
- the productivity parameter has a tendency to change with increasing E-SLOC (a fact not captured by the simple equations but is captured in the QSM®'s SLIM software)
- the accuracy-parametric inputs trade-off, essentially accuracy is a function of available data.
- The use of a non-calibrated productivity parameter. The tables in (Putn92) can be considered industry standards for the year of the publication. However, using the productivity parameter from a table does not provide the fidelity that can be obtained using the SLIM software (or even manually calculating the productivity parameter for each organization).

THIS PAGE INTENTIONALLY LEFT BLANK

# VIII. CONCLUSION

> Nearly every software engineering development project is plagued with numerous problems leading to late delivery and cost overruns, and sometimes, unsatisfied customers. Often the problems are technical, but just as often the software engineering development problems are managerial. How often have we personally heard or read that a project was late (or over budget, or reduced in scope, or terminated early, or did not satisfy the user, …) because:
>
> - Programmers did not tell the truth (or did not know the truth) about the status of their programs
> - Top management did not allow sufficient time for upfront planning
> - The true status of the project was never known
> - Programmer productivity was lower than planned
> - The customer did not know what he wanted (or changed the requirements)
> - Government standards for requirement specification (or government policies) were not suitable for software
> - Or, or, or…

This quote from Richard Thayer is just as applicable today, if not more, than when he originally published it over twenty years ago in 1981 (Thay81). Has any progress been made? There is proof that research has. Over the last forty years, research and technology have successfully overcome critical barriers to the advancement of the software engineering. Illustrated in Figure VIII-1, computing power can be considered one of the earliest barriers that was overcome. Having the opportunity to access a desktop computer and having computers readily available soon joined computational power as issues of the past. The internet era of computing has spawned enormous success in the creation of efficient bandwidth and better networks. With all of the wonderful advances in technology, state-of-the-art <u>still</u> suffers from the inability to "…

predict how much a software system will cost, when it will be operational, and whether or not it will satisfy user requirements …" (Mose02).

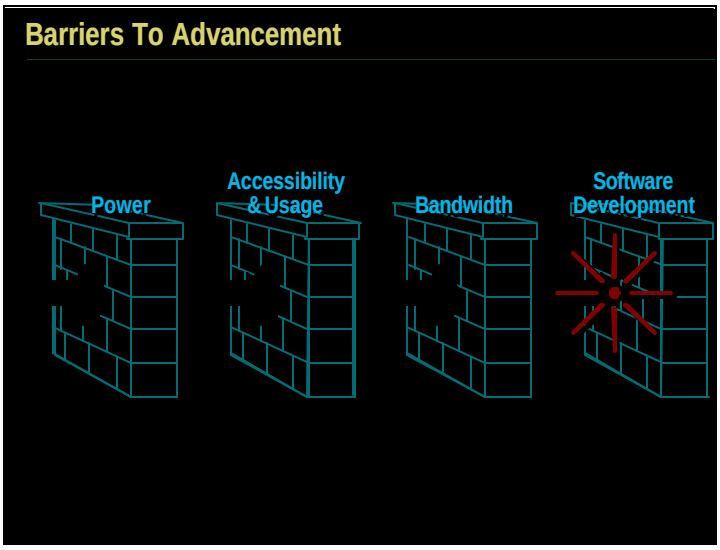



**Figure VIII-1. Barriers To Advance.**

This dissertation advances the state-of-the art in software estimation. Significant enhancements are introduced to the formal risk assessment model for software projects developed by (Nogu00). The research contained within validates the assertion that successful software risk assessment is achievable from a set of simple and easily derived artifacts. These artifacts can be automatically collected and are not subjective to an analyst's bias.

## A. ORIGINAL RESEARCH QUESTIONS

Chapter I of this dissertation presented three research questions that have provided a beacon for the development of this ongoing research. This dissertation addresses these questions and provides substantial supporting evidence as to the answers. The following revisits the research questions:

- **How do the metrics required for the current risk estimation methods correlate to metrics collected in real world projects?**

Validating the SRM proved challenging due to the unique nature of its required input parameters. Chapters IV and Appendix C provide 27 mitigation techniques to successfully interface the SRM with real project data. Interfacing the SRM model to actual project data provided invaluable insight to the model enhancement.

- **How do the current risk estimation methods perform when exercised on real projects?**

With the help of a successful interface between the SRM and the initial project subset, the SRM was capable of projecting the project durations. Unfortunately, the SRM projected the project durations dangerously optimistic, with a discrete and bi-polar contour. The excessively short projections required extending the validation to include models from commercial best practice; increasing the confidence in the overall validation process.

- **What are the necessary enhancements evolving the current risk estimation methods to provide improved results when exercised on real projects?**

The SRM projects the software project durations with more error than the macro models of the Basic COCOMO and the Simplified Software Equation. In fact there was so much error, that resources were refocused during this dissertation to discover the root causes. What was discovered changed the whole development course of the Software Risk Model.

Technological advances afforded the research the ability to conduct simulations in quantities never before achieved, documented in Chapter V and Appendix D. Utilizing this technology, hundreds of thousands of experiments with VitéProject were analyzed. Analysis of the performance data from actual software development activities provided a

unique technique to calibrate the VitéProject input parameters. Ultimately, our analysis with the VitéProject simulation uncovered a fallacy with VitéProject that suggested the simulation is not suitable for this purpose.

This research demonstrates, that alone, the original input parameters of the SRM are not ideally suited for software risk assessment (Chapter VI and Appendix E). A hybrid analysis of simulation and actual project data led to redefining the complexity representation. Chapter VI contains the new definition for both a *functional complexity* and a *functional size* measure.

Real projects do not behave with the discrete properties portrayed by the SRM. The MRM was developed to provide a true and continuous representation of software project development. To adequately represent the software behavior, the MRM is calibrated to project a software project's required *effort*.

The MRM projects within 25% of the project actuals 36% of the time for *real time* applications and 25% of project actual for *engineering* applications. As indicated in Table VII-6, the performance on *informational* applications trails slightly at 19%. The MRM is capable of projecting the required effort of software projects with greater accuracy than documented in (Boeh81) for the Basic COCOMO model.

## B. AREA FOR FUTURE WORK

The MRM is not a panacea. Software engineering and software risk assessment are a long way from making the opening quote from Richard Thayer obsolete. Although the MRM provides a suitable mitigation to approaching software development, there exist several areas where future work is warranted.

- The model must continue to be validated. The validation is conducted against 2,000 projects. However, these 2,000 projects were comprised of ten application types. Increasing the numbers of projects in each application type could serve two purposes, (1) provide additional data points to project trend lines, and (2) provide additional data points to test the different models.
- Due to resource constraints, this research did not challenge (Nogu00)'s interpretation of the organizational *efficiency* or the *requirements*

*volatility.* The collection techniques of these two parameter should be re-examined.

- The complexity table introduced in Chapter VI will have to be kept updated. This is not a frequent task or one that will provide dramatic differences. However, it is very reasonable to appreciate technologies change and software developers will always provide better techniques and practices to develop software solutions. Complex software today may or may not be complex tomorrow. However, a model that does not have a capability to extend and remain current minimizes its long-term sustainability.
- A PSDL parser must be developed to implement the work in (Dupo02). The parser, or direct implementation into CAPS, will provide an automated means to derive the complexity of the software prototype.
- Additional projects must be developed in the CAPS environment. The available data points from completed software project are very limited. Additional projects would permit the continued development of the complexity measure in (Dupo02) and provide empirical evidence to develop the translation of the PSDL complexity measure to the MRM functional complexity.
- The MRM, or the preceding SRM, has yet to be implemented on a project a priori. All model validations are conducted on post-mortem projects.

## C. CONTRIBUTIONS OF THE DISSERTATION

This body of research makes the following advancements to the software engineering state-of-the-art.

- A validated software risk model. This is the primary research and has been achieved.
- Interface techniques between the input parameters: *efficiency*, *requirements*, *functional complexity*, and *functional size*. Extensive analysis is documented to provide suitable procedures to interface with information frequently collected in actual project development.
- A technique to calibrate VitéProject to represent software development. Previously, VitéProject had never been calibrated to project software development. This research provides techniques to utilize a calibrated VitéProject for software estimations.
- Baseline software development trends. This research provides a recommended baseline to extend or develop any future software development models; or to tune additional simulations.

- Overwhelming evidence that VitéProject, configured according to (Nogu00), (Alex01), and (Murr02), is not suitable for software development; due to a flaw in the VitéProject simulation. This research is not prepared to claim that VitéProject does not perform suitably in any situation. Resources do not permit exhaustive operational tests of VitéProject's configuration parameters.
- Evidence to suggest the SRM, in its original form, is not suitable to project software risk assessment. This research documents, unequivocally, the mathematical implementation of the SRM is not correct. The SRM was built and validated with a single point of failure; a simulator that provides discontinuous projections. However, this research does support the theoretical basis of the SRM; that software risk assessment can be achieved with analysis of a small set of easily obtainable, mathematically quantifiable, software metrics.

# A. DATA DICTIONARY

This appendix details the measures that are available in the Quantitative Software Management (QSM®) database. Each of the collected attributes was considered and those deemed applicable were used in the research. Corporations contributing to the database, have the opportunity to work with a set of default measures, as well as provide information specialized to their particular needs. The following present the default measure and the definition of each attribute. This data dictionary is taken directly from a QSM® publication (QSM98).

## A. BASIC INFORMATION

**Project Name.** System name or project title.

**Status**. Indication of the current state of data entered for the project; users can choose one of the following:

*Estimate* – The collected data are estimated.
*In Progress* – The collected data contain partial actuals.
*Completed* – The collected data contains final actuals.

**Confidence.** Label that best describes the level of confidence in the accuracy and consistency of this project's data, determined from one of the following:

*Low* – Low Confidence (error > 10%)
*Moderate* – Moderate Confidence (5% < error < 10%)
*High* – High Confidence (error < 5%)

**Record Creation Date.** Date this project data was first entered.

**Last Modified.** Date this project data was last modified. *Note, in the extract of the database used in the research, date of last modified is determined by the date the company names were removed.*

**Predominant Application Type and Sub-type**

Label that best describes the application type most heavily represented in this system. There are nine primary application types and one or more subtypes within each of these nine types. First find the primary type that coincides with your typical project, according to the following descriptions.

**Microcode & Firmware.** Software that is either the architecture of a new piece of hardware or software that is burned into silicon and delivered as part of a hardware product. This software is the most complex because it must be compact, efficient, and extremely reliable.

**Real Time.** Software that must operate close to the processing limits of the CPU. This is interrupt driven software and is often written in C, Ada or Assembly language. Typical examples are military systems like radar, signal processors, missile guidance systems, etc.

**Avionics.** Software that is on-board and controls the flight and operation of the aircraft.

**System Software.** Layers of software that sit between the hardware and applications programs. Examples are operating systems (DOS, UNIX, VMS, etc.), GUI's (graphical user interfaces – Windows, Xwindows etc.), Executives or Database Management systems, Network products, and Image processing products.

**Command & Control.** Software that allows humans to mange a dynamic situation and respond in human real-time. Examples are battlefield command

systems, telephone network control systems, government disaster response systems, military intelligence systems, electric utility power control systems.

**Telecommunications.** Software that facilitates the transmission of information from one physical location to another. Examples are telephone switches, transmission systems, modem communication products, fax communication products, satellite communications products.

**Scientific.** Software that involves significant computations and analysis. Examples are statistical analysis systems, graphics products, data reduction systems.

**Process Control.** Software that controls an automated system. Examples are software that runs a nuclear power plant, or software that runs a petro-chemical plant.

**Business.** Software that automates a common business function. Examples are payroll, personnel, order entry, inventory, materials handling, and warranty products.

**Description**. Overview of the system under development.

**Size.** The function unit selected for sizing. If this is a new project, the companies default size will appear, otherwise, the primary unit will be the first unit for which sizing data has been entered.

**New.** Portion, measured in the selected size units, of the total system size that was developed for this application (designed, coded, integrated, and tested from scratch).

**Modified.** Portion, measured in the select size units, of the total system size contained in pre-existing entities (modules, packages, etc.) that were modified (pre-existing software requirement changes).

**Unmodified.** Portion, measure in the selected size units above, of the total system size contained in pre-existing entities (modules, packages, etc.) that were incorporated into this product unchanged (pre-existing software requirement no change)

**Requirements.** Total number of requirement defined during the Critical Design Review.

**Defects System Integration to Delivery**. Total number of defects found between System Integration Test (SIT) and the completion of the main build. SIT is a QSM®-defined milestone that represents the point at which a configuration item (e.g., package, compilation unit) is placed under change control and is ready to be integrated into the evolving system.

**Defects First Month after Delivery.** Total number of defects found during the first month following Full Operational Capability (FOC). FOC is a QSM®-defined milestone that represents full functionality with 95% reliability.

**Life Cycle Values.** The following information is provided for each phase of development.

**Phase Acronym.** The acronym, if any, that the project uses to refer to the activities of the associated life cycle phase.

**Start Date.** The date on which effort was first applied to any activity of the associated life cycle phase.

**End Date.** The date on which all activities of the associated life cycle phase were completed.

**Months.** The elapsed number of months for this phase.

**Effort.** Total amount of effort in the unit specified of labor to complete all activities of the associated life cycle phase. Includes all development staff: analysts, designers, programmers, coders, integration and test-team members, quality assurance, documentation, supervision, and management.

**Cost.** Total cost, in thousands of monetary units, to complete all activities of the associated life cycle phase.

**Peak Staff.** Peak number of full time equivalent people used to complete all activities of the associated life cycle phase.

**Staffing Shape-Phases 1, 2, 3.** Label that best describes the shape of the staffing curve for the associated life cycle phase:

*Unknown* – Valid for any of the four life cycle phases.

*Level Load* – A staffing plan that has the same number of people, or essentially the same number, from beginning to end. Usually found in the development of small systems.

*Front Load Rayleigh* – The front load Rayleigh shape will peak about 40% of the way through the phase and then taper down.

*Medium Front Load Rayleigh.* The medium front load Rayleigh profile will peak close to the middle of the phase and then start to taper down in staffing.

*Medium Rear Load Rayleigh.* The medium rear load Rayleigh peaks about three quarters of the way through the phase.

*Rear Load Rayleigh.* Peaks at the end of phase 3.

*Default Rayleigh.* A roughly bell shaped curve used to represent the ideal buildup and decline of manpower, effort, or cost, followed by a long tail representing manpower, effort, or cost devoted to enhancement or maintenance.

*Front Weibull.* The front load Weibull shape will peak about 25% of the way through the phase and then taper down.

*Normal.* The normal shape peaks at about midway through the phase and then tapers down.

*Rear Weibull.* The real load Weibull shape peaks about 70% of the way through the phase and then taper down.

*Custom*. Any other staffing pattern not described by the other options presented.

**Staffing Shape-Phase 4.** Label that best describes the shape of the staffing curve for the associated life cycle phase :

*Stair Step*. Staffing starts at about half of the staffing level at FOC and stair-steps down.

*Straight Line.* A straight-line decrease in staffing.

*Exponential.* One of the phase four staffing patterns, represented by the fastest tailing off of people from milestone seven to milestone nine.

*Rayleigh*. Tail-off of Rayleigh curve selected in phase three. Valid only if phase three staffing is a Rayleigh shape.

*Normal.* The normal shape peaks at about midway through the phase and then tapers down.

*Weibull.* Tail-off of Weibull curve selected in phase three.

*Custom.* Any other staffing pattern not described by the other options presented.

**Life Cycle Semantics.**

**Phase 1: Acronym FEAS - Feasibility Study.** The earliest phase in the software life cycle, where complete and consistent requirements and top-level, feasible plans for meeting them are developed.

**Phase 2: Acronym FUNC - Functional Design.** A phase of the software development process prior to phase three that develops a technically feasible, modular design with the scope of the system requirements.

**Phase 3: Acronym MB - Main Build.** A phase in the software development process that produces a working system that implements the system specifications and meets system requirements in terms of performance and reliability.

**Phase 4: Acronym MAINT - Maintenance.** The phase that usually coincides with the operations phase. It may include correcting errors that operations turn up and enhancing the system to accommodate new user needs, to adapt to environmental changes, and to accommodate new hardware.

## B. APPLICATION

**Organization**. The name of the company or organization entity that was responsible for the project. *Due to the sensitive nature of the contents of the database, company names have been removed.*

**Division**. The name of the organization's particular business unit that was responsible for the project.

**Country.** The name of the country in which the project was done.

**Design Complexity.** Label that best describes the complexity of the developed system (Low, Medium, High)

**Development Classification.** Label that best describes the type of development effort undertaken by the project.

*Brand New System* - > 75% new function
*Major Enhancement* – 25 to 75% new function
*Minor Enhancement* – 5 to 25% new function
*Conversion* - < 5% new function

**Industry Sector**

The Industry/Sector field is presented in a treed list. Table A-1 presents the sector description with an industry that best represents the majority of projects in your organization.

| Industry | Sectors |
|---|---|
| Academic | (General) |
| Aerospace | (General Aviation, Commercial, Military) |
| Construction | (General) |
| Distribution | (General) |
| Military | (General) |
| Electronics | (General) |
| Engineering | (General) |
| Financial | (General, Banking, Insurance) |
| Government | (General, National, Local, State, County) |
| Leisure | (General, Entertainment, Hotel) |
| Manufacturing | (General, Airplane, Automobile, Brewery, Chemical, Computer, Comp Peripherals, Image Processing, Medical, Pharmaceutical, Software, Telecom Equipment) |
| Health | (General) |
| Mining | (General, Metals, Gem, Coal) |
| Oil | (General) |
| Retail | (General, Food, Clothing) |
| Service | (General, Consulting) |
| Transportation | (General, Air, Bus, Rail/Metro, Sea, Trucking) |
| System Integrators | (General) |
| Utilities | (General, Gas, Electric, Telecom, Water) |
| Other | (General) |

**Table A-1. Industry Sectors.**

**Application Type**. The percentage of the total system size dedicated to the associated application type.

**Development Machine Type.** The general type of the host development machine.

**Development Machine Specific.** The specific name and version of the host development machine.

**Operating System Type.** The general type of the host operating system.

**Operating System Specific.** The specific name and version of the host operating system.

## C. SIZING

**Function Unit.** The name of a sizing unit used to size your software. Users can create a new sizing unit if an appropriate one is not available.

**Counting Method**. The method used for counting the associated sizing unit:

*Manual* – Physically counting the source code by hand

*Estimated* – An educated guess, perhaps based on analogy from past experiences.

*Sampled* - Using small representative sample statistics to forecast the complete system size.

*Code Counter* – An automated code counter.

**Gearing Factor.** The factor that, when multiplied by the total of units you will enter, yields the total system size in logical Source Lines of Code (SLOC).

**Language.** The name of a language used to code the system.

**% of Total Size.** The percentage of the total system size accounted for by the associated language.

**Language Type.** Select the language type that best describes the level of the associated language.

## D. ACCOUNTING

**Effort Units.** An effort unit allows you to select man-years, man months, man-days or man-hours. They have a first character designation of "M". You may select person years, person months, person days or person hours (designation "P"), or you may select from staff years, staff months, staff days, staff hours (designation "S"). There is no difference between a man-year, person year, or staff year. It just provides you the flexibility of selecting the gender and nomenclature that's most appropriate in your organization.

**Person Hours per Person Month.** The average number of hours a typical Full Time Equivalent (FTE) employee works in a 30.4375-day month allowing for vacation, holidays, and paid personal time off. The value should range between 120 and 744.

**Labor Rate.** The average wage and salary rate of the people directly involved in a software development, including detailed design, coding, testing, validation, documentation, supervision, and management, plus a percentage of the direct average to account for overhead.

**Labor Rate Unit.** The name of the effort unit associated with the given Labor Rate; choose one the following: MYR, MM, MWK, MDAY, MHR, PYR, PM, PWK, PDAY, PHR, SYR, SM, SWK, SDAY.

**Monetary Unit.** In the monetary units drop down panel, you may select a code for the currency to be used on this project.

**Conversion Factor (to $US).** The factor that, when multiplied by an amount expressed in the selected Monetary Units, yields an equivalent amount in United States dollars.

## E. ENVIRONMENT

**Development Environment.** The default development environment is where the specific environment in which your software was developed is defined. In most cases, this will be 5 days a week, 8 hours a day, 60 minutes per hour and 60 seconds per minute.

**Operational Environment.** The default operational environment is where you specify the environment in which your software will run.

## F. QUALITY

**Days / Week.** The number of days per week that the runtime environment was up and running.

**Hours / Day.** The number of hours per day that the runtime environment was up and running.

**Minutes / Hour.** The number of minutes per hour that the runtime environment was up and running.

**Seconds / Minute.** The number of seconds per minute that the runtime environment was up and running.

**Defect Categories.** The labels of the five defect categories.

**Cosmetic Defects.** The name that corresponds to QSM®'s cosmetic defects. Cosmetic defects can be described as deferred, such as errors in format of displays or printouts. They should be fixed for appearance reasons, but fix may be delayed until convenient.

**Tolerable Defects.** The name that corresponds to QSM®'s tolerable defects. Tolerable defects can be described as ones that do not affect the correctness of outputs; they should be fixed but may be delayed until convenient.

**Moderate Defects.** The name that corresponds to QSM®'s moderate defects. Moderate defects can be described as not critical to execution of the program but behavior is only partially correct. Should be fixed in the release.

**Critical Defects.** The name that corresponds to QSM®'s critical defects. Critical defects can be described as ones that prevent further execution; unrecoverable; they should be fixed before the program is used again.

## G. REVIEW (PROJECT OVERRUNS)

**Time (months).** Planned time in calendar months (or % difference from plan) for the specified phase. A planned time less than the actual time will result in a positive overrun %. A negative % overrun indicates early completion.

**Effort.** Planned labor in the indicated effort unit (or % difference from plan) for the specified phase.

**Cost.** Planned cost in thousands of the indicated monetary unit (or % difference from plan) for the specified phase.

**Peak Staff.** Planned peak staff in number of full time equivalent people (or % difference from plan) for the specified phase. (Calculated by dividing the total effort in person months by the total time in months for a given phase.)

**Size and Requirements Growth.** How much did the project change from the original plan, either the percentage of the original planned value or as the actual planned value?

**System Benefit / Effectiveness.** An indication of how beneficial or effective the final system was in solving the problem it was intended to solve. If the system was a commercial product, then the effectiveness should be interpreted as its commercial success.

**Significant Factors.** Describe any factors that had a positive or a negative impact on the project.

# B. PROJECT OVERVIEW

## A. OVERVIEW OF DATABASE

It is important for readers to understand the quality of the information obtained in the subset of projects available for conducting research. This appendix includes extracts from the initial set of 112 projects used in the research. The projects were chosen based on their similarities with other projects already exercised against the SRM (Nogu00), (John01). The following is an overview of the subset of projects and a limited number of extracts on the actual data. All project data analyzed in the research enjoyed either *high* or *medium* confidence, indicating that all projects contained less than 10% error.

Figure B-1 details how the total number of projects are categorized according to their application type. The majority of the projects are from the Avionics industry while the least number of projects represent Microcode and Real-time systems.

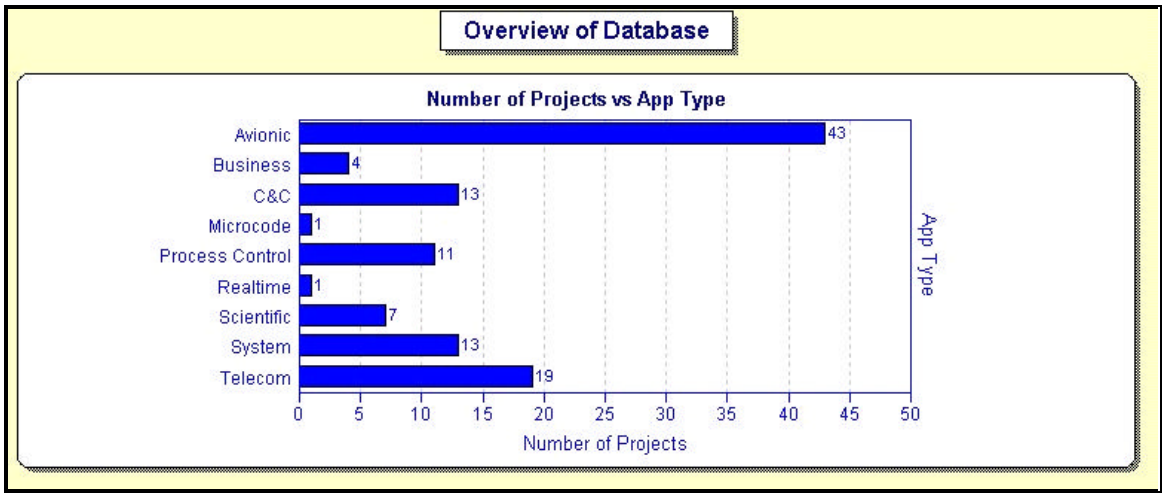



**Figure B-1. Projects vs. Application Type.**

In the subset of initial projects available to evaluate, there is a total of 26 different organizations. Figure B-2 displays the number of projects captured in the database by

each organization. In an effort to preserve proprietary information, a generic identifier has replaced organization names.

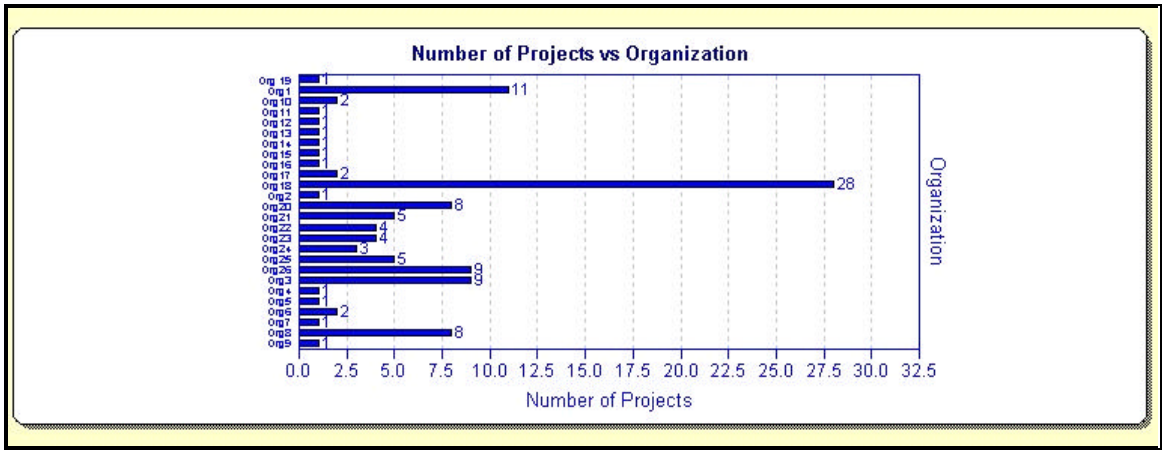



**Figure B-2. Number of Projects per Organization.**

Organizations represented in the database, use a metric called a "productivity index" or PI. The PI of an organization, for a particular project, is a management scale from one to 40, corresponding to the productivity parameter, that represents the overall process productivity achieved by an organization during the main build (Putn92). The PI is a measure that quantifies the net effect of everything that makes projects different from one another.

In the initial project subset, the majority of projects were developed by organizations with a PI between nine and eleven, see Figure B-3. Two projects have been developed by organizations with a very low PI and one project has been developed by an organization with a PI between twenty-two and twenty-three. It should be noted that an increase in one PI yields approximately a 10% reduction in schedule and approximately a 25 – 30% reduction in Effort. The typical improvement rate is approximately one PI per 2 – 2.5 years; similar to the normal time required moving up one level of the SEI CMM (QSMc).

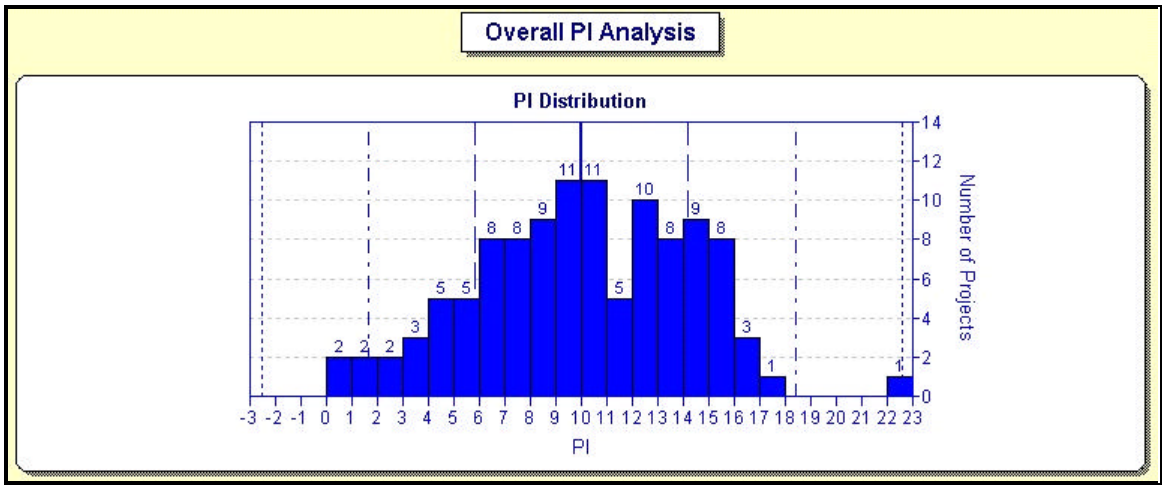



**Figure B-3. Overall PI Analysis.**

Figure B-4 illustrates the 26 organizations in the database and their corresponding PI, in terms of an average, minimum, and maximum. For example, Organization 8 developed projects with a minimum PI of approximately 4.9 and a maximum PI of approximately 22.9. The average PI recorded for Organization 8 is approximately 9.55.

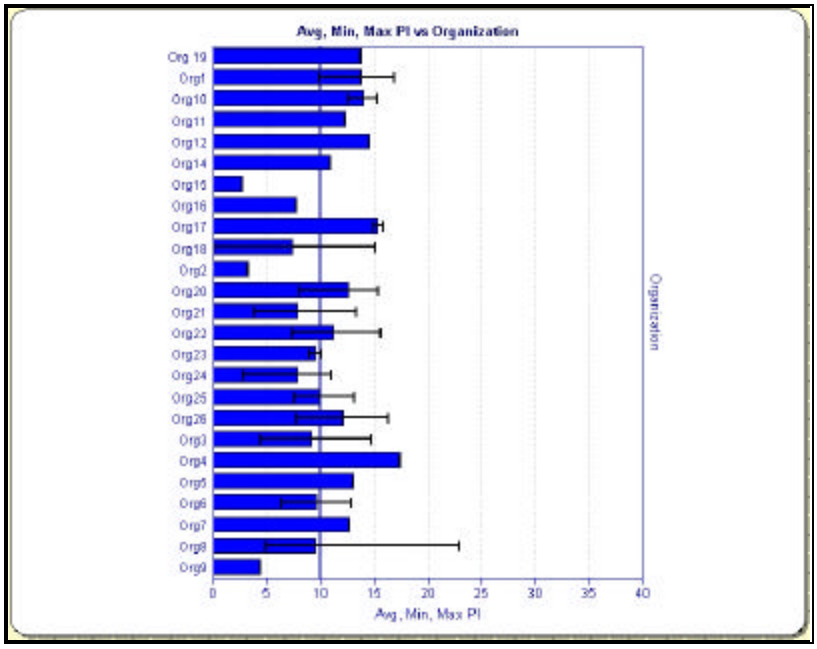



**Figure B-4. Organizational PI Distribution.**

In Figure B-5, an indication of the average staff months required during all four phases of project development is illustrated. In this view, the Main Build took an average of 270 staff months to complete.

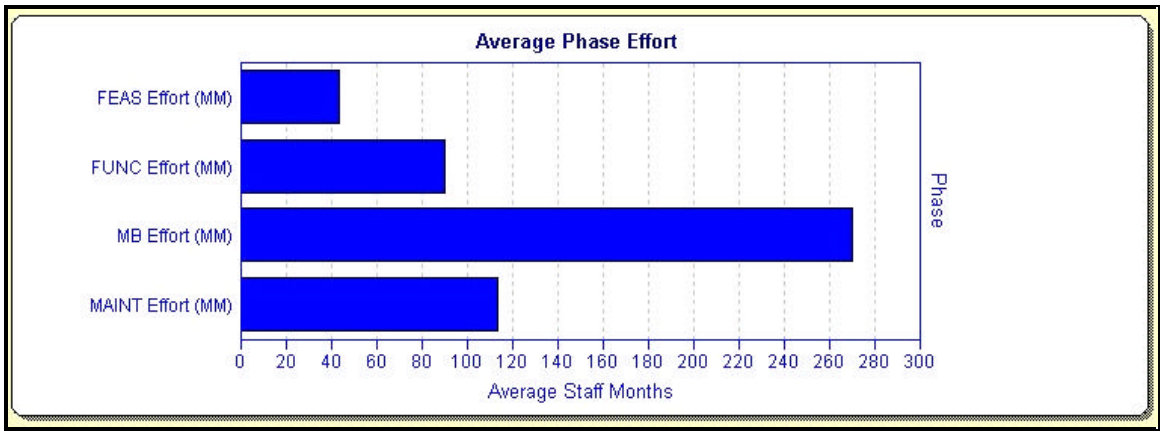



**Figure B-5. Average Phase Effort.**

To further clarify the effort expended to produce the software projects, Figure B-6 considers the actual elapsed calendar time to complete each software development phase.

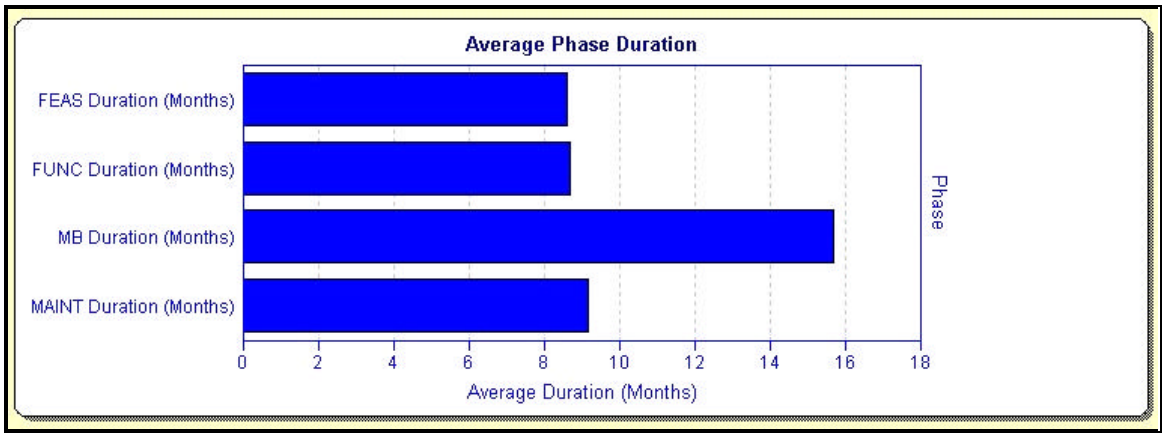



**Figure B-6. Average Phase Duration.**

Figure B-7 demonstrates four views to help understand trends in the size of the software projects under consideration. Without consideration for the staffing size, the top left chart compares the size of the projects (in source lines of code) against the elapsed

time constructing the project. The top right chart does consider the staffing size and completes a similar comparison between the project size and effort.

Staffing sizes are captured in the database by various staffing profiles. The bottom left chart expresses the project size against the peak staff while the bottom right chart expresses the project size against the average staff.

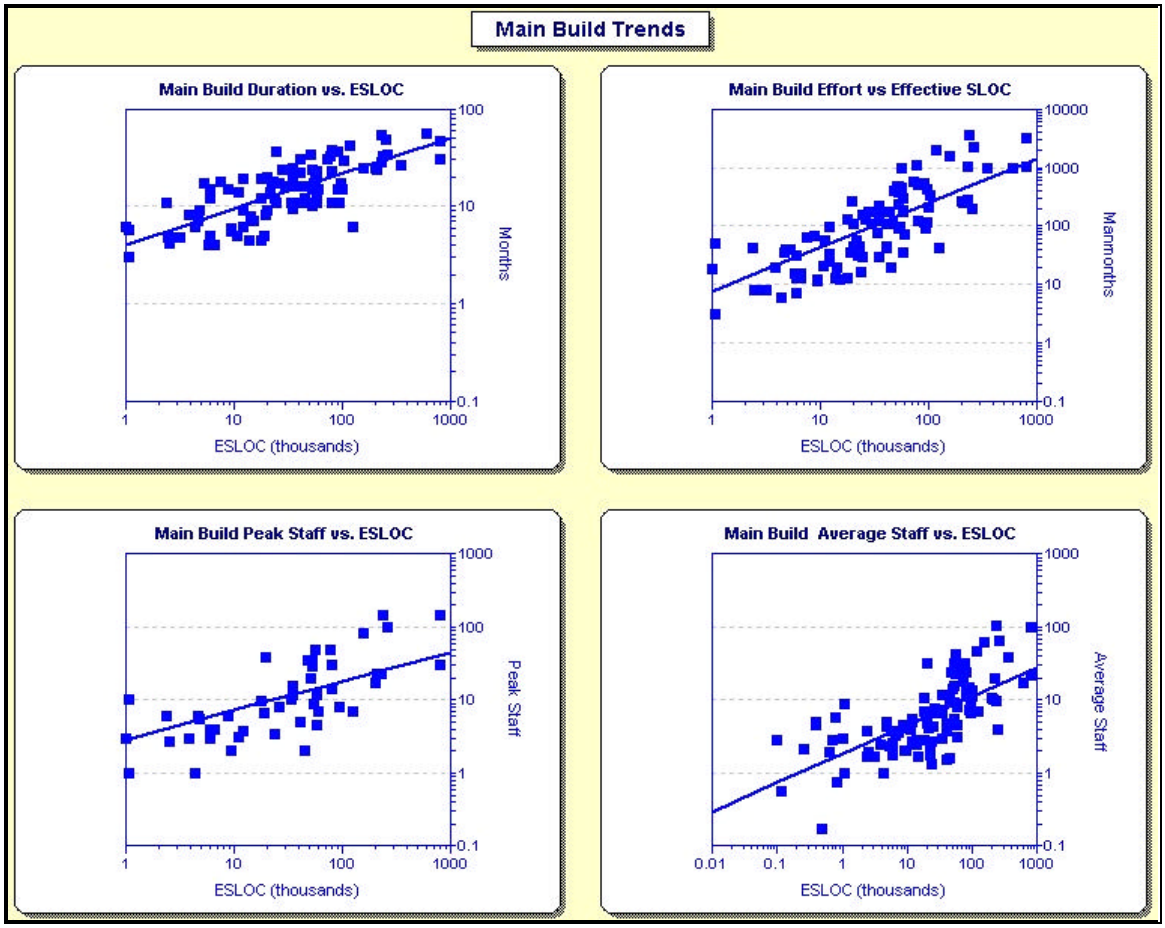



**Figure B-7. Main Build Trends.**

Schedule slippages are also captured in the database, Figure B-8. For example, during the main build, 33% of all of the projects exceeded their initial estimation by as much as 25%. Probably more alarming is the fact that out of the 112 projects under consideration, only 13% finished on time or earlier that their schedule estimates.

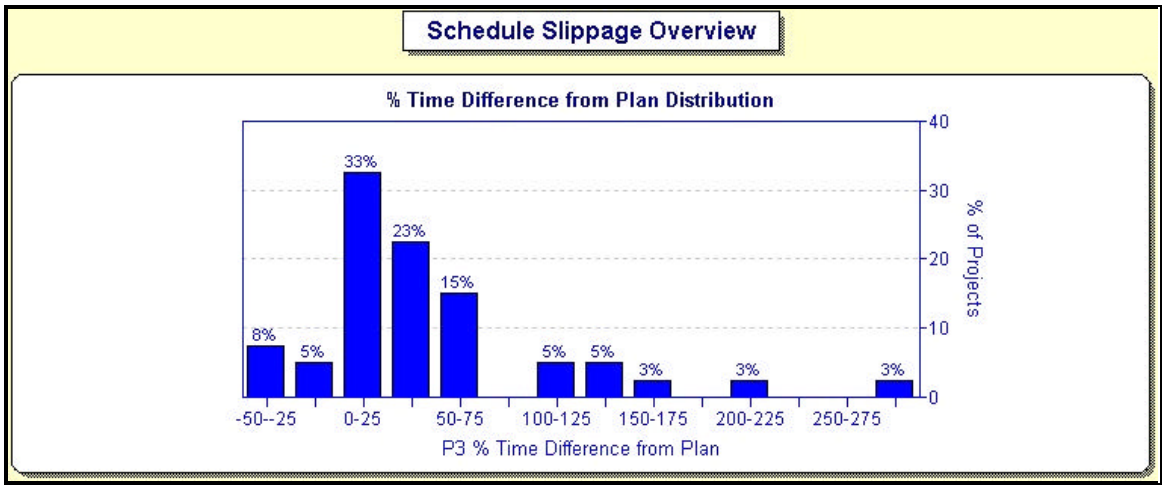



**Figure B-8. Schedule Slippage Overview.**

Figure B-9 illustrates that the entire subset of 112 software projects have Ada as their primary development language. Additionally, Figure B-9 provides the average productivity index, the average productivity in SLOC / main build month, and finally the average cost per E-SLOC.

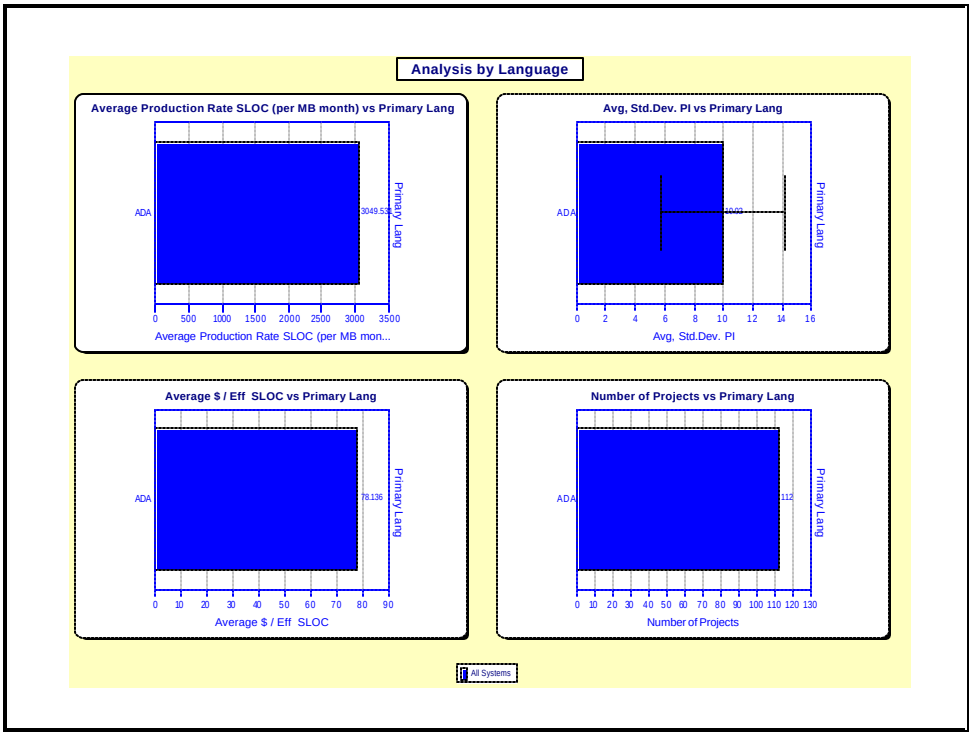
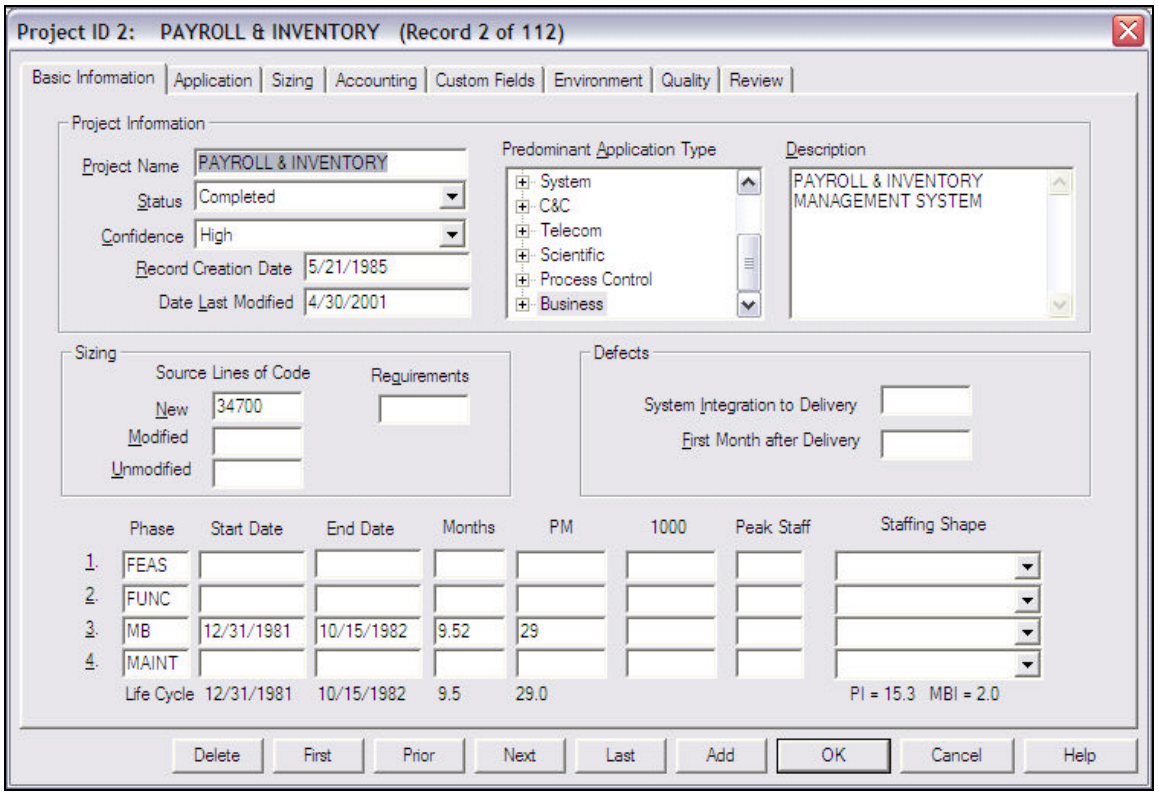


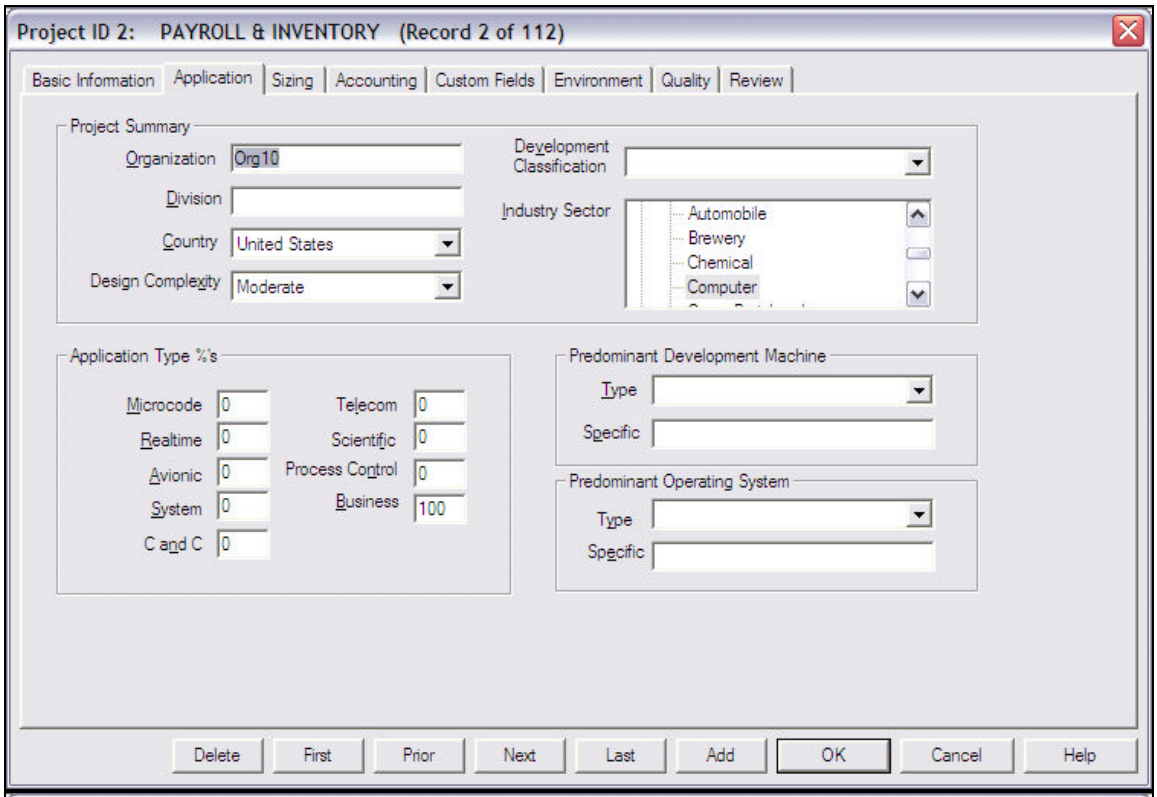

**Figure B-9. Analysis by Language.**

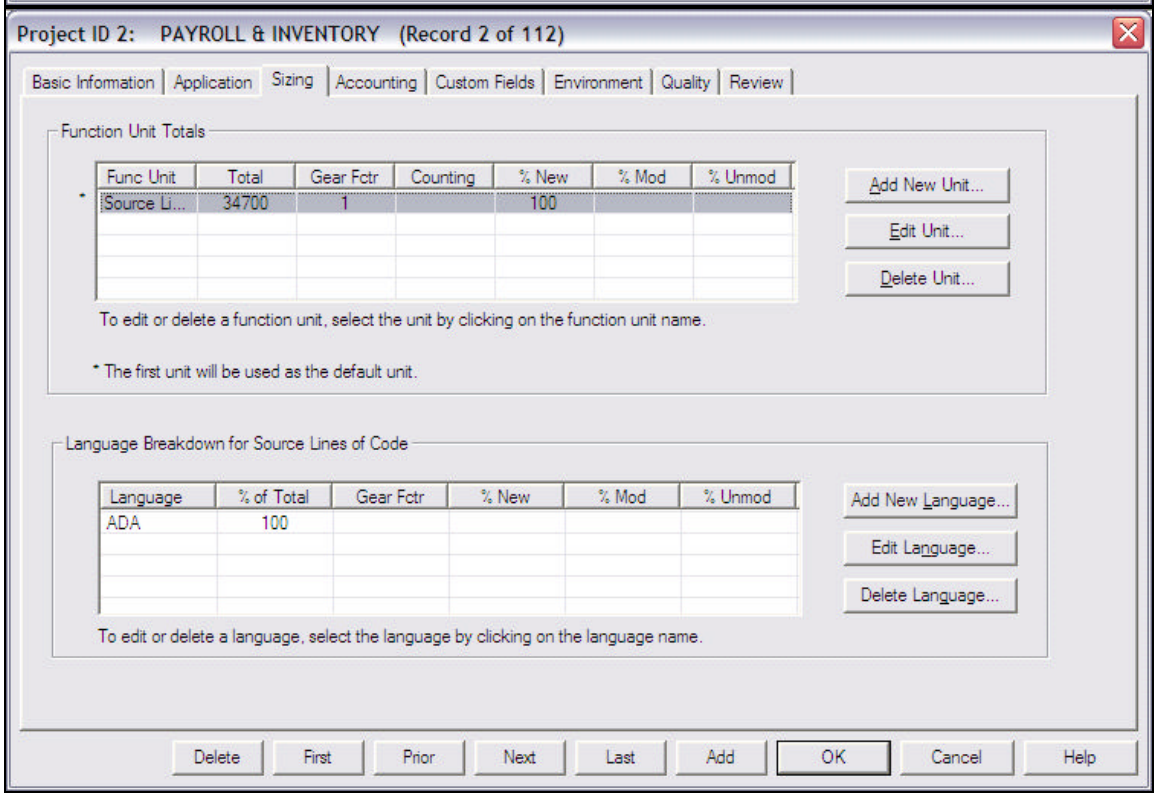
## B. SPECIFIC SOFTWARE PROJECT EXTRACTS

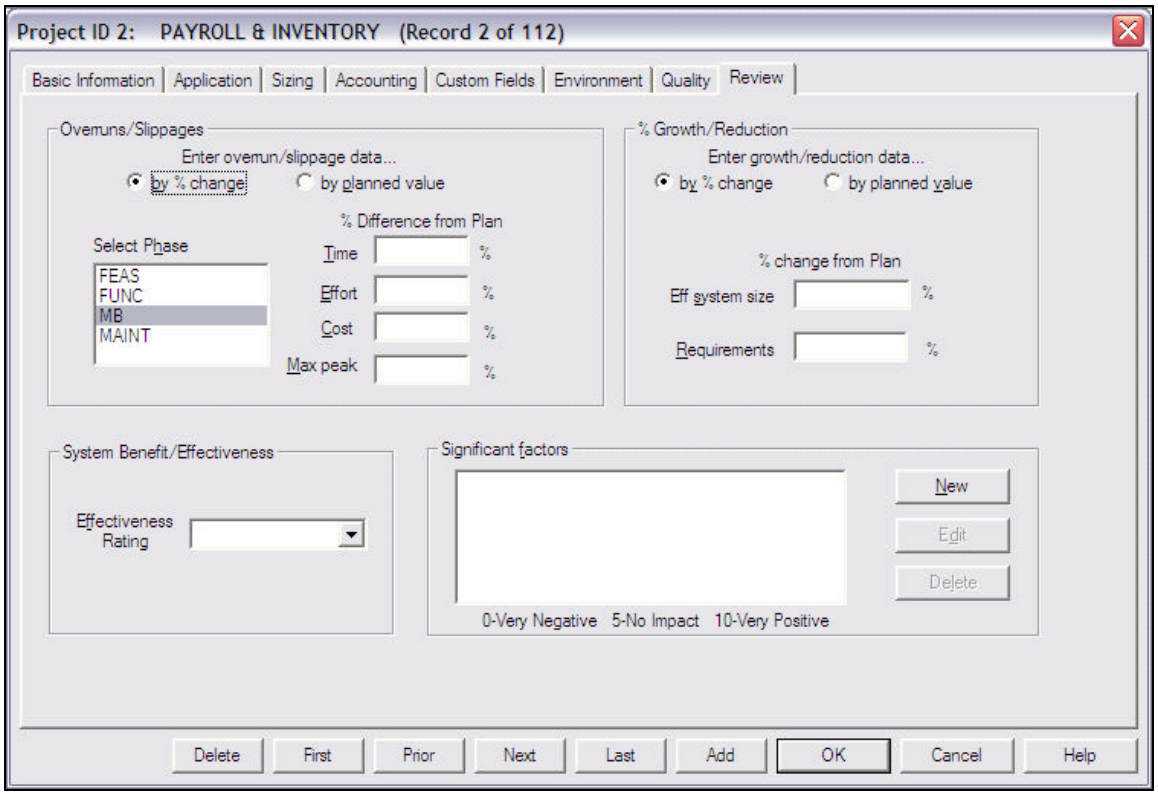
The next section illustrates some actual data as it is represented in the project database. Data is included from four projects. The difference in these four projects is the amount of information that is captured on each. The minimum set of data representing a project is data from the main build. The maximum set of data collected is data collected from all four phases of the project life cycle.

### 1. Main Build Documentation

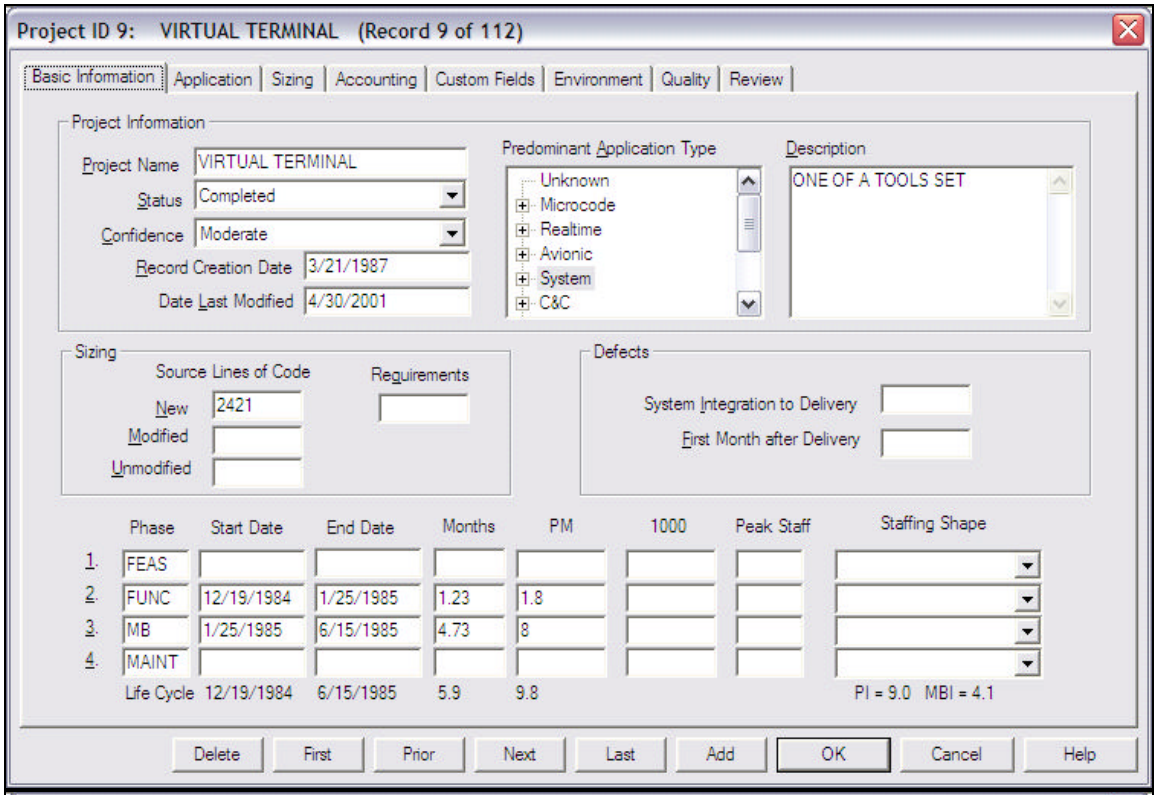


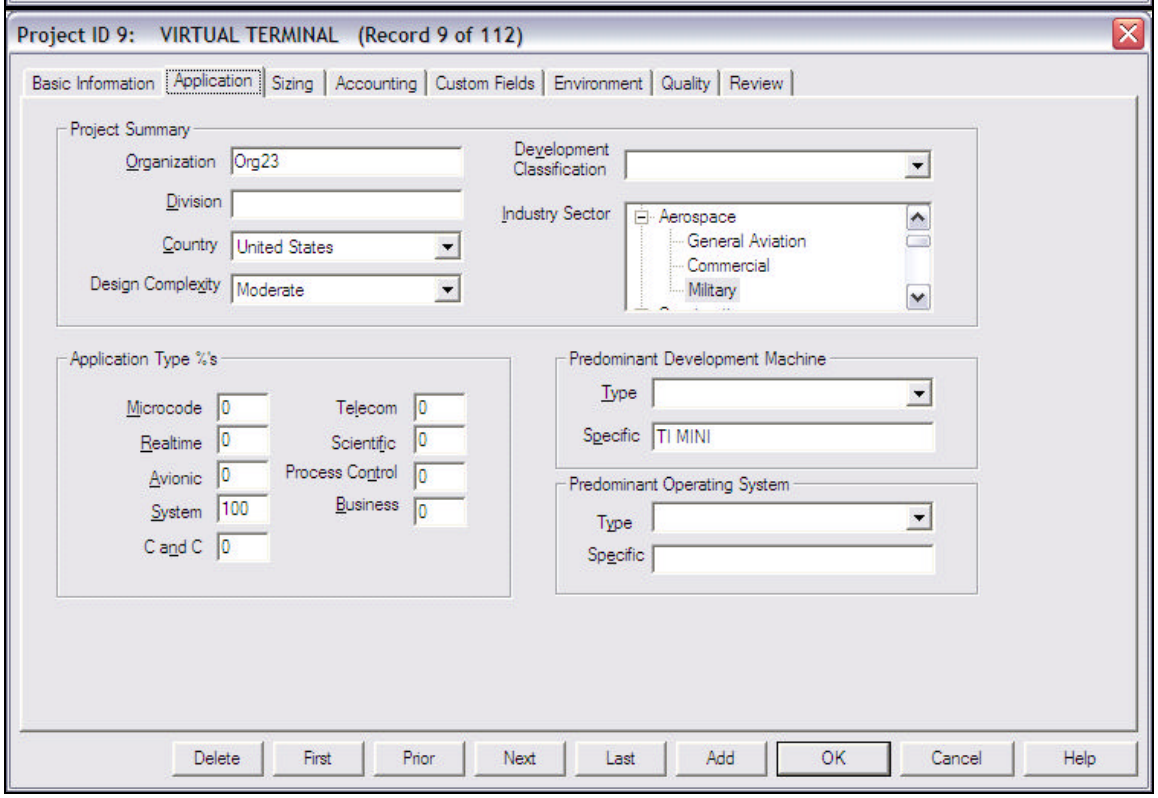

Project ID 2: PAYROLL & INVENTORY (Record 2 of 112)
Basic Information
Application
Sizing
Accounting
Custom Fields
Environment
Quality
Review
Project Summary
Organization
Org10
Division
Country
United States
Design Complexity
Moderate
Development Classification
Industry Sector
Automobile
Brewery
Chemical
Computer
Application Type %'s
Microcode
Realtime
Avionic
System
C and C
Telecom
Scientific
Process Control
Business
Predominant Development Machine
Type
Specific
Predominant Operating System
Type
Specific
Delete
First
Prior
Next
Last
Add
OK
Cancel
Help

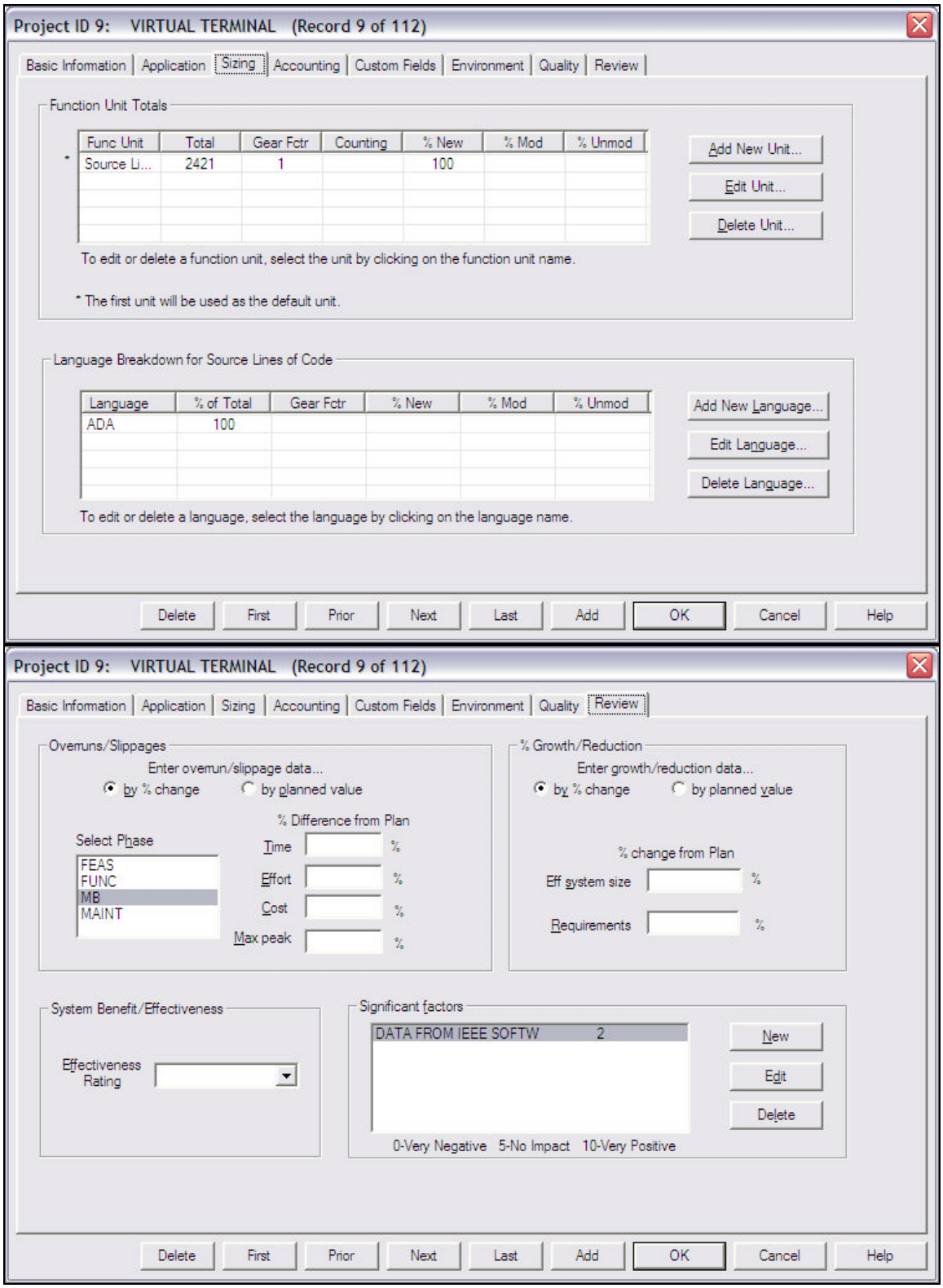

Project ID 2: PAYROLL & INVENTORY (Record 2 of 112)
Basic Information
Application
Sizing
Accounting
Custom Fields
Environment
Quality
Review
Function Unit Totals
Func Unit
Total
Gear Fctr
Counting
% New
% Mod
% Unmod
Source Li...
34700
1
100
Add New Unit...
Edit Unit...
Delete Unit...
To edit or delete a function unit, select the unit by clicking on the function unit name.
* The first unit will be used as the default unit.
Language Breakdown for Source Lines of Code
Language
% of Total
Gear Fctr
% New
% Mod
% Unmod
ADA
100
Add New Language...
Edit Language...
Delete Language...
To edit or delete a language, select the language by clicking on the language name.
Delete
First
Prior
Next
Last
Add
OK
Cancel
Help


Project ID 2: PAYROLL & INVENTORY (Record 2 of 112)
Basic Information
Application
Sizing
Accounting
Custom Fields
Environment
Quality
Review
Overruns/Slippages
Enter overrun/slippage data...
by % change
by planned value
Select Phase
FEAS
FUNC
MB
MAINT
% Difference from Plan
Time
Effort
Cost
Max peak
% Growth/Reduction
Enter growth/reduction data...
by % change
by planned value
% change from Plan
Eff system size
Requirements
System Benefit/Effectiveness
Effectiveness Rating
Significant factors
New
Edit
Delete
0-Very Negative 5-No Impact 10-Very Positive
Delete
First
Prior
Next
Last
Add
OK
Cancel
Help

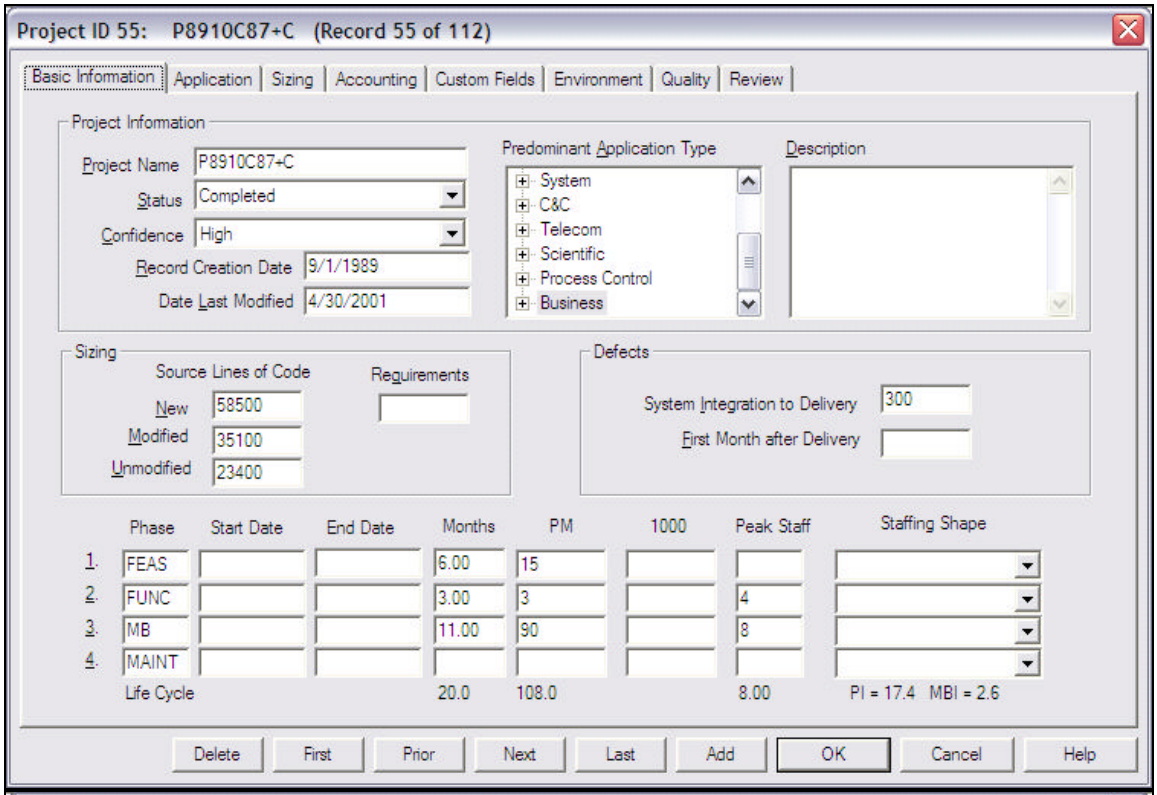
## 2. Functional Design and Main Build Documentation

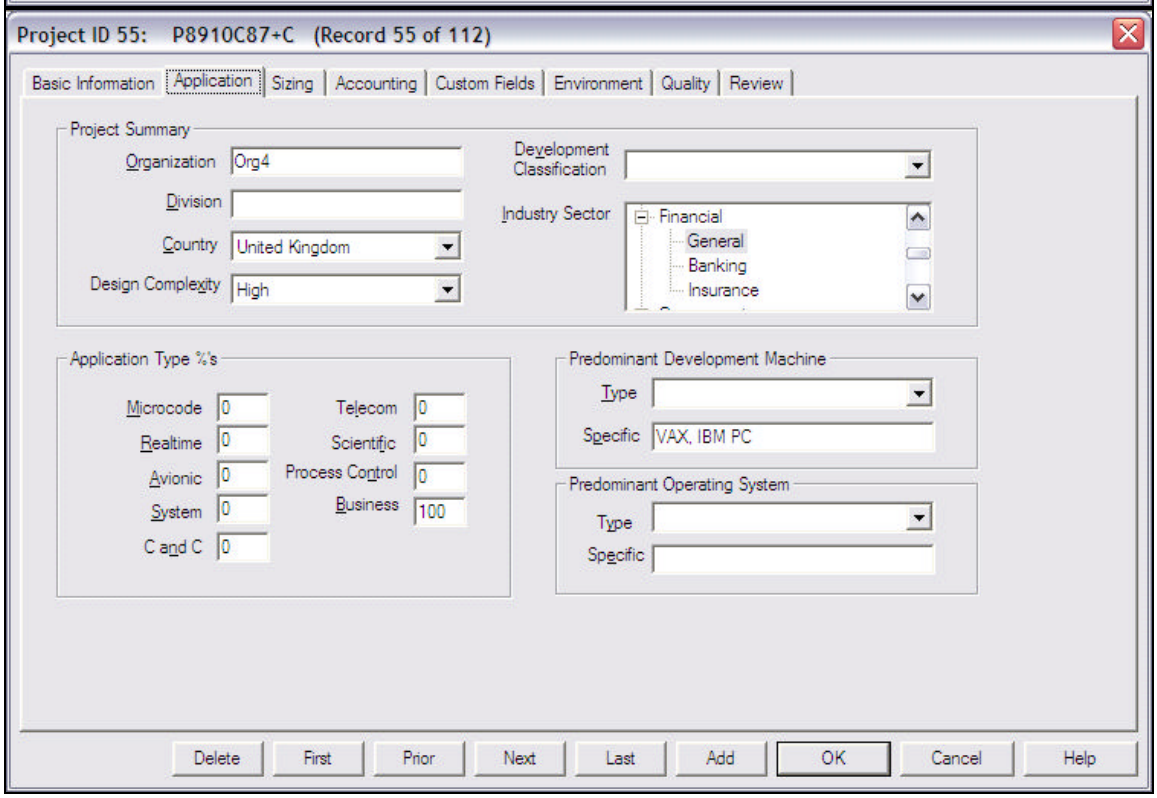


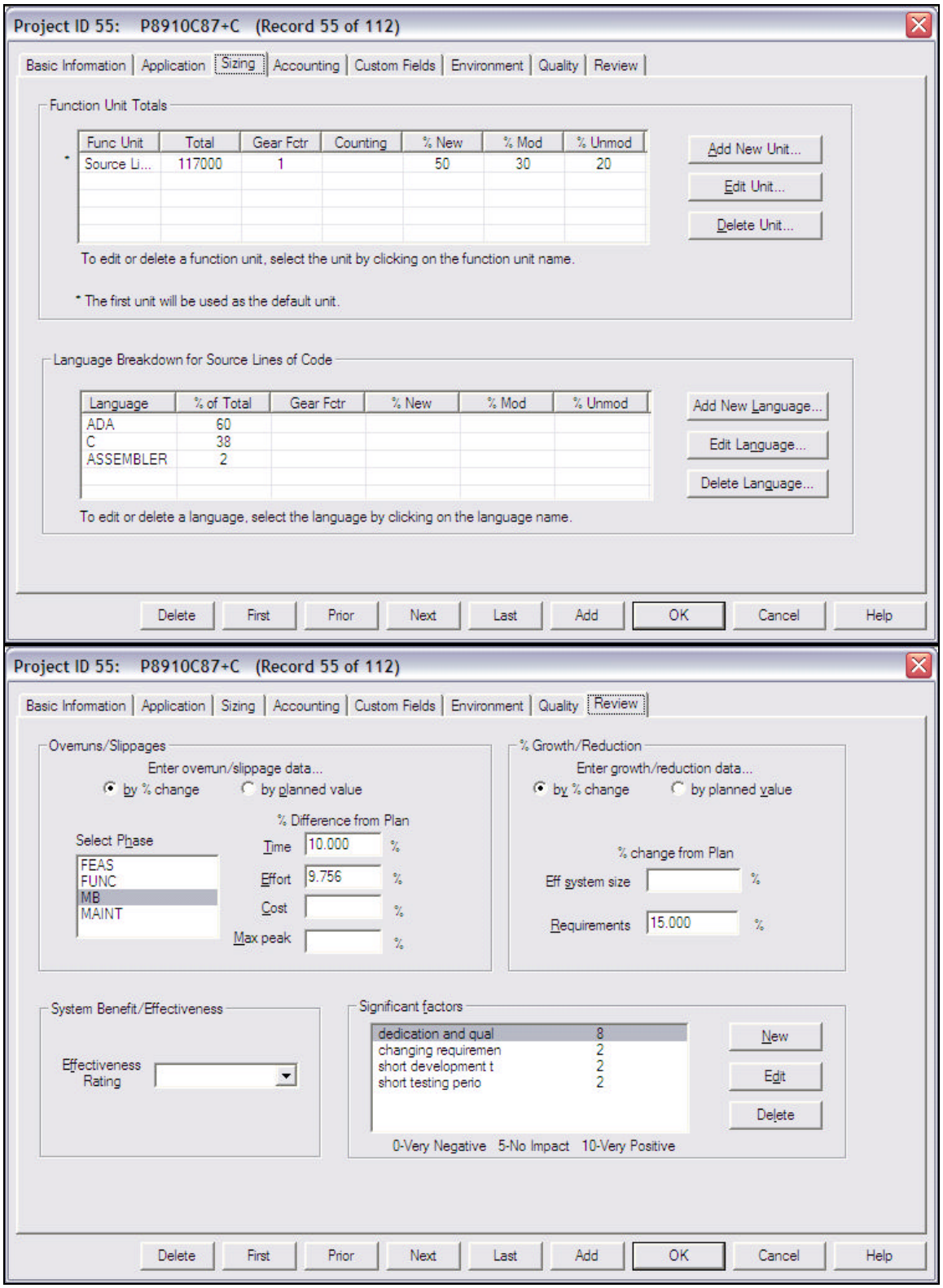




Project ID 9: VIRTUAL TERMINAL (Record 9 of 112)
Basic Information
Application
Sizing
Accounting
Custom Fields
Environment
Quality
Review
Function Unit Totals
Func Unit
Total
Gear Fctr
Counting
% New
% Mod
% Unmod
Source Li...
2421
1
100
Add New Unit...
Edit Unit...
Delete Unit...
To edit or delete a function unit, select the unit by clicking on the function unit name.
* The first unit will be used as the default unit.
Language Breakdown for Source Lines of Code
Language
% of Total
Gear Fctr
% New
% Mod
% Unmod
ADA
100
Add New Language...
Edit Language...
Delete Language...
To edit or delete a language, select the language by clicking on the language name.
Delete
First
Prior
Next
Last
Add
OK
Cancel
Help
Project ID 9: VIRTUAL TERMINAL (Record 9 of 112)
Basic Information
Application
Sizing
Accounting
Custom Fields
Environment
Quality
Review
Overruns/Slippages
Enter overrun/slippage data...
by % change
by planned value
Select Phase
FEAS
FUNC
MB
MAINT
% Difference from Plan
Time
Effort
Cost
Max peak
% Growth/Reduction
Enter growth/reduction data...
by % change
by planned value
% change from Plan
Eff system size
Requirements
System Benefit/Effectiveness
Effectiveness Rating
Significant factors
DATA FROM IEEE SOFTW 2
New
Edit
Delete
0-Very Negative 5-No Impact 10-Very Positive
Delete
First
Prior
Next
Last
Add
OK
Cancel
Help

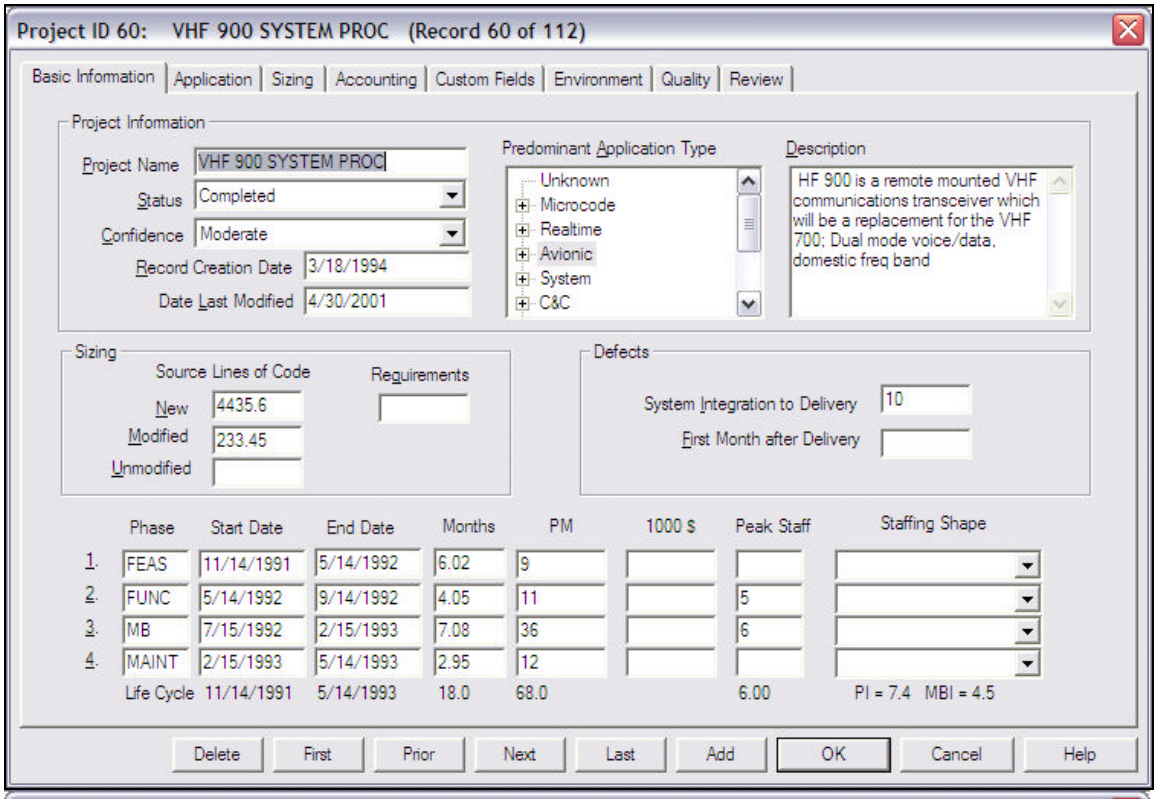
## 3. Feasibility, Functional Design and Main Build

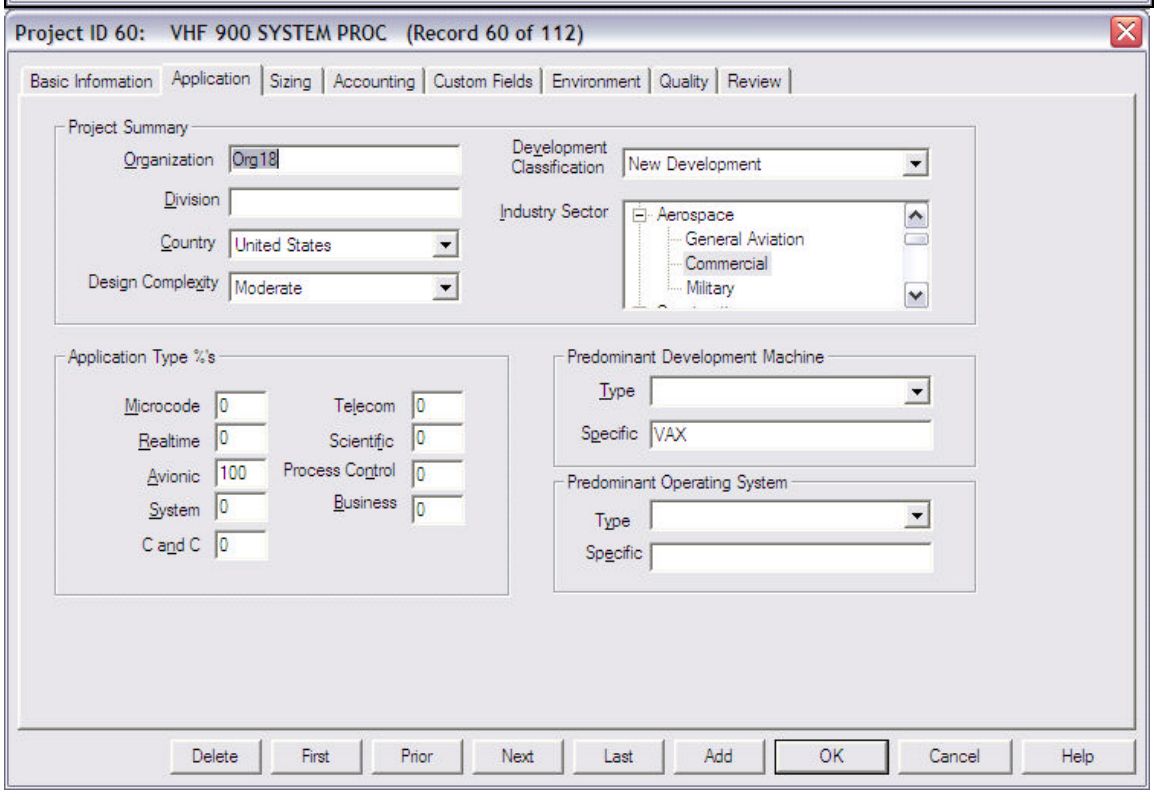


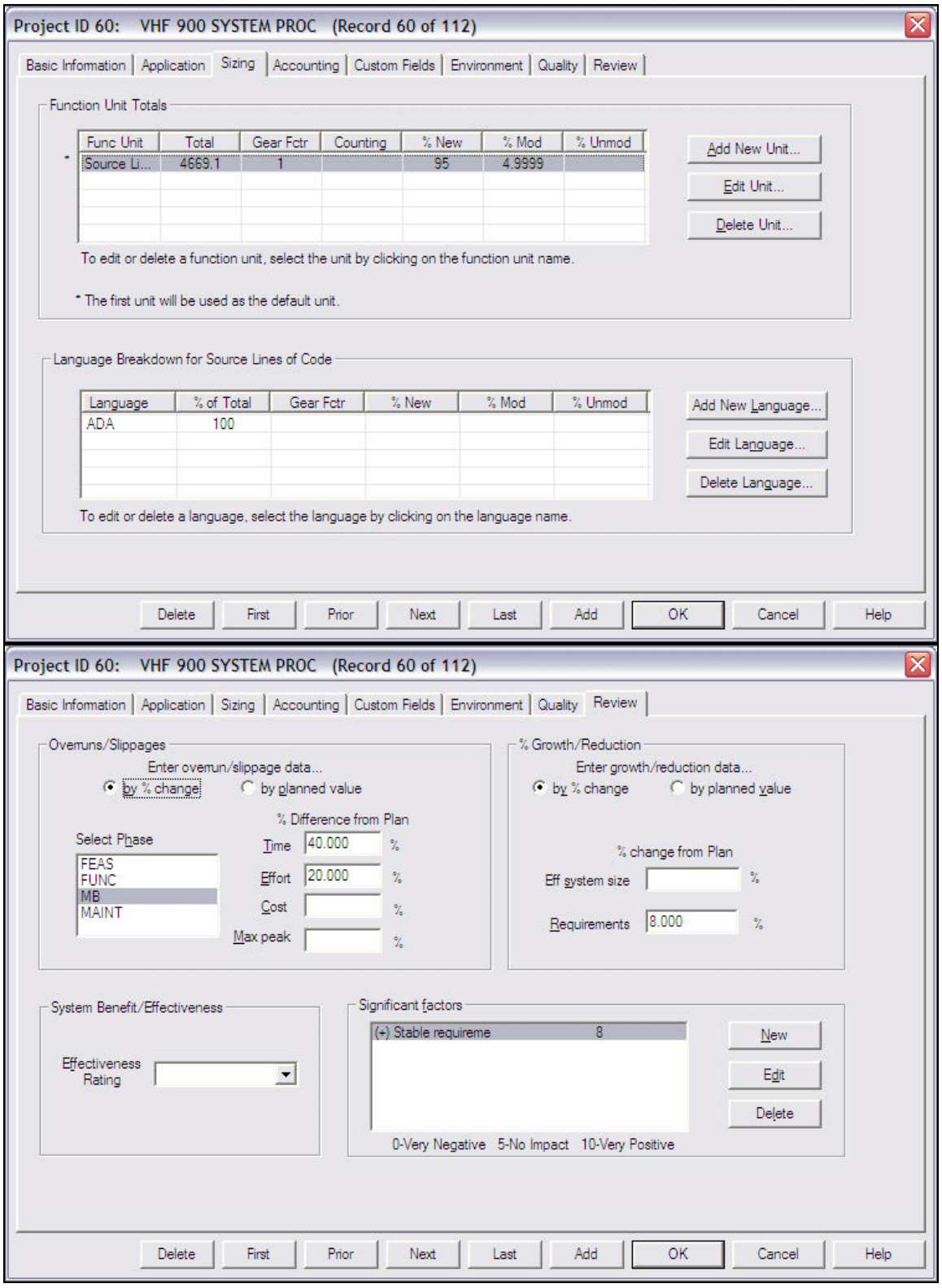




Project ID 55: P8910C87+C (Record 55 of 112)
Basic Information | Application | Sizing | Accounting | Custom Fields | Environment | Quality | Review
Function Unit Totals
Func Unit | Total | Gear Fctr | Counting | % New | % Mod | % Unmod
Source Li... | 117000 | 1 | | 50 | 30 | 20
Add New Unit...
Edit Unit...
Delete Unit...
To edit or delete a function unit, select the unit by clicking on the function unit name.
* The first unit will be used as the default unit.
Language Breakdown for Source Lines of Code
Language | % of Total | Gear Fctr | % New | % Mod | % Unmod
ADA | 60
C | 38
ASSEMBLER | 2
Add New Language...
Edit Language...
Delete Language...
To edit or delete a language, select the language by clicking on the language name.
Delete | First | Prior | Next | Last | Add | OK | Cancel | Help
Project ID 55: P8910C87+C (Record 55 of 112)
Basic Information | Application | Sizing | Accounting | Custom Fields | Environment | Quality | Review
Overruns/Slippages
Enter overrun/slippage data...
by % change
by planned value
Select Phase
FEAS
FUNC
MB
MAINT
% Difference from Plan
Time 10.000 %
Effort 9.756 %
Cost %
Max peak %
% Growth/Reduction
Enter growth/reduction data...
by % change
by planned value
% change from Plan
Eff system size %
Requirements 15.000 %
System Benefit/Effectiveness
Effectiveness Rating
Significant factors
dedication and qual 8
changing requiremen 2
short development t 2
short testing perio 2
New
Edit
Delete
0-Very Negative 5-No Impact 10-Very Positive
Delete | First | Prior | Next | Last | Add | OK | Cancel | Help

## 4. Feasibility, Functional Design, Main Build and Maintenance




Project ID 60: VHF 900 SYSTEM PROC (Record 60 of 112)
Basic Information | Application | Sizing | Accounting | Custom Fields | Environment | Quality | Review
Function Unit Totals
Func Unit | Total | Gear Fctr | Counting | % New | % Mod | % Unmod
Source Li... | 4669.1 | 1 | | 95 | 4.9999 |
Add New Unit...
Edit Unit...
Delete Unit...
To edit or delete a function unit, select the unit by clicking on the function unit name.
* The first unit will be used as the default unit.
Language Breakdown for Source Lines of Code
Language | % of Total | Gear Fctr | % New | % Mod | % Unmod
ADA | 100 |
Add New Language...
Edit Language...
Delete Language...
To edit or delete a language, select the language by clicking on the language name.
Delete | First | Prior | Next | Last | Add | OK | Cancel | Help
Project ID 60: VHF 900 SYSTEM PROC (Record 60 of 112)
Basic Information | Application | Sizing | Accounting | Custom Fields | Environment | Quality | Review
Overruns/Slippages
Enter overrun/slippage data...
by % change
by planned value
Select Phase
FEAS
FUNC
MB
MAINT
% Difference from Plan
Time 40.000 %
Effort 20.000 %
Cost %
Max peak %
% Growth/Reduction
Enter growth/reduction data...
by % change
by planned value
% change from Plan
Eff system size %
Requirements 8.000 %
System Benefit/Effectiveness
Effectiveness Rating
Significant factors
(+) Stable requireme 8
New
Edit
Delete
0-Very Negative 5-No Impact 10-Very Positive
Delete | First | Prior | Next | Last | Add | OK | Cancel | Help

THIS PAGE INTENTIONALLY LEFT BLANK

# C. CURRENT RISK MODEL VALIDATION DETAILS

## A. METRICS MAP

Figure C-2 contains 27 scenarios established to test the accuracy of the software risk model. Scenario AAA[64] is the most logical choice and deviates from the (Nogu00) metric definitions the least. The most extreme interpretation of the metric definition is scenario CCC.

The scenarios were developed to constrain the SRM after initial observations demonstrated that Scenario AAA was too optimistic in 98.2% of the experiments. The resulting scenarios were designed to exhaust the available resources and attempt to produce less optimistic results.

**Efficiency** – If the scenario begins with an -A-, this means that the break point between low and high efficiency is a productivity index of ten. In this configuration, a low efficiency organization performs with a productivity index of ten or less. Establishing the *efficiency* breakpoint at ten identifies 55 projects out of the 112 initial projects as low efficiency.

Changing the *efficiency* field to -B- changes the efficiency break point to 13, increasing the number of low efficient projects to 81. And finally, a value of -C- considers all projects with a productivity index less than 18 as low efficiency. Low efficiency projects now account for about 99% (three standard deviations) of all the projects.

**Requirements** - If the second digit in the scenario is an -A-, the recorded data in the project database maps directly to the SRM. A project with a 40 in the Requirement Growth Percentage field translates as RV = 0.40. A -B- in the requirements digit replaces all zeros and non-entries with a base-line value of 25%. A field with a value greater than zero remains unaltered. Finally, a -C- causes all entries to be increases by 25%. Figure C-1 provides additional clarification:

[64] A scenario represented as ABC is interpreted as (Efficiency – A, Requirements – B, Complexity – C).

| Scenario | Db Field | Db Value | SRM - EF | SRM - RV | SRM - CX |
|---|---|---|---|---|---|
| Eff – A (Axx) | Productivity Index | 10 | Low | - | - |
| Eff – A (Axx) | Productivity Index | 10.2 | High | - | - |
| Eff – B (Bxx) | Productivity Index | 13 | Low | - | - |
| Eff – B (Bxx) | Productivity Index | 14 | High | - | - |
| Eff – C (Cxx) | Productivity Index | 18 | Low | - | - |
| Eff – C (Cxx) | Productivity Index | 18.4 | High | - | - |
| Rv – A (xAx) | Requirements Change % | 0 | - | 0.00 | - |
| Rv – A (xAx) | Requirements Change % | 0.4 | - | 0.40 | - |
| Rv – B (xBx) | Requirements Change % | 0 | - | 0.25 | - |
| Rv – B (xBx) | Requirements Change % | 0.4 | - | 0.40 | - |
| Rv – C (xCx) | Requirements Change % | 0 | - | 0.25 | - |
| Rv – C (xCx) | Requirements Change % | 0.4 | - | 0.65 | - |
| Cx – A (xxA) | E-SLOC | 100000 | - | - | 2496.25 |
| Cx – B (xxB) | E-SLOC | 100000 | - | - | 3328.33 |
| Cx – C (xxC) | E-SLOC | 100000 | - | - | 4992.50 |

**Figure C-1. Metric Conversion Samples.**

The first column represents a digit in the conversion (A scenario represented as ABC is interpreted as (*efficiency* – A, *requirements* – B, *complexity* – C)). The second column, *Db Field,* identifies the field referenced in the project database. The third column, *Db Value,* list hypothetical samples of how the data is represented. The columns labeled *SRM-EF, SRM-RV, and SRM-CX* represent how the information in the *Db Field* interprets into the SRM.

| | Metric Mapping Scenarios | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| **Scenario** | **EF** | | | **RV** | | | **CX** | | |
| | **A-10** | **B-13** | **C-18** | **A** | **B** | **C** | **A** | **B** | **C** |
| **AAA** | X | | | X | | | X | | |
| **AAB** | X | | | X | | | | X | |
| **AAC** | X | | | X | | | | | X |
| **ABA** | X | | | | X | | X | | |
| **ABB** | X | | | | X | | | X | |
| **ABC** | X | | | | X | | | | X |
| **ACA** | X | | | | | X | X | | |
| **ACB** | X | | | | | X | | X | |
| **ACC** | X | | | | | X | | | X |
| **BAA** | | X | | X | | | X | | |
| **BAB** | | X | | X | | | | X | |
| **BAC** | | X | | X | | | | | X |
| **BBA** | | X | | | X | | X | | |
| **BBB** | | X | | | X | | | X | |
| **BBC** | | X | | | X | | | | X |
| **BCA** | | X | | | | X | X | | |
| **BCB** | | X | | | | X | | X | |
| **BCC** | | X | | | | X | | | X |
| **CAA** | | | X | X | | | X | | |
| **CAB** | | | X | X | | | | X | |
| **CAC** | | | X | X | | | | | X |
| **CBA** | | | X | | X | | X | | |
| **CBB** | | | X | | X | | | X | |
| **CBC** | | | X | | X | | | | X |
| **CCA** | | | X | | | X | X | | |
| **CCB** | | | X | | | X | | X | |
| **CCC** | | | X | | | X | | | X |
| | | | | | | | | | |
| | | | | | | | | | |
| **Legend** | EF | PI Breakpoint | | | CX | | | | |
| | A | 10 | | | A | Conversion KLOC = 40 * LGC | | | |
| | B | 13 | | | B | Conversion KLOC = 30 * LGC | | | |
| | C | 18 | | | C | Conversion KLOC = 20 * LGC | | | |
| | | | | | | | | | |
| RV | | | | | | | | | |
| A | Req Growth % = Number used | | | | | | | | |
| B | 0's in Req Growth replaced by 25% | | | | | | | | |
| C | All Req Growth % increased by 25% | | | | | | | | |

**Figure C-2. Metrics Mapping Legend.**

**Complexity** - Complexity is mapped in three different configurations. (Nogu00) recommends the equation

$$KLOC = (40LGC + 150)/1000 \quad \text{(C.1)}$$

For the -A- scenario, this equation is converted for the purposes of mapping to

$$LGC = \frac{ESLOC - 150}{40} \tag{C.2}$$

where

ESLOC = Effective Source Lines of Code
LGC = Large Granular Complexity

Preliminary trials of the validation revealed that the original equation supplied by (Nogu00) could possibly treat the conversion too conservatively. For that reason additional validation trials were set up to represent more restricted conversions. The denominator is 30 and 20 for scenario -B- and -C- respectively.

Figure C-1 illustrates the impact of changing the denominator for three views of 100K E-SLOC.

$$LGC = \frac{ESLOC - 150}{30} \tag{C.3}$$

and

$$LGC = \frac{ESLOC - 150}{20} \tag{C.4}$$

where

ESLOC = Effective Source Lines of Code
LGC = Large Granular Complexity

## B. VALIDATION RESULTS

| | | | Actual Error (wt 5) | | | | Absolute Error (wt 4) | | | | Balance (wt 3) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **N** | **CORREL** | **Mean** | **Std** | **Rank** | **Score** | **Mean** | **Std** | **Rank** | **Score** | **% Under** | **Rank** | **Score** |
| SRM - AAA | 111 | 0.8214 | -77% | 28% | 12 | 60 | 78% | 24% | 12 | 48 | 98% | 12 | 36 |
| SRM - ACC | 111 | 0.8485 | -55% | 41% | 9 | 45 | 59% | 35% | 8 | 32 | 95% | 10 | 30 |
| SRM - BAA | 111 | 0.8403 | -71% | 35% | 11 | 55 | 73% | 31% | 11 | 44 | 96% | 11 | 33 |
| SRM - BCC | 111 | 0.8686 | -43% | 51% | 8 | 40 | 55% | 38% | 7 | 28 | 80% | 7 | 21 |
| SRM - CAA | 111 | 0.7645 | -57% | 54% | 10 | 50 | 70% | 34% | 10 | 40 | 83% | 8 | 24 |
| SRM - CCC | 111 | 0.7952 | -18% | 78% | 2 | 10 | 69% | 41% | 9 | 36 | 62% | 1 | 3 |
| SSE - Business Systems | 111 | 0.7815 | -36% | 33% | 7 | 35 | 42% | 24% | 5 | 20 | 88% | 9 | 27 |
| SSE - Systems Software | 111 | 0.7815 | -11% | 45% | 1 | 5 | 36% | 30% | 3 | 12 | 68% | 3 | 9 |
| SSE - Process Control | 111 | 0.7815 | 34% | 69% | 6 | 30 | 53% | 55% | 6 | 24 | 37% | 2 | 6 |
| COCOMO - Organic | 111 | 0.785 | -20% | 40% | 5 | 25 | 36% | 26% | 3 | 12 | 75% | 6 | 18 |
| COCOMO - Semi-Detached | 111 | 0.7857 | -18% | 41% | 2 | 10 | 35% | 27% | 1 | 4 | 71% | 4 | 12 |
| COCOMO - Embedded | 111 | 0.7864 | -18% | 40% | 2 | 10 | 35% | 26% | 1 | 4 | 71% | 4 | 12 |

| | | | Under Estimation (wt 2) | | | | | Over Estimation (wt 1) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **N** | **CORREL** | **N-under** | **Mean** | **Std** | **Rank** | **Score** | **N-over** | **Mean** | **Std** | **Rank** | **Score** | **Total** |
| SRM - AAA | 111 | 0.8214 | 109 | -79% | 23% | 12 | 24 | 2 | 37% | 40% | 7 | 7 | 175 |
| SRM - ACC | 111 | 0.8485 | 105 | -61% | 34% | 8 | 16 | 6 | 38% | 48% | 8 | 8 | 131 |
| SRM - BAA | 111 | 0.8403 | 107 | -74% | 29% | 10 | 20 | 4 | 27% | 26% | 1 | 1 | 153 |
| SRM - BCC | 111 | 0.8686 | 89 | -61% | 37% | 7 | 14 | 22 | 31% | 30% | 5 | 5 | 108 |
| SRM - CAA | 111 | 0.7645 | 92 | -77% | 32% | 11 | 22 | 19 | 40% | 27% | 10 | 10 | 146 |
| SRM - CCC | 111 | 0.7952 | 69 | -70% | 37% | 9 | 18 | 42 | 67% | 47% | 11 | 11 | 78 |
| SSE - Business Systems | 111 | 0.7815 | 98 | -44% | 21% | 6 | 12 | 13 | 27% | 40% | 1 | 1 | 95 |
| SSE - Systems Software | 111 | 0.7815 | 76 | -34% | 21% | 2 | 4 | 35 | 39% | 44% | 9 | 9 | 39 |
| SSE - Process Control | 111 | 0.7815 | 41 | -26% | 16% | 1 | 2 | 70 | 70% | 63% | 12 | 12 | 74 |
| COCOMO - Organic | 111 | 0.785 | 83 | -37% | 21% | 5 | 10 | 28 | 30% | 39% | 3 | 3 | 68 |
| COCOMO - Semi-Detached | 111 | 0.7857 | 79 | -37% | 20% | 3 | 6 | 32 | 30% | 38% | 3 | 3 | 35 |
| COCOMO - Embedded | 111 | 0.7864 | 79 | -37% | 20% | 3 | 6 | 32 | 31% | 37% | 5 | 5 | 37 |

**Table C-1. Consolidated Validation Results.**

Table C-1 is reproduced from Chapter IV to support explanation of the validation results in this appendix. The detail results are presented in three sections, one for each of model under comparison.

### 1. The Software Risk Model (SRM)

The following illustrations demonstrate the results of mapping the 112[65] projects to the software risk model and projecting their project durations. The figures present analysis of the 27 scenarios from three perspectives; each representing nine scenarios. Also provided are specific details for two scenarios from each perspective. Each of these two scenarios bound the possible values of the SRM under these conditions.

[65] This dissertation refers to the "original 112" software projects, however, during validation one project was discarded due to inconsistent information. The actual validation occurs against 111 software projects.

### *a. SRM Results ($EF_{high}$ > 10)*

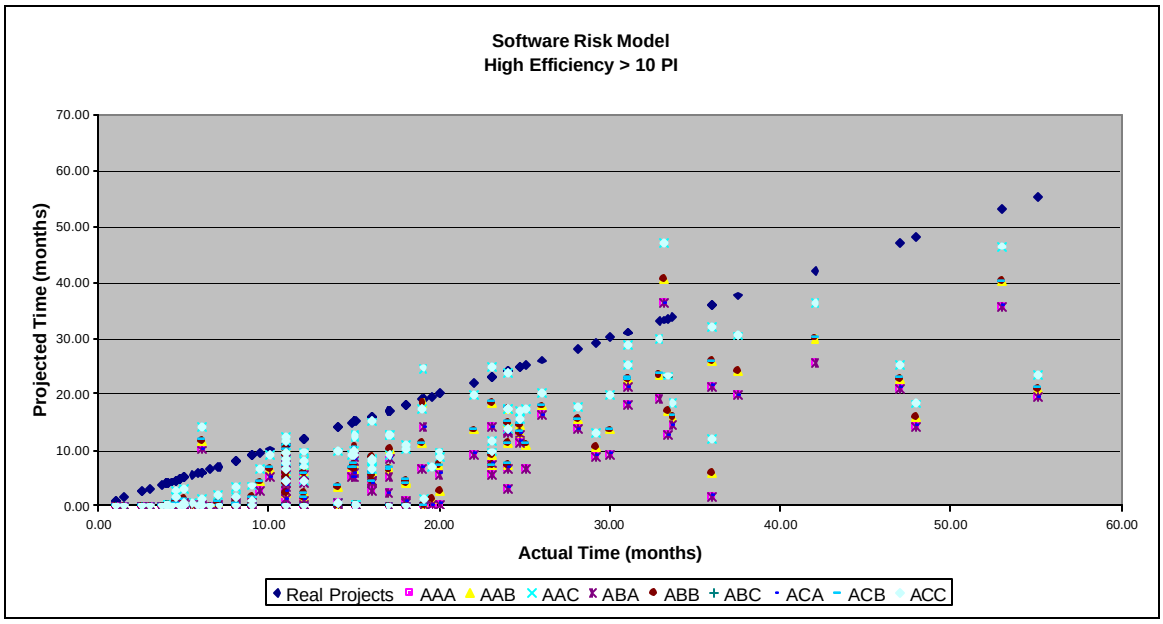



**Figure C-3. SRM Results ($EF_{high}$ > 10).**

Figure C-3 represents the SRM projection of 112 projects when *Productivity Index* of 10 servers as the break point between high and low efficiency organizations. Scenario AAA uses the most logical interpretation of the model's metric definitions yet is the least accurate. Scenario ACC constrains the input parameters and produces results with less error. The data provides the absolute error between the actual project data and the projections of the SRM. As indicated, scenario AAA has an average absolute error of 78% percent, projecting the actual project duration at only 22 percent of the actual value. Altering the *requirements* and the *complexity* conversion while fixing the *efficiency* achieves an average net gain of 18%. Scenario ACC is the most conservative scenario when the efficiency is fixed. Scenario ACC projects the project durations at approximately 40% the actual value. The SRM is most sensitive to changes in the complexity. Altering the requirement volatility produces only nominal effects.

Figure C-4 and Figure C-5 below provide additional detail obtain during the validation of the SRM. The scenarios AAA and ACC are the only two represented

out of the nine scenarios illustrated in Figure C-3. These two scenarios were chosen because they bound the effects of all of the scenarios with the *efficiency* break point of ten (Axx).

Figure C-4 and Figure C-5, as well as the additional support figures; provide comparison information in four views to support Table C-1. The four views support four out of the five[66] evaluation criteria in Table C-1. The four views presented, beginning in the top left quadrant and continuing left to right are:

- Actual Error
- Absolute Error
- Under Estimation
- Over Estimation

[66] The fifth view, balance, does not require a figure to elaborate.

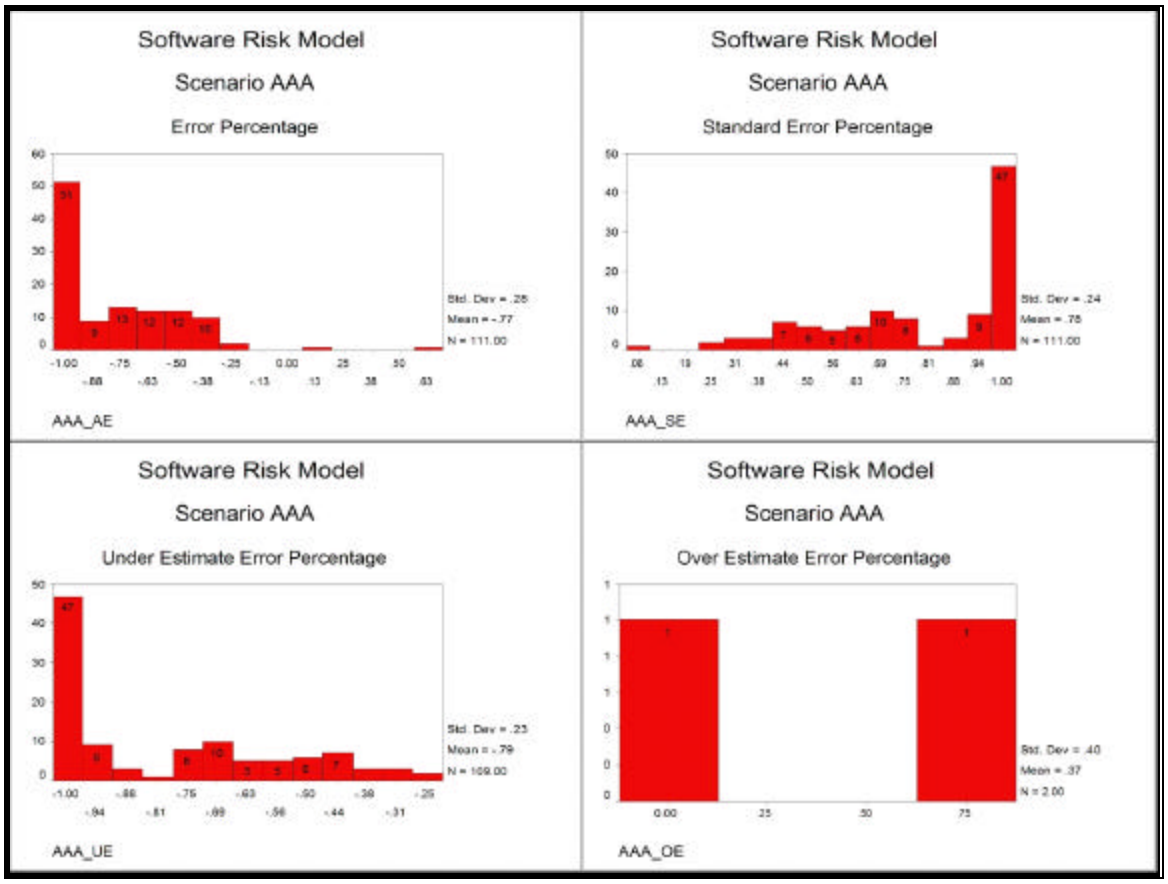



**Figure C-4. Scenario AAA Validation Performance.**

Scenario AAA ranked 12th in overall model performance:

- Actual Error – The mean error is -0.77 or an average of 77% from the actual value. The average error has a standard deviation of 28%. Three projects, or 2.7%, are projected within 25% of the actual value. Scenario AAA ranked 12 of 12 for average actual error.
- Absolute Error – The absolute error considers all errors (either over or under) as equal and provides a single value to represent the error. Scenario AAA projected the actual project performance with an absolute error of 78% ± 24%. Scenario AAA ranked 12 of 12 for average absolute error.
- Under Estimate – The SRM (Scenario AAA) projected 109 out of 111 projects under the actual value, 98.2%. When the SRM projects short, it does so with an average error of 79%. The majority of the under estimates occur when the SRM projects too close to zero (47). (Nogu00) states that

the SRM, "… can be used for any range of complexity and requirements volatility." Our validations determine that the SRM required at least an LGC of 576, or approximately 23 thousand lines of code. Scenario AAA ranked 12 of 12 for balance and 12 of 12 for average under-estimation error.

- Over Estimation – A model projecting beyond the required time can prevent successful bidding on project proposals. However, the effects of "over estimating" a project are far fewer than the potential effects of under estimation. The SRM, implementing Scenario AAA, over estimated two projects. One of the projects was very close (8%), the second project had an error of around 65%. Scenario AAA ranked 7 of 12 for average over-estimation error.

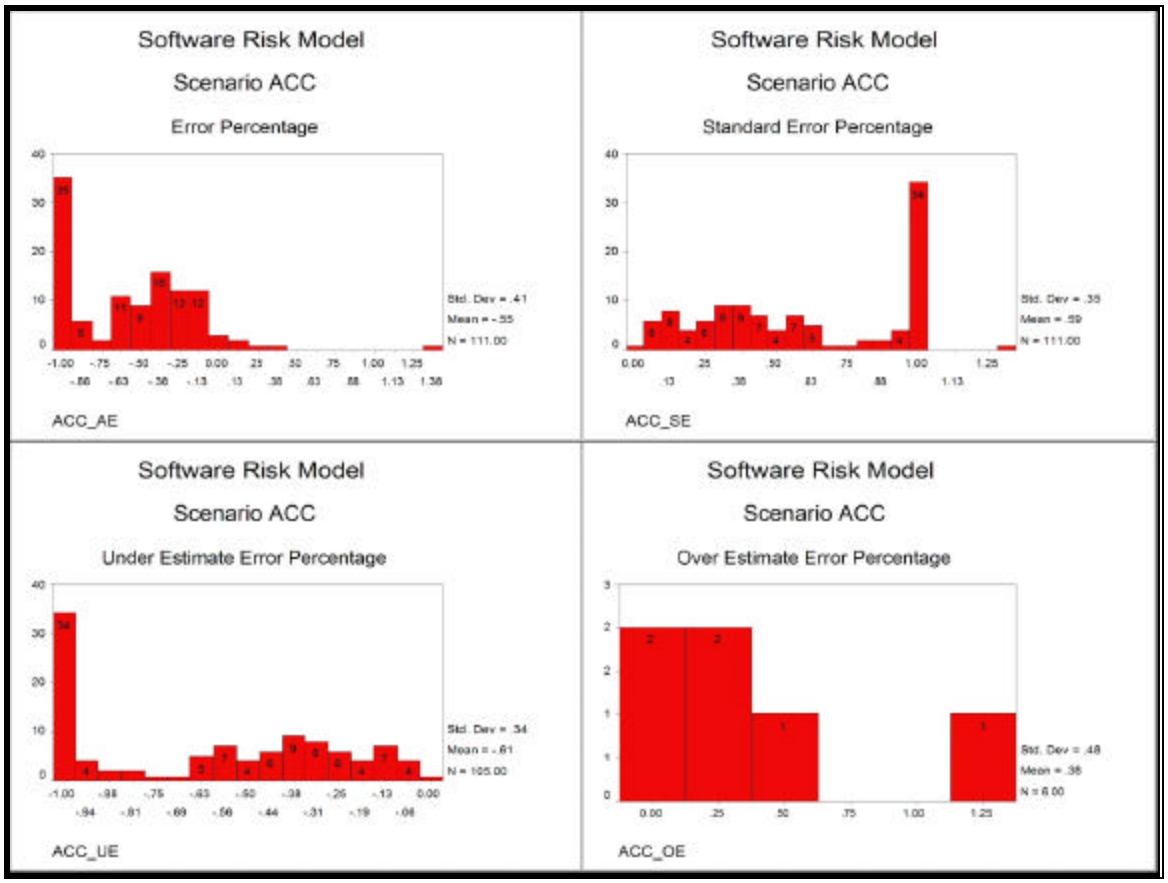



**Figure C-5. Scenario ACC Validation Performance.**

Scenario ACC ranked 9th in overall model performance:

- Actual Error – The mean error is -0.55 or an average of 55% from the actual value. The average error has a standard deviation of 41%. Scenario ACC delivers a 22% improvement over Scenario AAA. Twenty-five

projects, or 22.5%, are projected within 25% of the actual value. Scenario AAA ranked 9 of 12 for average actual error.

- Absolute Error – Scenario ACC projected the actual project performance with an absolute error of 59% ± 35%. Scenario ACC delivers a 19% improvement over Scenario AAA. Scenario ACC ranked 8 of 12 for average absolute error.
- Under Estimate – The SRM (Scenario ACC) projected 105 out of 111 projects under the actual value. When the SRM projects short, it does so with an average error of 61%. The majority of the under estimates occur when the SRM projects too close to zero (34). Scenario ACC ranked 10 of 12 for balance and 8 of 12 for average under-estimation error.
- Over Estimation – The SRM, implementing Scenario ACC, over estimated six projects. The average error was 38% with a standard deviation of 48%. Scenario ACC ranked 8 of 12 for average over-estimation error.

### *b. SRM Results (EFhigh > 13)*

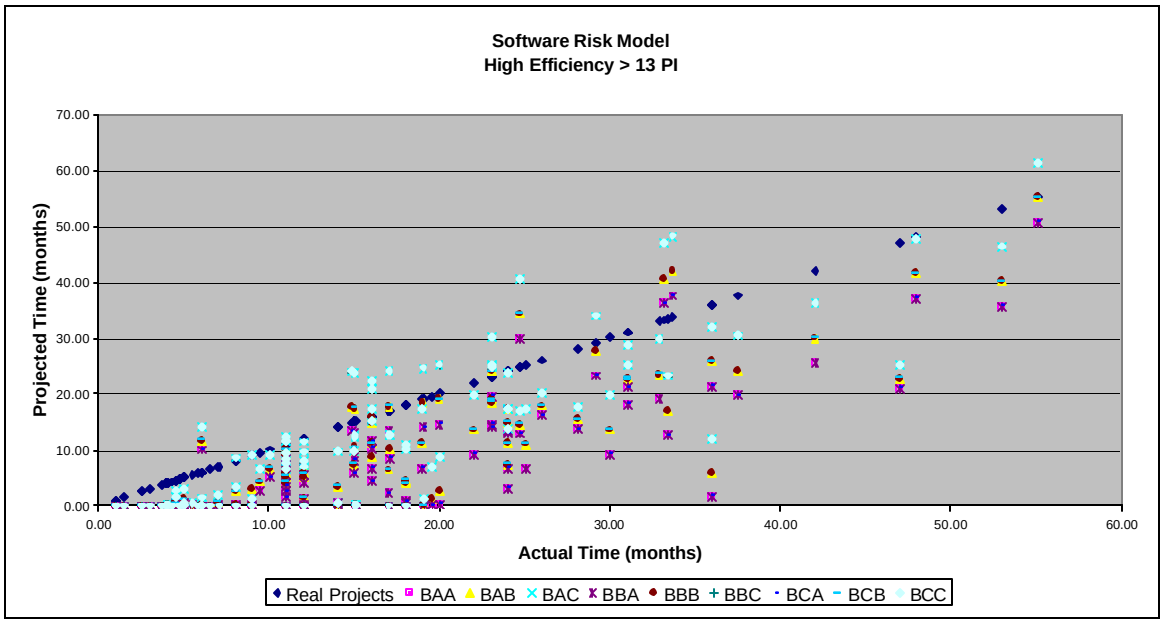



**Figure C-6. SRM Results ($EF_{high}$ > 13).**

Figure C-6 represents the risk assessment model's projection of 112 projects when Productivity Index of 13 is used to establish the break-point between high and low efficiency organizations. Scenario BAA uses the most logical interpretation of the model's metric definitions yet produces the most error. Scenario BCC constrains the

input parameters and produces better results with scenario BBC trailing closely. The data below provides the standard error between the actual project data and the projections of the SRM. As indicated, scenario BAA has an average absolute error of 73% percent, projecting the project duration at only 27 percent of the actual value. An average net gain of 18% is achieved by restricting the mapping of the requirements and the complexity conversion. Scenario BCC projects the project durations at approximately 46% the actual value. Again different requirement volatility produces only nominal effects.

Figure C-7 and Figure C-8 below provide additional detail obtained during the validation of the SRM. The scenarios BAA and BCC are the only two represented out of the nine scenarios illustrated in Figure C-6. These two scenarios were chosen because they bound the effects of all of the scenarios with the *efficiency* break point of thirteen (Bxx).


**Figure C-7. Scenario BAA Validation Performance.**

Scenario BAA ranked 11th (second worst) in overall model performance:

- Actual Error – The mean error is -0.71 or an average of 71% from the actual value. The average error has a standard deviation of 35%. Nine projects, or 8.1%, are projected within 25% of the actual value. Scenario BAA ranked 11 of 12 for average actual error.
- Absolute Error – Scenario BAA projected the actual project performance with an absolute error of 73% ± 31%. Scenario BAA ranked 11 of 12 for average absolute error.
- Under Estimate – The SRM (Scenario BAA) projected 107 out of 111 projects under the actual value. When the SRM projects short, it does so with an average error of 74%. The majority of the under estimates occur when the SRM projects too close to zero (48). Scenario BAA ranked 11 of 12 for balance and 10 of 12 for average under-estimation error.
- Over Estimation – The SRM, implementing Scenario BAA, over estimated four projects. The average error was 27% with a standard deviation of 26%. Scenario BAA ranked 1 of 12 (the best) for average over-estimation error.

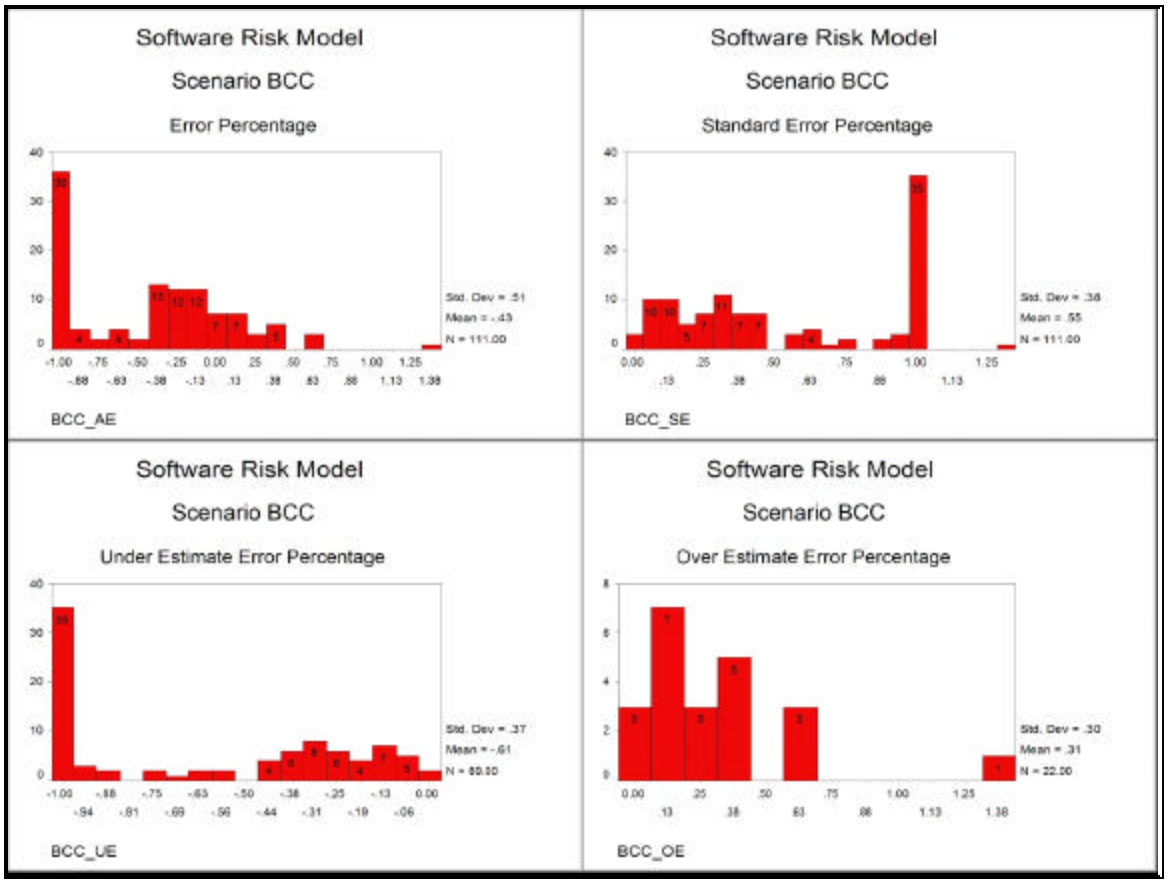



**Figure C-8. Scenario BCC Validation Performance.**

Scenario BCC ranked 8th in overall model performance:

- Actual Error – The mean error is -0.43 or an average of 43% from the actual value. The average error has a standard deviation of 51%. Scenario BCC delivers a 28% improvement over Scenario BAA. Thirty-five projects, or 31.5%, are projected within 25% of the actual value. Scenario BCC ranked 8 of 12 for average actual error.
- Absolute Error – Scenario BCC projected the actual project performance with an absolute error of 55% ± 38%. Scenario BCC delivers a 18% improvement over Scenario BAA. Scenario BCC ranked 7 of 12 for average absolute error.
- Under Estimate – The SRM (Scenario BCC) projected 89 out of 111 projects under the actual value. When the SRM projects short, it does so with an average error of 61%. The majority of the under estimates occur when the SRM projects too close to zero (35). Scenario BCC ranked 7 of 12 for balance and 7 of 12 for average under-estimation error.

- Over Estimation – The SRM, implementing Scenario BCC, over estimated twenty-two projects. The average error was 31% with a standard deviation of 30%. Scenario BCC ranked 5 of 12 for average over-estimation error.

### *c. SRM Results (EFhigh > 18)*

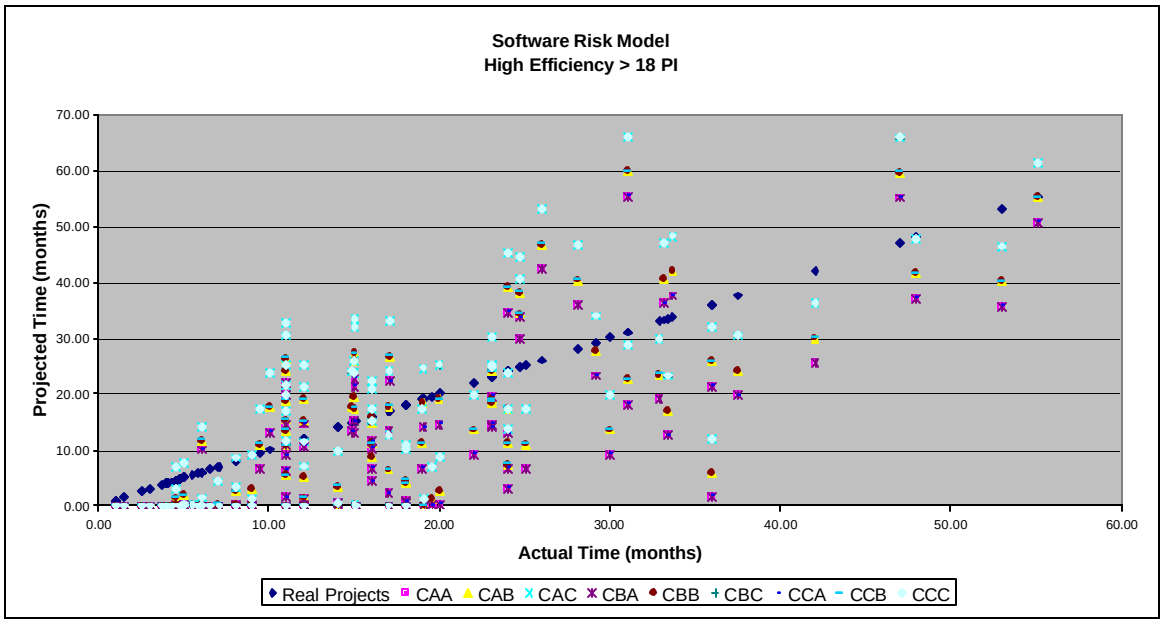



**Figure C-9. SRM Results ($EF_{high}$ > 18).**

Figure C-9 represents the risk assessment model's projection of 112 projects when Productivity Index of 18 servers as the break point between high and low efficiency organizations. Scenario CAA uses the most logical interpretation of the model's metric definitions yet produces the most absolute error. Scenario CCC constrains the input parameters and produces better results. The data below provides the absolute error between the actual project data and the projections of the SRM. As indicated, scenario CAA has an average error of 70% percent, projecting the actual project duration at only 30 percent of the actual value. An average net gain of 1% is achieved by restricting the mapping of the requirements and the complexity conversion. Scenario CCC projects the project durations at approximately 31% the actual value. Again different requirement volatility produces only nominal effects.

Figure C-10 and Figure C-11 below provide additional detail obtained during the validation of the SRM. The scenarios CAA and CCC are the only two represented out of the nine scenarios illustrated in Figure C-9. These two scenarios were chosen because they bound the effects of all of the scenarios with the *efficiency* break point of eighteen (Cxx).

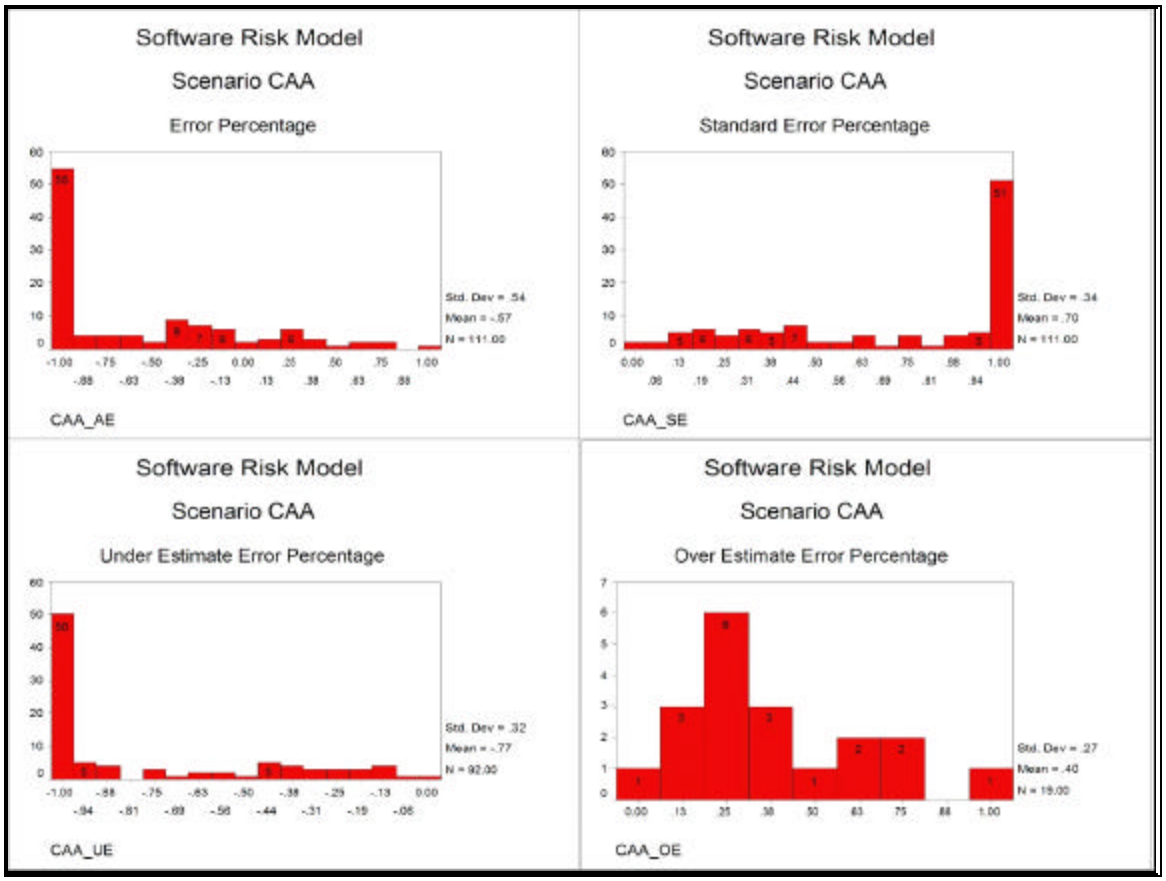



**Figure C-10. Scenario CAA Validation Performance.**

Scenario CAA ranked 10th in overall model performance:

- Actual Error – The mean error is -0.57 or an average of 57% from the actual value. The average error has a standard deviation of 54%. Nineteen projects, or 17.1%, are projected within 25% of the actual value. Scenario CAA ranked 10 of 12 for average actual error.

- Absolute Error – Scenario CAA projected the actual project performance with an absolute error of 70% ± 34%. Scenario CAA ranked 10 of 12 for average absolute error.
- Under Estimate – The SRM (Scenario CAA) projected 92 out of 111 projects under the actual value. When the SRM projects short, it does so with an average error of 77%. The majority of the under estimates occur when the SRM projects too close to zero (50). Scenario CAA ranked 8 of 12 for balance and 11 of 12 for average under-estimation error.
- Over Estimation – The SRM, implementing Scenario CAA, over estimated nineteen projects. The average error was 40% with a standard deviation of 27%. Scenario CAA ranked 10 of 12 for average over-estimation error.

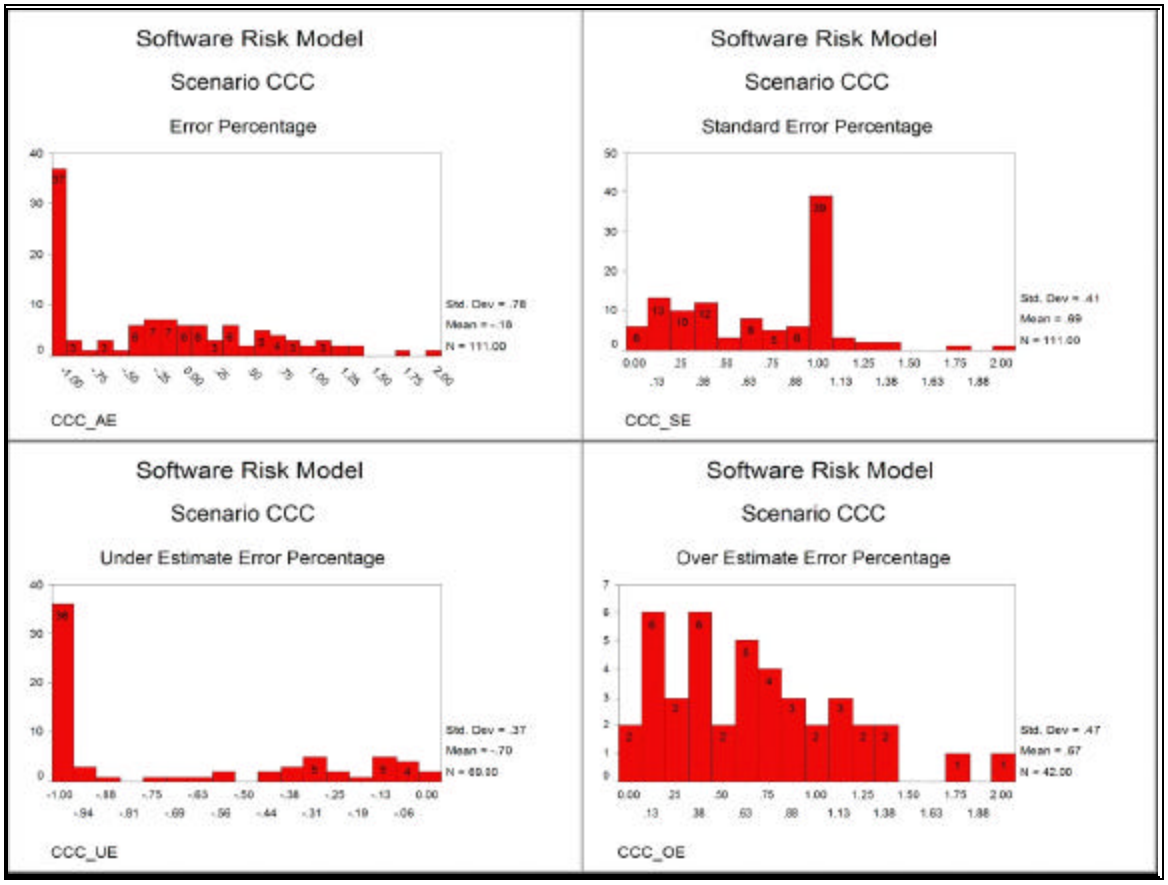



**Figure C-11. Scenario CCC Validation Performance.**

Scenario CCC ranked 6th in overall model performance (best of SRM models):

- Actual Error – The mean error is -0.18 or an average of 18% from the actual value. The average error has a standard deviation of 78%. Scenario CCC delivers a 39% improvement over Scenario CAA. Twenty-five projects, or 26.1%, are projected within 25% of the actual value. Scenario CCC tied at 2 of 12 for average actual error.
- Absolute Error – Scenario CCC projected the actual project performance with an absolute error of 69% ± 41%. Scenario CCC delivers a 1% improvement over Scenario CAA. Scenario CCC ranked 9 of 12 for average absolute error.
- Under Estimate – The SRM (Scenario CCC) projected 69 out of 111 projects under the actual value. When the SRM projects short, it does so with an average error of 70%. The majority of the under estimates occur when the SRM projects too close to zero (36). Scenario CCC ranked 1 of 12 (the best) for balance and 9 of 12 for average under-estimation error.
- Over Estimation – The SRM, implementing Scenario CCC, over estimated forty-two projects. The average error was 67% with a standard deviation of 47%. Scenario CCC ranked 11 of 12 for average over-estimation error.

#### *d. Baseline Results*

The following illustrations demonstrate the results of mapping the 112 projects to (Putn92)'s Simplified Software Equation and (Boeh81)'s Basic COCOMO Model. Following each figure are the actual results of the analysis. For each of these two models, the results presented in this validation can be considered the worst-case scenario. Three variations of each model are presented because there are not ample real world projects to test each individual variation of the model. Subsequently, projecting all of the projects validates each model.

## 2. Simplified Software Equation

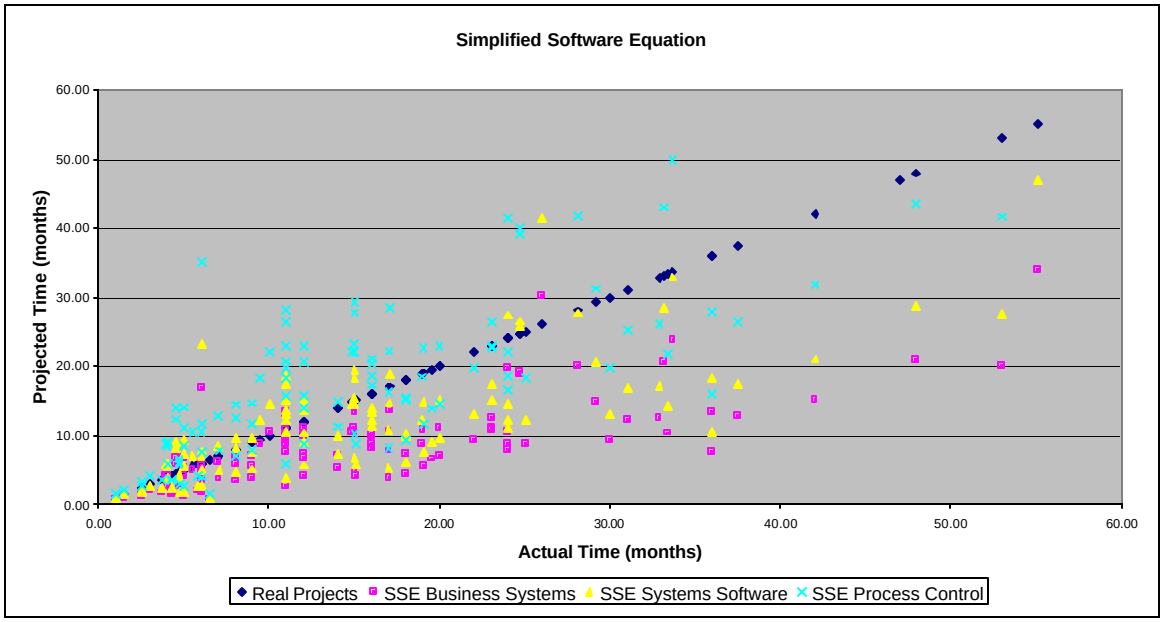



**Figure C-12. Simplified Software Equation Projections.**

The Simplified Software Equation, Product=Constant*Effort*Time, is designed to project the amount of work that has to be performed over a period of time to produce a product (Putn92). The productivity term is a proportionality constant between the other three terms of the equation. To make the comparison, the productivity parameter was obtained from tables provided in (Putn92) and represents industry averages for three different types of project developments (Business Systems, Systems Software, and Process Control). Ideally, each software development organization, would have a unique corresponding productivity term calibrated to their specific performance history. Specializing the SSE to each organization would no longer provide an equal comparison between the models (i.e. projecting in the macro perspective).

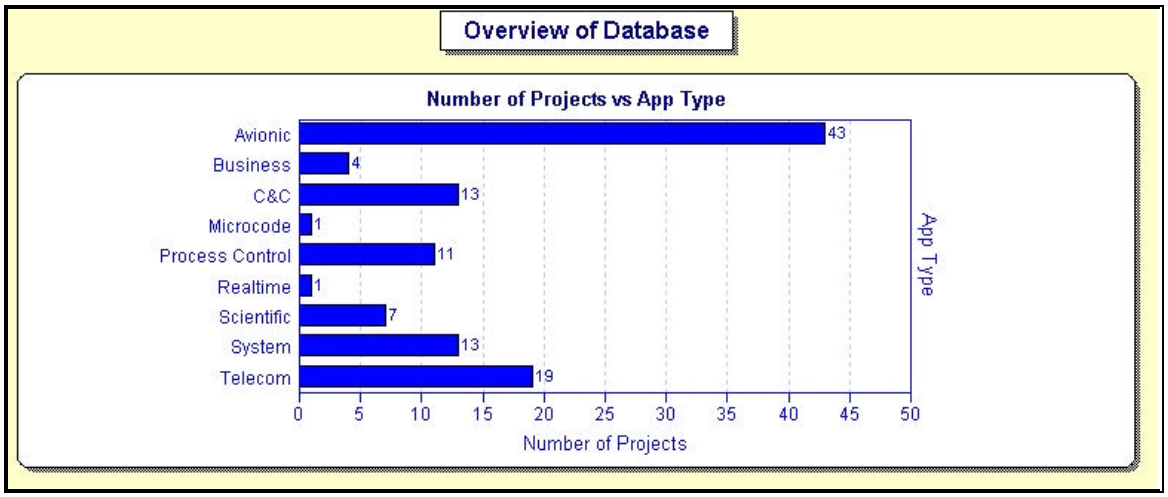



**Figure C-13. Overview of Project Database.**

For rapid analysis, (Putn92) provides a capability to compute the required calculations in a simplistic fashion. However, the margin of error increases with the gain of simplicity. The use of the simple equations requires referencing Table C-2.Process Productivity Parameters (Putn92). For comparison of the of minimum development time, the following variation of the Simplified Software Equation is implemented.

$$t_{d-\min} = 8.14(\text{SLOC}/\,PP)^{(0.43)} \qquad \text{(C.5)}$$

where,

- $t_{d-\min}$ is the minimum time for development for the Main Build
- SLOC is the number of Effective Source Lines of Code
- PP is the productivity parameter obtained from Table C-2

The Simplified Software Equation, implemented under these conditions, projected the actual software durations between 47% and 65% accuracy. It is not surprising that using a productivity parameter aligned with systems software produces the least error. Figure C-13 reiterates that 35% of the project database is comprised of projects within the standard deviation of the systems software's productivity parameter.

| Productivity Index | Productivity Parameter | Application Type | Standard Deviation |
|---|---|---|---|
| 1 | 754 | | |
| 2 | 987 | Microcode | +/- 1 |
| 3 | 1,220 | | |
| 4 | 1,597 | Firmware (ROM) | +/- 2 |
| 5 | 1,974 | Real-time embedded/ avionics | +/- 2 |
| 6 | 2,584 | | |
| 7 | 3,194 | Radar systems | +/- 3 |
| 8 | 4,181 | Command and Control | +/- 3 |
| 9 | 5,186 | Process Control | +/- 3 |
| 10 | 6,765 | | |
| 11 | 8,362 | Telecommunications | +/- 3 |
| 12 | 10,946 | | |
| 13 | 13,530 | Systems software / Scientific systems | +/- 3 |
| 14 | 17,711 | | |
| 15 | 21,892 | | |
| 16 | 28,657 | Business systems | +/- 4 |
| 17 | 35,422 | | |
| 18 | 46,368 | | |
| 19 | 57,314 | | |
| 20 | 75,025 | | |
| 21 | 92,736 | | |
| 22 | 121,393 | | |
| 23 | 150,050 | | |
| 24 | 196,418 | | |
| 25 | 242,786 | Highest Value Found | |
| 26 | 317,811 | | |
| 27 | 392,836 | | |
| 28 | 514,229 | | |
| 29 | 635,622 | | |
| 30 | 832,040 | | |
| 31 | 1,028,458 | | |
| 32 | 1,346,269 | | |
| 33 | 1,664,080 | | |
| 34 | 2,178,309 | | |
| 35 | 2,692,538 | | |
| 36 | 3,524,578 | | |

**Table C-2. Process Productivity Parameters (Putn92).**

**Three scenarios are illustrated in Figure C-12. Figure C-14,**

Figure C-15, and Figure C-16 below provide additional detail obtained during the validation of the SRM.

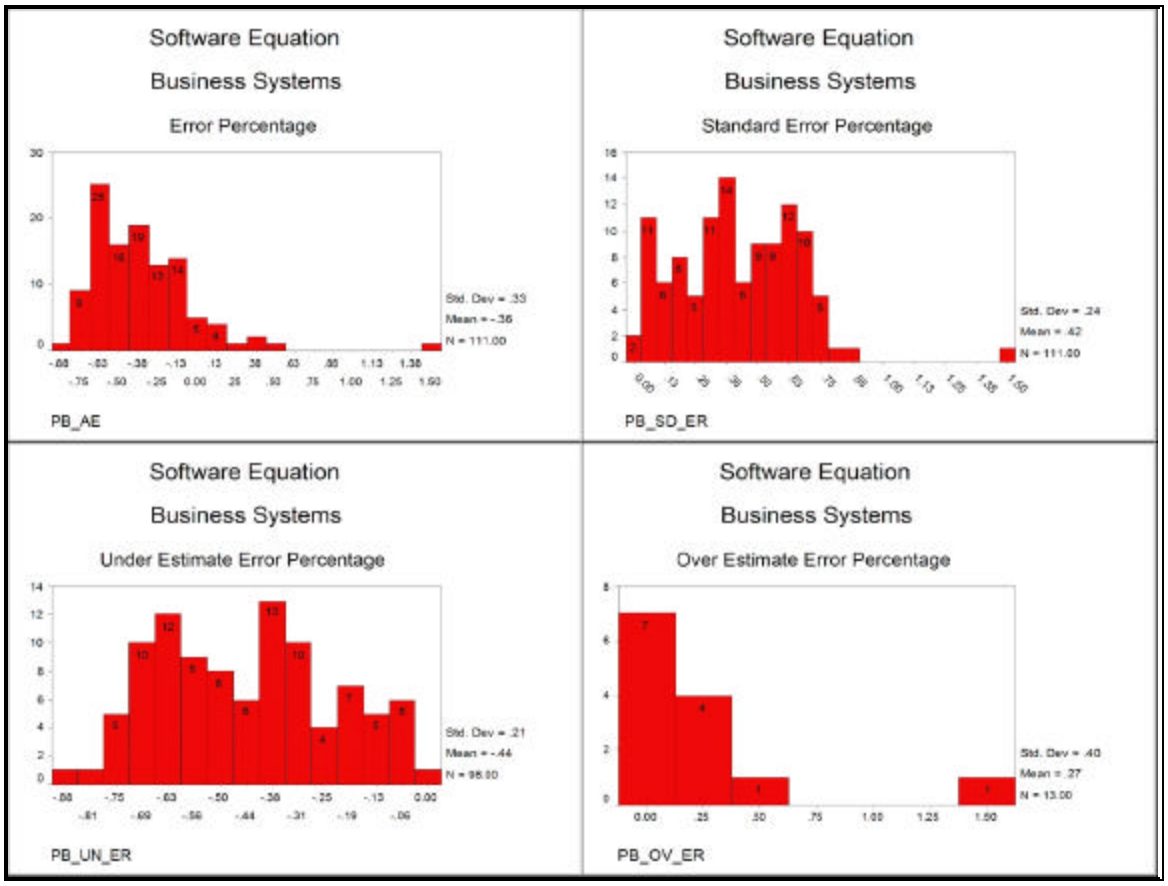



**Figure C-14. SSE (*Business System*) Validation Performance.**

Simplified Software Equation (*Business Systems*) ranked 7th in overall model performance:

- Actual Error – The mean error is -0.36 or an average of 36% from the actual value. The average error has a standard deviation of 33%. Thirty-two projects, or 28.8%, are projected within 25% of the actual value. Simplified Software Equation (*Business Systems*) ranked 7 of 12 for average actual error.
- Absolute Error – Simplified Software Equation (*Business Systems*) projected the actual project performance with an absolute error of 42% ± 24%. Simplified Software Equation (*Business Systems*) ranked 5 of 12 for average absolute error.
- Under Estimate – The Simplified Software Equation (*Business Systems*) projected 98 out of 111 projects under the actual value. When the Simplified Software Equation (*Business Systems*) projects short, it does so

with an average error of 44%. Simplified Software Equation (*Business Systems*) ranked 9 of 12 for balance and 6 of 12 for average under-estimation error.

- Over Estimation – The Simplified Software Equation, implementing *Business Systems*, over estimated thirteen projects. The average error was 27% with a standard deviation of 40%. Simplified Software Equation (*Business Systems*) tied at 1 of 12 (best) for average over-estimation error.

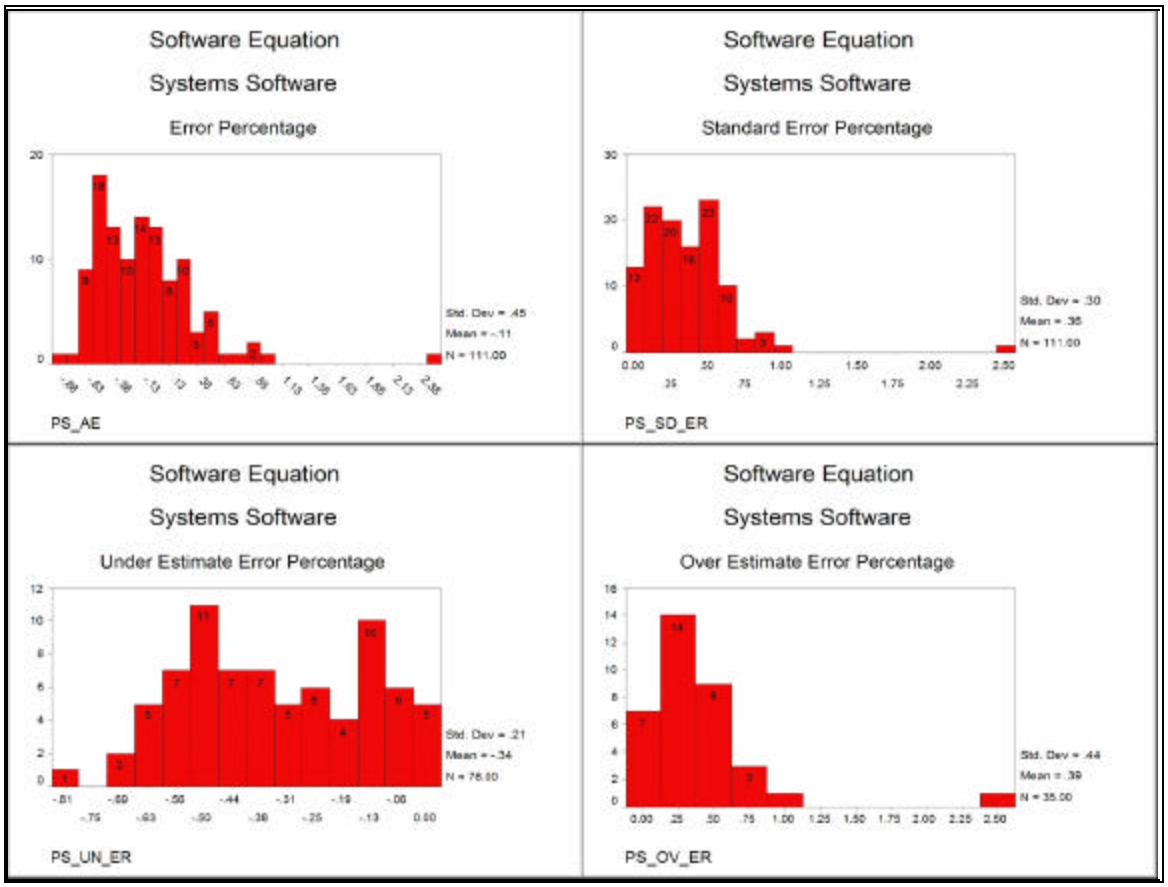



**Figure C-15. SSE (*Systems Software*) Validation Performance.**

Simplified Software Equation (*Systems Software*) ranked 3rd in overall model performance:

- Actual Error – The mean error is -0.11 or an average of 11% from the actual value. The average error has a standard deviation of 45%. Fifty-five projects, or about 50%, are projected within 25% of the actual value. Simplified Software Equation (*Systems Software*) ranked 1 of 12 (the best) for average actual error.
- Absolute Error – Simplified Software Equation (*Systems Software*) projected the actual project performance with an absolute error of 36% ± 30%. Simplified Software Equation (*Systems Software*) ranked 3 of 12 for average absolute error.
- Under Estimate – The Simplified Software Equation (*Systems Software*) projected 76 out of 111 projects under the actual value. When the Simplified Software Equation (*Systems Software*) projects short, it does so with an average error of 34%. Simplified Software Equation (*Systems Software*) ranked 3 of 12 for balance and 2 of 12 for average under-estimation error.
- Over Estimation – The Simplified Software Equation, implementing *Systems Software*, over estimated thirty-five projects. The average error was 39% with a standard deviation of 44%. Simplified Software Equation (*Systems Software*) ranked 9 of 12 for average over-estimation error.

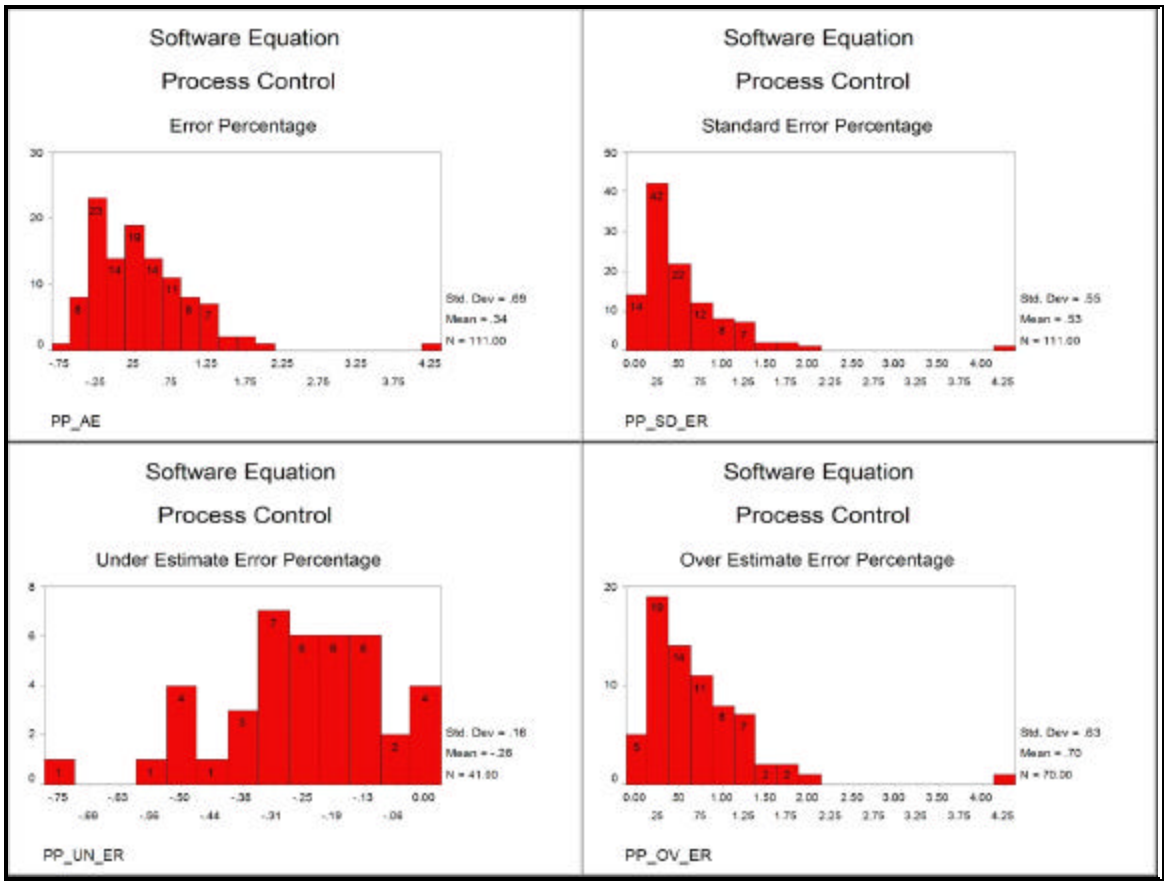



**Figure C-16. SSE (*Process Control*) Validation Performance.**

Simplified Software Equation (*Process Control*) ranked 5th in overall model performance:

- Actual Error – The mean error is 0.34 (positive) or an average of 34% above the actual value (only model that projected, on average, high). The average error has a standard deviation of 69%. Fifty-six projects, or 50.4%, are projected within 25% of the actual value. Simplified Software Equation (*Process Control*) ranked 6 of 12 for average actual error.
- Absolute Error – Simplified Software Equation (*Process Control*) projected the actual project performance with an absolute error of 53% ± 55%. Simplified Software Equation (*Process Control*) ranked 6 of 12 for average absolute error.
- Under Estimate – The Simplified Software Equation (*Process Control*) projected 41 out of 111 projects under the actual value. When the

Simplified Software Equation (*Process Control*) projects short, it does so with an average error of 26%. Simplified Software Equation (*Process Control*) ranked 2 of 12 for balance and 1 of 12 (the best) for average under-estimation error.

- Over Estimation – The Simplified Software Equation, implementing *Process Control*, over estimated seventy projects. The average error was 70% with a standard deviation of 63%. Simplified Software Equation (*Process Control*) ranked 12 of 12 (worst) for average over-estimation error.

### 3. Basic COCOMO Model

(Boeh81) provides three different versions of an equation to project the $T_{DEV}$ or software development schedule in months[67]. Managers decide on the appropriate equation depending on some distinguishing features of the software developments. Unlike the Simplified Software Equation, deriving the $T_{DEV}$ with the COCOMO family of models is a two-step process. First, users must calculate the required *Effort* and then use this value to calculate the development time. Figure C-17 illustrates the projections of the Basic COCOMO models.

[67] (Boeh81) indicates that the $T_{dev}$ considers the Main Build and the time required to produce the software specifications. The SRM and the Simplified Software Equation consider the Main Build beginning after the completion of the software specifications. All COCOMO model projections account for this difference.

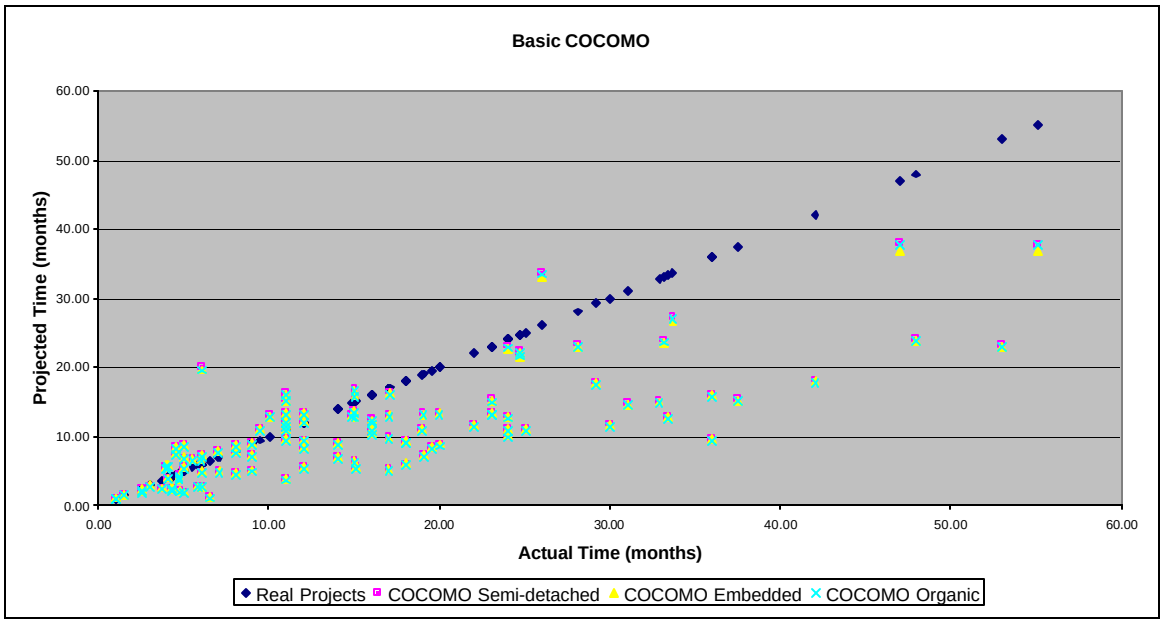



**Figure C-17. Basic COCOMO Model Projections.**

The projections have a very small standard deviation between them. This is due to the similarities of the formulas. The second step in the process is the actual calculation of the $T_{DEV}$. Marginal variation is introduced in the Effort calculation due to a larger range of constants.

The Basic COCOMO Effort and Schedule Equations utilized in the projections of Figure C-17 are listed below.

| Mode | Effort | Schedule |
|---|---|---|
| Organic | $PM = 2.4(KDSI)^{1.05}$ | $T_{DEV} = 2.5(PM)^{0.38}$ |
| Semi-detached | $PM = 3.0(KDSI)^{1.12}$ | $T_{DEV} = 2.5(PM)^{0.35}$ |
| Embedded | $PM = 3.6(KDSI)^{1.20}$ | $T_{DEV} = 2.5(PM)^{0.32}$ |

*where*

*PM = effort in person months*
*KDSI = delivered source lines of code in thousands*
*$T_{DEV}$ = development time from beginning of spec thru main build*

**Table C-3. Basic COCOMO Model Equations.**

In order to conduct validation with all three models (SRM, SSE, and COCOMO), care was taken to ensure the models were implemented in accordance with their original designs. Footnote 67 explains the need to adjust all of the projections obtained with the COCOMO equation to account for the additional time projected for the specifications. (Bohm83) states that COCOMO estimates from the beginning of the product design phase (approved, validated software requirements specifications), this key difference between the models make it necessary to account for the different model projections.

(Putn92) provides a formula, $t_{func} = t_{d\text{-min}} / 3$ (months), to project the minimum time length of the high-level functional design phase. To calculate the total time required for the *function design* and *main build,* the equivalent to COCOMO $T_{DEV}$, use the formula $\text{COCOMO}(t_{\text{dev}}) = \text{SLIM}(t_{\text{func}} + t_{\text{d-min}})$. From this equation derive

$$\text{COCOMO}((t_{\text{dev}})*0.75) = \text{SLIM}(t_{\text{d-min}}) \qquad \text{(C.6)}$$

Taking 75% of the COCOMO projection provides a suitable conversion for our analysis. Essentially, the specifications consume 25% of the total development after the requirement's phase (not accounting for the maintenance phase). Lawrence Putnam, Sr. was consulted on the conversion and he stated, "probably as good as you can do, I would do it that way").

As mentioned, the Basic COCOMO Models all perform within a small deviation of each other. In the experiments against Figure C-13, the model projects with approximately 65% accuracy. That is stating that the model on average projected a value that is 35% of actual value. Figure C-18, Figure C-19, and Figure C-20 provide additional detail obtained during the validation of the SRM. Three scenarios are illustrated in Figure C-17.

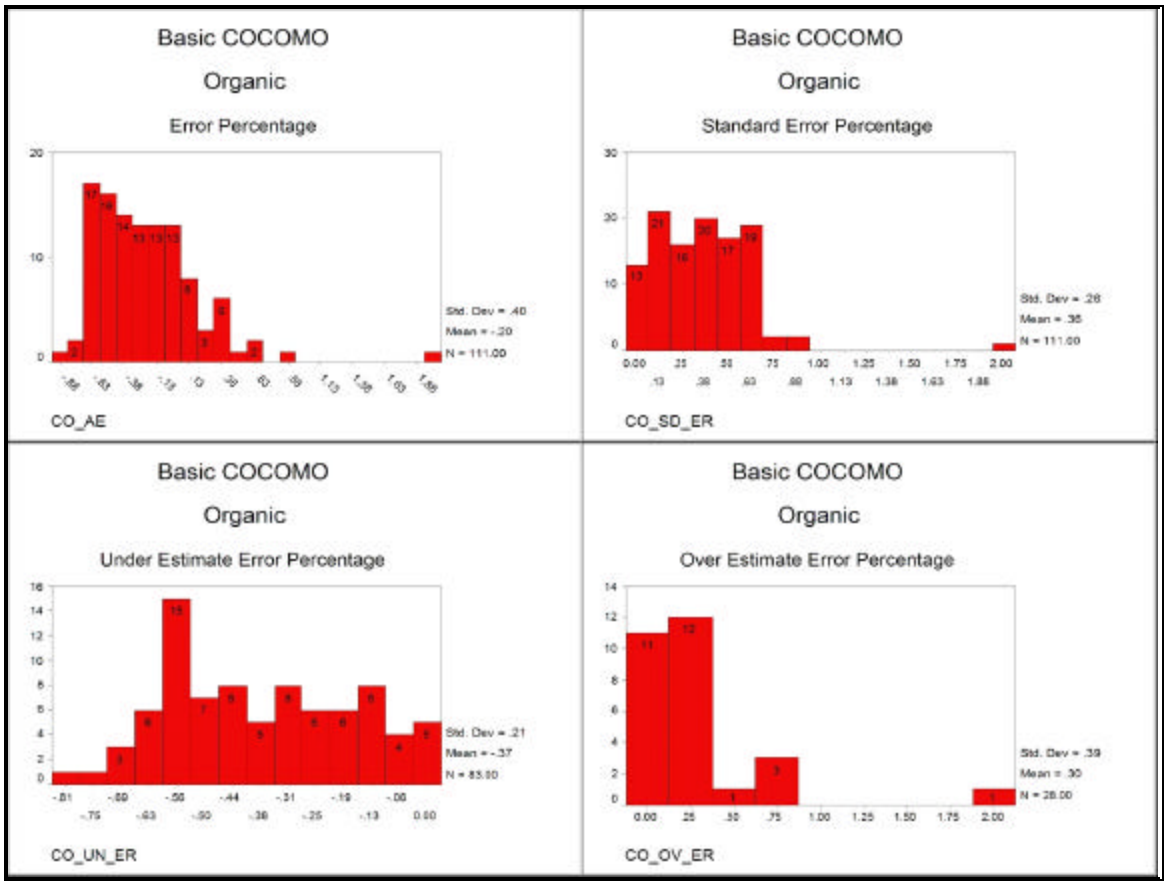



**Figure C-18. Basic COCOMO (*Organic*) Validation Performance.**

Basic COCOMO (*Organic*) ranked 4th in overall model performance:

- Actual Error – The mean error is -0.20 or an average of 20% from the actual value. The average error has a standard deviation of 40%. Fifty projects, or 45%, are projected within 25% of the actual value. Basic COCOMO (*Organic*) ranked 5 of 12 for average actual error.
- Absolute Error – Basic COCOMO (*Organic*) projected the actual project performance with an absolute error of 36% ± 26%. Basic COCOMO (*Organic*) ranked 3 of 12 for average absolute error.
- Under Estimate – The Basic COCOMO (*Organic*) projected 83 out of 111 projects under the actual value. When the Basic COCOMO (*Organic*) projects short, it does so with an average error of 37%. Basic COCOMO (*Organic*) ranked 6 of 12 for balance and 5 of 12 for average under-estimation error.

- Over Estimation – The Basic COCOMO, implementing *Organic*, over estimated twenty-eight projects. The average error was 30% with a standard deviation of 39%. Basic COCOMO (*Organic*) ranked 3 of 12 for average over-estimation error.

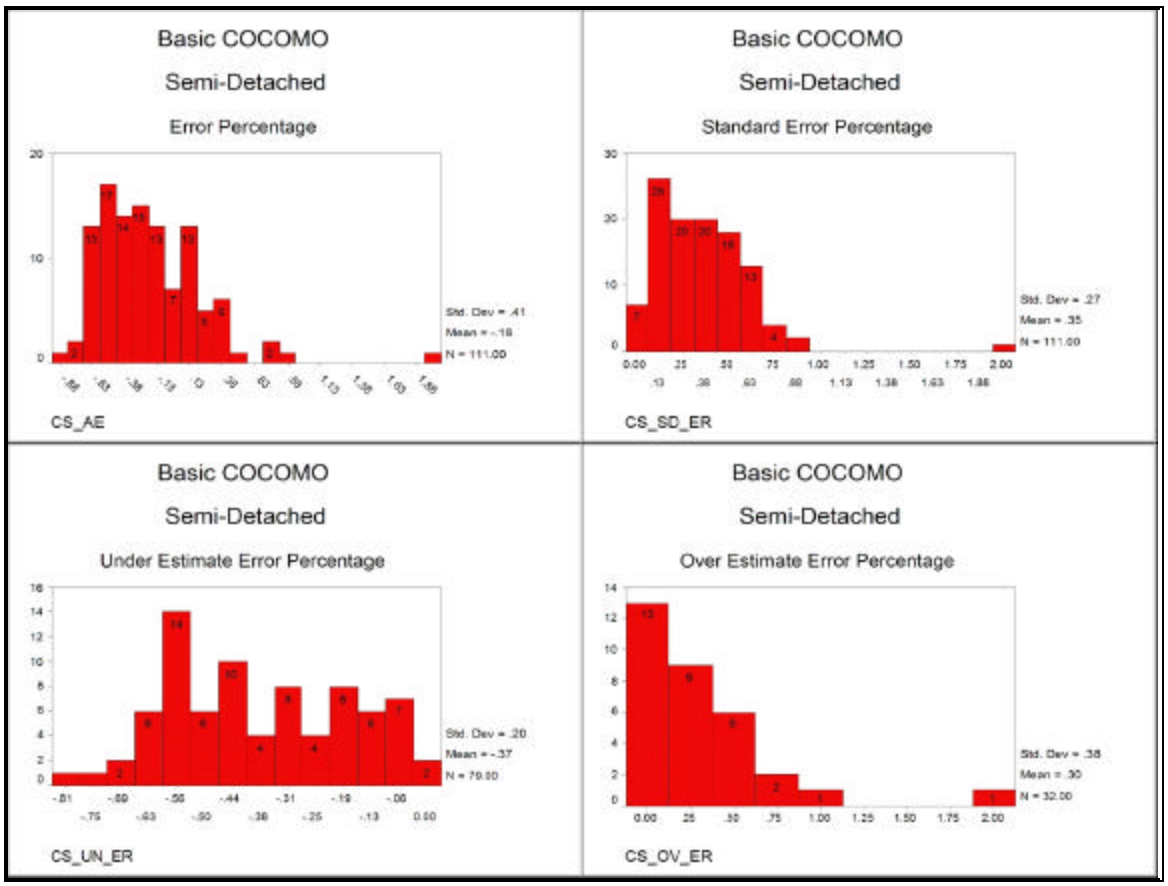



**Figure C-19. Basic COCOMO (*Semi-Detached*) Validation Performance.**

Basic COCOMO (*Semi-Detached*) ranked 1st in overall model performance:

- Actual Error – The mean error is -0.18 or an average of 18% from the actual value. The average error has a standard deviation of 41%. Fifty-three projects, or 47.7%, are projected within 25% of the actual value. Basic COCOMO (*Semi-Detached*) tied with 2 of 12 for average actual error.
- Absolute Error – Basic COCOMO (*Semi-Detached*) projected the actual project performance with an absolute error of 35% ± 27%. Basic COCOMO (*Semi-Detached*) ranked 1 of 12 for average absolute error.

- Under Estimate – The Basic COCOMO (*Semi-Detached*) projected 79 out of 111 projects under the actual value. When the Basic COCOMO (*Semi-Detached*) projects short, it does so with an average error of 37%. Basic COCOMO (*Semi-Detached*) ranked 4 of 12 for balance and 3 of 12 for average under-estimation error.
- Over Estimation – The Basic COCOMO, implementing *Semi-Detached*, over estimated thirty-two projects. The average error was 30% with a standard deviation of 38%. Basic COCOMO (*Semi-Detached*) ranked 3 of 12 for average over-estimation error.

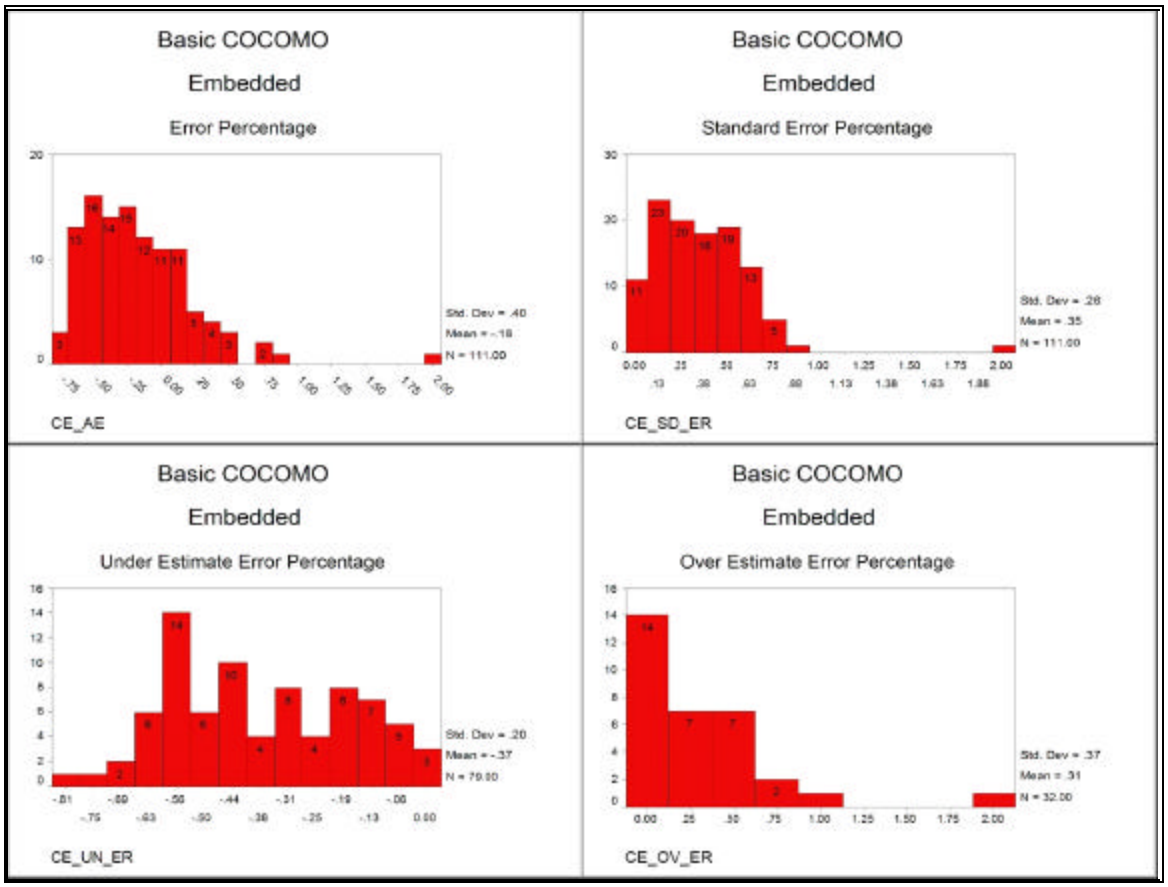



**Figure C-20. Basic COCOMO (*Embedded*) Validation Performance.**

Basic COCOMO (*Embedded*) ranked 2nd in overall model performance:

- Actual Error – The mean error is -0.18 or an average of 18% from the actual value. The average error has a standard deviation of 40%. Fifty-

four projects, or 48.6%, are projected within 25% of the actual value. Basic COCOMO (*Embedded*) tied with 2 of 12 for average actual error.

- Absolute Error – Basic COCOMO (*Embedded*) projected the actual project performance with an absolute error of 35% ± 26%. Basic COCOMO (*Embedded*) ranked 1 of 12 for average absolute error.
- Under Estimate – The Basic COCOMO (*Embedded*) projected 79 out of 111 projects under the actual value. When the Basic COCOMO (*Embedded*) projects short, it does so with an average error of 37%. Basic COCOMO (*Embedded*) ranked 4 of 12 for balance and 3 of 12 for average under-estimation error (tie with *semi-detached* in both cases).
- Over Estimation – The Basic COCOMO, implementing *Embedded*, over estimated thirty-two projects. The average error was 31% with a standard deviation of 37%. Basic COCOMO (*Embedded*) ranked 5 of 12 for average over-estimation error.

THIS PAGE INTENTIONALLY LEFT BLANK

# D. SIMULATION CALIBRATION DETAILS

The development of the SRM involved using VitéProject to simulate software development; ultimately developing a mathematical model to replicate the behavior of the simulation. To extend this body of work, the research strategy involves duplicating the initial simulation projections. Using the initial simulation projections as our baseline, the research implemented an Application Program Interface (API) to produce future VitéProject simulations with increased efficiency.

This appendix documents our attempts to duplicate the simulation data that serves as the foundation of the SRM. Supporting evidence is included to explain the discrepancies between our findings and (Nogu00). Information is also provided to aid future research conduct simulations on the VitéProject. Future research should be able to avoid some of the resource exhausting pitfalls that plagued this research.

This appendix contains support data for the benchmark projections of the *average staffing* and the minimum *effort* using the COCOMO and SSE estimation models. The appendix concludes with conclusive evidence that VitéProject is flawed and presents two theories why the development of the SRM failed to discover this flaw.

## A. VITÉPROJECT PARAMETERS

The simulation data founding the SRM was configured according to Figure D-1 (Nogu00). This research maintains the original integrity. However, the following extract from (Nogu00) needs additional explanations. This extracts provides the VitéProject parameter settings. Users should be able to duplicate the original work using Figure D-1 and the probability settings provided in the extract from (Nogu00). The remaining simulation parameters use the system provided default values.

VitéProject uses a set of default values for the variables of the model. These values are stored in a file named "behmatrx.opd" in the subdirectory of VitéProject. The behavior of the model depends on the values of these variables that are collectively called

Behavior Matrix. This Appendix discusses the concepts considered in the behavior matrix and their relationship with software projects. The simulations used the default values for this file.

- Participant attention rule: Defines the probability distribution applied to the different selection methods (e.g. priority, FIFO, LIFO, random) of picking items to process.
- Participant tool selection rules: Defines the probability distribution applied to different information exchange tools (e.g. conversation, email, fax, memo, phone, video, voice-mail) given the type of message (e.g. Exception, Decision, etc.) A tool selected for an information exchange determines (1) the time needed for the message to move from one participant to another and (2) the time the message will stay in the in-tray of the receiver participant.
- Activity Verification Failure Probability (VFP) adjustment: There are two VFP (internal and external). The internal VFP depends on the complexity of the requirement and the skills of the participants. The external VFP depends on the complexity of the solution and the skills of the participants. The processing speed of responsible participants is affected by the solution complexity and the requirement complexity.
- Activity Information Exchange Frequency adjustment: This adjustment depends on the uncertainty of the activity and the team experience.
- Participant Processing Speed adjustment: This adjustment depends on the match between the participant and activity skill requirements.
- Definition of Rework, Quick-Fix, and Ignore decisions: This matrix defines how much of the original failed work should be reworked, quick-fixed or ignored. The values depend on the following failure types:
    - Internal|Internal: Amount of rework of an activity given internal activity failure (based on VFPInternal.).
    - Internal|External: Amount of rework of an activity given external failure (based on VFP External.).
    - External|External: Amount of rework of a failure dependent activity given external failure of an independent activity (based on VFP External of the independent activity.).
- Impact of participant information exchange behavior on its VFP: This adjustment depends on the attendance or non-attendance of the participant to information exchange events related to the activities.

- Impact of participant decision-making behavior on the VFP of failed activity: This adjustment depends on the centralization level of the organization.
- The probabilities used by VitéProject were set as follows:
    - *Functional Error Rate* (0.01 low). Functional errors are the number of generated internal functional errors, shown in the Simulator Analysis Summary.
    - *Project Error Rate* (0.01 low). Project errors are the number of generated project errors, shown in the Simulator Analysis Summary.
    - *Information Exchange* (0.8 high)
    - *Noise* (0.1 normal)
- Finally there is a set of matrices to implement Project Decision Making Policies including how to determine to whom to report an exception, how to make a decision for an exception, what is the maximum time a participant will wait before it takes delegation by default.

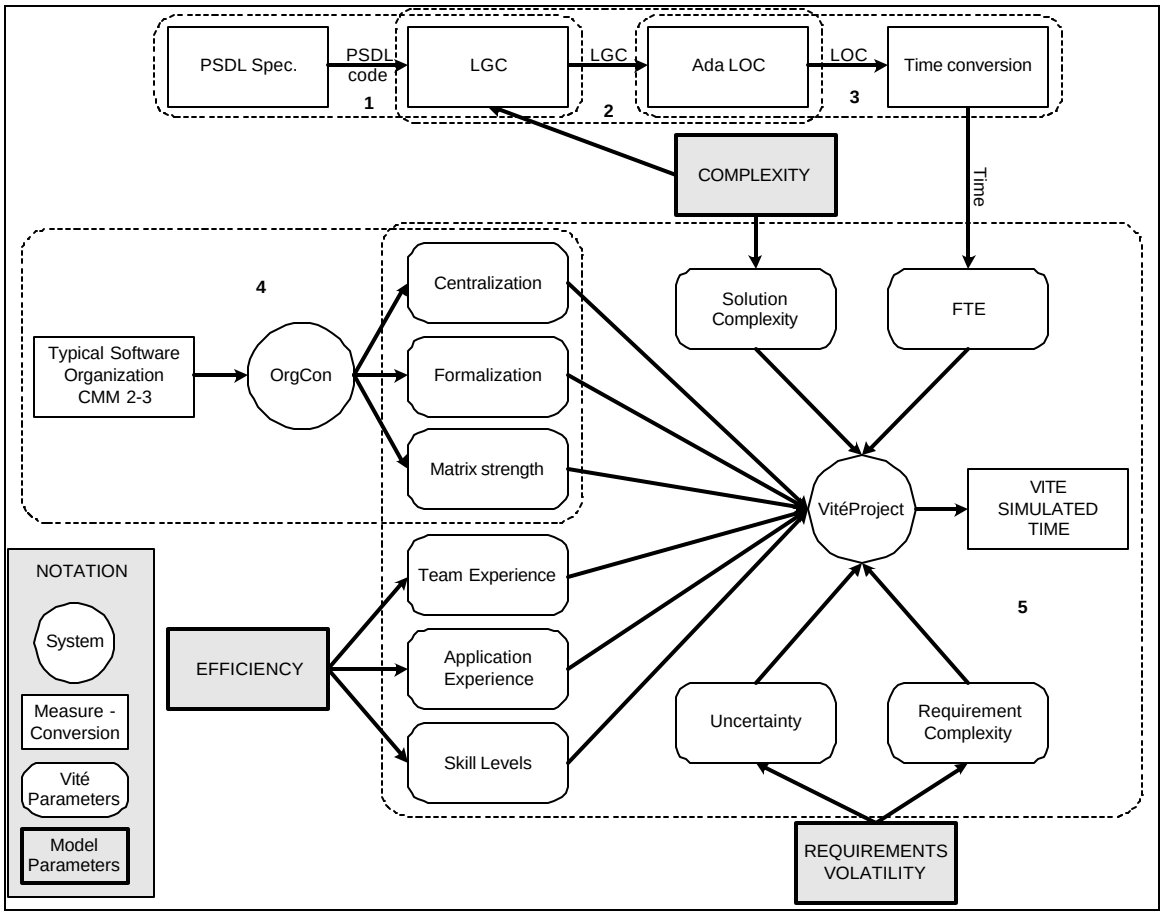



**Figure D-1. The Validation Process from "Nogu00".**

Figure D-1 demonstrates how to represent the parameters required in the SRM to the VitéProject. Consider for example *Requirements Volatility*, *RV*. The simulation setting for RV can be one of three values (Low, Medium, or High). If analysts intends on representing the RV as Medium, then set the VitéProject parameters of *uncertainty* and *requirement complexity* to Medium. This procedure is repeated for *efficiency* and *complexity.*

## B. DISCREPANCIES WITH SRM

### 1. Probability Configuration

As mentioned in Chapter V, difficulty was encountered reproducing the simulation results of the SRM. The problem first surfaced during the validation of the VitéProject API. Eventually, one source of the error was determined. The error is contained in the section titled a.VitéProject Parameters from above. Specifically the probability settings: *Functional Error Rate, Project Error Rate, Information Exchange,* and *Noise.*

Table D-1 documents 108,000 simulations conducted for eight possible VitéProject scenario probabilities ((27 base scenarios * 3000) + (9 extended scenarios * 3000)). These executions were necessary to determine the correct simulation parameters utilized to develop the SRM. The rows indicated in Table D-1 do not demonstrate all of the combinations attempted. Only included are the attempts that match the original dissertation work. For example, our simulations utilize an additional setting of medium (M); the original work uses Low and High. So, EF, RV, and CX could have three different base states instead of the original two. The original work did not use a medium (M) setting, so there is not comparison data to examine.

The cell color is determined by taking the estimated value and comparing to the equivalent value in (Nogu00). If a cell in Table D-1 is not colored, this value is greater than two standard deviations away from the expected value. Green cells are within one standard deviation and yellow cells are within two standard deviations.

The section of Table D-1 under the heading "Probability Code A" is the VitéProject result when setting up the simulation according to the values that are documented in (Nogu00). The resulting estimations, in days, are not close to the actual results when the simulation is configured accordingly. So, the conclusion is that the simulation parameters cannot be documented correctly in the dissertation.

The combinations of Probability Code were changed until the simulation produced consistent results similar to (Nogu00); each change supported by 108,000 simulations. Ultimately, the results under the heading "Probability Code G / H" were produced. These results are consistently within acceptable parameters (one standard deviation). However, these results are not close to the SRM documented results under any scenario xxHy where y>0. Testing on the simulation parameters continued and never reproduced the (Nogu00) results.

## 2. Duplication of Results

This research eventually duplicated all of the results in (Nogu00). However, not by actually conducting simulation. Table D-2 is a reprint of the simulation results that produced the SRM. Table D-2 should contain values very close to the values in Table D-1 (Probability Code A). However, it does not. Notice that Table D-1 (Probability Code G) is actually the closer match. Even using the probability parameters of *Probability Code G*, all attempts to duplicate columns xxH2.5 and xxH5 were unsuccessful in the simulation.

To reproduce Table D-2 use Microsoft® Excel, or some other spreadsheet. The columns labeled *xxHy*, where *y>0*, are produced by multiplying the entry in *xxH* by *y*. For example, Probability Code G produces an average value of 377 for scenario LHH5 (Table D-1). The SRM documentation (Table D-2) claims the value should be 638 ± 49. The simulation does not produce this value. This value was computed by taking the average value in column LHH and multiplying by 5 (i.e. 127.6 * 5 = 638).

This evidence provides discrepancies in the foundation data of the SRM. (Nogu00) does not accurately document the VitéProject parameters utilized to create the SRM. The simulation data is not entirely produced from simulations (only 50% of the data is provided by simulation, the other 50% is derived according to the previous paragraph). This single fact does not discredit the SRM; only provides supporting evidence to the weak performance of the SRM during the validation.

D - Dissertation (Tbl 5.3)

| Scenario | EF | RV | CX | D E(t) days | D SD(t) days | D CPM | D LGC | File: 161910 Probability Code A Proj E(t) days | Proj SD(t) days | Proj CPM | File: 134738 Probability Code B Proj E(t) days | Proj SD(t) days | Proj CPM | File: 144041 Probability Code C Proj E(t) days | Proj SD(t) days | Proj CPM | File: 152951 Probability Code D Proj E(t) days | Proj SD(t) days | Proj CPM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LLL | L | L | L | 88 | 5 | 67 | 746 | 68.02 | 0.09 | 67.2 | 68.01 | 0.09 | 67.2 | 76.39 | 0.21 | 67.2 | 76.4 | 0.32 | 67.2 |
| LLH | L | L | H | 101 | 6 | 67 | 781 | 68.99 | 0.11 | 67.2 | 68.96 | 0.13 | 67.2 | 93.63 | 0.72 | 67.2 | 93.33 | 0.56 | 67.2 |
| LLH5 | L | L | H5 | 507 | 31 | 335 | 3230 | 331.1 | 0.21 | 336 | 331.13 | 0.22 | 336 | 355.06 | 0.67 | 336 | 355.05 | 0.74 | 336 |
| LHL | L | H | L | 101 | 7 | 67 | 746 | 69.09 | 0.12 | 67.2 | 69.06 | 0.12 | 67.2 | 77.84 | 0.36 | 67.2 | 77.77 | 0.35 | 67.2 |
| LHH | L | H | H | 128 | 10 | 67 | 781 | 70.11 | 0.16 | 67.2 | 70.1 | 0.11 | 67.2 | 95.12 | 0.55 | 67.2 | 95.04 | 0.52 | 67.2 |
| LHH5 | L | H | H5 | 638 | 49 | 335 | 3230 | 332.36 | 0.23 | 336 | 332.41 | 0.26 | 336 | 356.12 | 0.67 | 336 | 356.06 | 0.7 | 336 |
| HLL | H | L | L | 32 | 2 | 67 | 746 | 25.68 | 0.1 | 16.8 | 25.66 | 0.09 | 16.8 | 28.87 | 0.2 | 16.8 | 28.83 | 0.14 | 16.8 |
| HLH | H | L | H | 42 | 3 | 67 | 781 | 26.14 | 0.11 | 16.8 | 26.12 | 0.13 | 16.8 | 33.59 | 0.27 | 16.8 | 33.55 | 0.29 | 16.8 |
| HLH5 | H | L | H5 | 209 | 14 | 335 | 3230 | 82.54 | 0.06 | 84 | 82.55 | 0.06 | 84 | 86.73 | 0.17 | 84 | 86.81 | 0.21 | 84 |
| HHL | H | H | L | 42 | 3 | 67 | 746 | 31.26 | 0.12 | 16.8 | 31.24 | 0.09 | 16.8 | 34.58 | 0.15 | 16.8 | 34.61 | 0.19 | 16.8 |
| HHH | H | H | H | 49 | 4 | 67 | 781 | 31.73 | 0.11 | 16.8 | 31.71 | 0.09 | 16.8 | 39.5 | 0.21 | 16.8 | 39.59 | 0.32 | 16.8 |
| HHH5 | H | H | H5 | 244 | 18 | 335 | 3230 | 82.95 | 0.07 | 84 | 82.98 | 0.09 | 84 | 87.6 | 0.24 | 84 | 87.49 | 0.2 | 84 |

D - Dissertation (Tbl 5.3)

| Scenario | EF | RV | CX | D E(t) days | D SD(t) days | D CPM | D LGC | File: 85738 Probability Code E Proj E(t) days | Proj SD(t) days | Proj CPM | File: 94448 Probability Code F Proj E(t) days | Proj SD(t) days | Proj CPM | File: 102919 Probability Code G Proj E(t) days | Proj SD(t) days | Proj CPM | File: 111949 Probability Code H Proj E(t) days | Proj SD(t) days | Proj CPM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LLL | L | L | L | 88 | 5 | 67 | 746 | 77.23 | 0.35 | 67.2 | 77.25 | 0.29 | 67.2 | 87.54 | 0.44 | 67.2 | 87.46 | 0.46 | 67.2 |
| LLH | L | L | H | 101 | 6 | 67 | 781 | 78.53 | 0.34 | 67.2 | 78.48 | 0.35 | 67.2 | 106.13 | 0.47 | 67.2 | 106.07 | 0.77 | 67.2 |
| LLH5 | L | L | H5 | 507 | 31 | 335 | 3230 | 340.26 | 0.28 | 336 | 340.33 | 0.33 | 336 | 363.98 | 0.79 | 336 | 363.89 | 0.71 | 336 |
| LHL | L | H | L | 101 | 7 | 67 | 746 | 96.07 | 0.54 | 67.2 | 95.85 | 0.52 | 67.2 | 107.67 | 0.66 | 67.2 | 107.67 | 0.8 | 67.2 |
| LHH | L | H | H | 128 | 10 | 67 | 781 | 97.61 | 0.78 | 67.2 | 97.44 | 0.7 | 67.2 | 127.48 | 0.91 | 67.2 | 126.45 | 1.06 | 67.2 |
| LHH5 | L | H | H5 | 638 | 49 | 335 | 3230 | 353.69 | 0.59 | 336 | 353.6 | 0.5 | 336 | 377.27 | 0.88 | 336 | 376.78 | 0.79 | 336 |
| HLL | H | L | L | 32 | 2 | 67 | 746 | 29.48 | 0.18 | 16.8 | 29.51 | 0.2 | 16.8 | 32.72 | 0.24 | 16.8 | 32.7 | 0.19 | 16.8 |
| HLH | H | L | H | 42 | 3 | 67 | 781 | 29.87 | 0.21 | 16.8 | 29.91 | 0.19 | 16.8 | 37.5 | 0.29 | 16.8 | 37.57 | 0.36 | 16.8 |
| HLH5 | H | L | H5 | 209 | 14 | 335 | 3230 | 84.27 | 0.11 | 84 | 84.31 | 0.1 | 84 | 88.96 | 0.22 | 84 | 89 | 0.23 | 84 |
| HHL | H | H | L | 42 | 3 | 67 | 746 | 40.67 | 0.25 | 16.8 | 40.71 | 0.27 | 16.8 | 44.17 | 0.32 | 16.8 | 44.16 | 0.42 | 16.8 |
| HHH | H | H | H | 49 | 4 | 67 | 781 | 41.08 | 0.35 | 16.8 | 41.17 | 0.31 | 16.8 | 49.39 | 0.42 | 16.8 | 49.29 | 0.34 | 16.8 |
| HHH5 | H | H | H5 | 244 | 18 | 335 | 3230 | 87.77 | 0.2 | 84 | 87.69 | 0.19 | 84 | 93.64 | 0.35 | 84 | 93.53 | 0.34 | 84 |

**Table D-1. Probability Experiments with VitéProject.**

| LLL | LLH | LLH(2.5) | LLH(5) | LHL | LHH | LHH(2.5) | LHH(5) | HLL | HLH | HLH(2.5) | HLH(5) | HHL | HHH | HHH(2.5) | HHH(5) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 78 | 91 | 228 | 455 | 91 | 108 | 270 | 540 | 29 | 37 | 93 | 185 | 37 | 43 | 108 | 215 |
| 80 | 91 | 228 | 455 | 91 | 112 | 280 | 560 | 29 | 38 | 95 | 190 | 38 | 44 | 110 | 220 |
| 81 | 93 | 233 | 465 | 91 | 115 | 288 | 575 | 30 | 38 | 95 | 190 | 38 | 45 | 113 | 225 |
| 82 | 94 | 235 | 470 | 92 | 115 | 288 | 575 | 30 | 38 | 95 | 190 | 38 | 45 | 113 | 225 |
| 82 | 94 | 235 | 470 | 93 | 118 | 295 | 590 | 30 | 39 | 98 | 195 | 39 | 45 | 113 | 225 |
| 82 | 95 | 238 | 475 | 94 | 118 | 295 | 590 | 31 | 39 | 98 | 195 | 39 | 46 | 115 | 230 |
| 83 | 95 | 238 | 475 | 95 | 120 | 300 | 600 | 31 | 40 | 100 | 200 | 39 | 46 | 115 | 230 |
| 85 | 96 | 240 | 480 | 96 | 122 | 305 | 610 | 31 | 40 | 100 | 200 | 39 | 46 | 115 | 230 |
| 85 | 97 | 243 | 485 | 96 | 123 | 308 | 615 | 31 | 40 | 100 | 200 | 39 | 46 | 115 | 230 |
| 86 | 98 | 245 | 490 | 96 | 123 | 308 | 615 | 31 | 40 | 100 | 200 | 40 | 46 | 115 | 230 |
| 87 | 98 | 245 | 490 | 96 | 124 | 310 | 620 | 31 | 40 | 100 | 200 | 40 | 46 | 115 | 230 |
| 88 | 99 | 248 | 495 | 96 | 124 | 310 | 620 | 31 | 40 | 100 | 200 | 40 | 47 | 118 | 235 |
| 88 | 100 | 250 | 500 | 98 | 125 | 313 | 625 | 32 | 41 | 103 | 205 | 41 | 48 | 120 | 240 |
| 88 | 100 | 250 | 500 | 100 | 126 | 315 | 630 | 32 | 41 | 103 | 205 | 41 | 48 | 120 | 240 |
| 88 | 101 | 253 | 505 | 100 | 127 | 318 | 635 | 32 | 41 | 103 | 205 | 41 | 48 | 120 | 240 |
| 88 | 102 | 255 | 510 | 101 | 127 | 318 | 635 | 32 | 41 | 103 | 205 | 42 | 48 | 120 | 240 |
| 89 | 102 | 255 | 510 | 101 | 128 | 320 | 640 | 32 | 42 | 105 | 210 | 42 | 48 | 120 | 240 |
| 89 | 103 | 258 | 515 | 102 | 129 | 323 | 645 | 33 | 42 | 105 | 210 | 42 | 49 | 123 | 245 |
| 90 | 104 | 260 | 520 | 102 | 129 | 323 | 645 | 33 | 43 | 108 | 215 | 43 | 49 | 123 | 245 |
| 90 | 104 | 260 | 520 | 103 | 130 | 325 | 650 | 33 | 43 | 108 | 215 | 43 | 49 | 123 | 245 |
| 90 | 107 | 268 | 535 | 104 | 131 | 328 | 655 | 33 | 44 | 110 | 220 | 43 | 50 | 125 | 250 |
| 90 | 107 | 268 | 535 | 104 | 132 | 330 | 660 | 33 | 44 | 110 | 220 | 43 | 51 | 128 | 255 |
| 91 | 107 | 268 | 535 | 107 | 134 | 335 | 670 | 34 | 45 | 113 | 225 | 44 | 51 | 128 | 255 |
| 91 | 107 | 268 | 535 | 107 | 137 | 343 | 685 | 34 | 45 | 113 | 225 | 44 | 52 | 130 | 260 |
| 92 | 107 | 268 | 535 | 108 | 138 | 345 | 690 | 34 | 45 | 113 | 225 | 44 | 54 | 135 | 270 |
| 92 | 109 | 273 | 545 | 109 | 139 | 348 | 695 | 35 | 45 | 113 | 225 | 44 | 54 | 135 | 270 |
| 93 | 110 | 275 | 550 | 110 | 139 | 348 | 695 | 35 | 45 | 113 | 225 | 45 | 54 | 135 | 270 |
| 95 | 110 | 275 | 550 | 111 | 142 | 355 | 710 | 35 | 45 | 113 | 225 | 46 | 54 | 135 | 270 |
| 97 | 111 | 278 | 555 | 113 | 143 | 358 | 715 | 35 | 46 | 115 | 230 | 46 | 55 | 138 | 275 |
| 100 | 112 | 280 | 560 | 119 | 150 | 375 | 750 | 35 | 47 | 118 | 235 | 47 | 57 | 143 | 285 |

**Table D-2. Simulation Results from "Nogu00".**

## C. BASELINE DATA

Chapter V presented the research with an unresolved issue. The logic implemented by (Nogu00) to correlate VitéProject LGC 1334 and LGC 3230 to simulation scenarios xxH2.5 and xxH5 respectively (see Chapter V, Table V-2) is not understood. This fact led to the development of a baseline to calibrate the VitéProject simulation. The projections (average staff and minimum effort) from the models of (Boeh83) and (Putn92) spawned an idea.

Since these two models project software development consistently with each other, could simulation parameters be established to duplicate the projections? Doing so would provide a baseline to conduct additional simulations and provide a level of confidence in the results. The use of a baseline would allow analyst to confidently discern the interpretation of the simulation, removing the ambiguity.

Figure D-2 and Figure D-3 are reproduced from Chapter V with the addition of trend line data. Annexes D-1 and D-2 contain the actual data calculated for the projections.

### 1. Average Staff

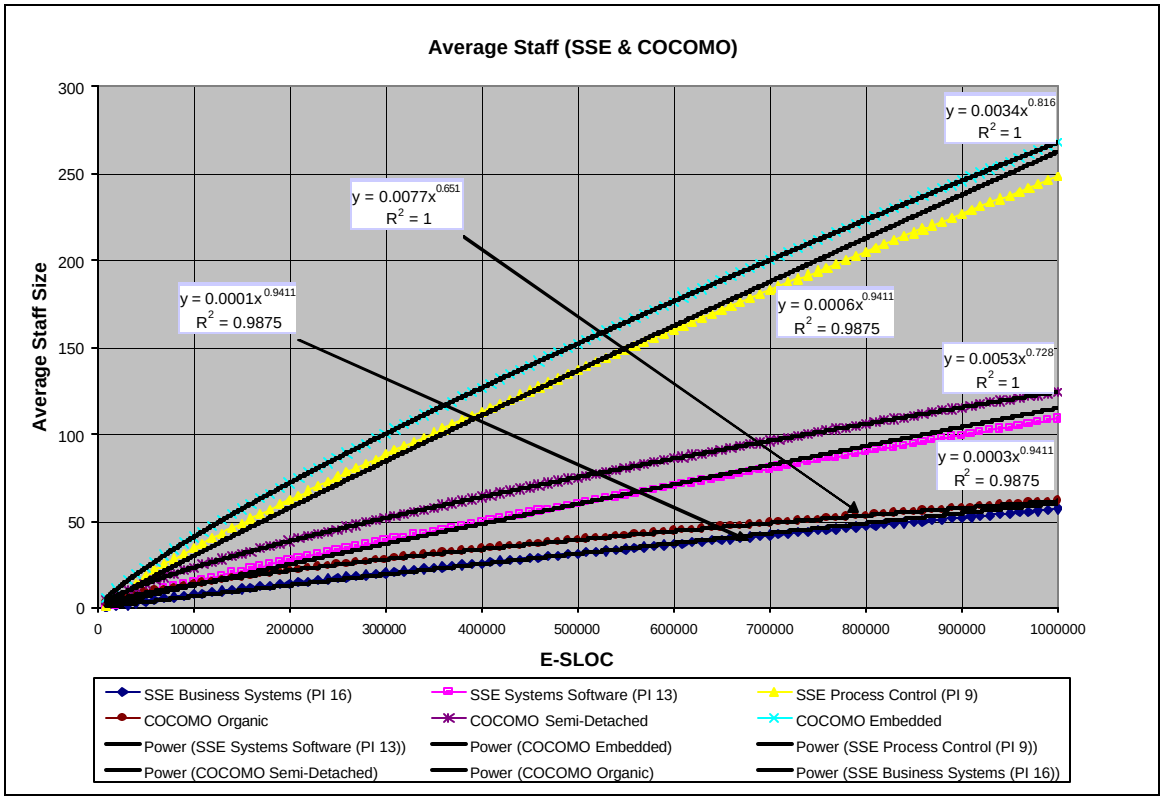



**Figure D-2. Baseline Staff Projections.**

The average staff projections indicate that a productivity index of 16 (Business Systems) is roughly equivalent to Basic COCOMO Organic projection. Systems Software (PI of 13) performs very close to Basic COCOMO Semi-Detached. And finally, a Process Control system, (PI of 9), correlates well with (Boeh81) concept of an embedded system.

Annex D-1 provides the supporting data used to project Figure D-2. The three columns representing the Simplified Software Equation projections were produced using Equation (D.1).

$$\text{Average Staff} = \frac{E}{t_d} \tag{D.1}$$

where,

$E$ is the required effort in man months
$t_d$ is the minimum development time for the main build

The three columns representing the Basic COCOMO projections were projected using the Equation (D.2).

$$\#\ \text{FTE} = \frac{\text{Effort}}{\text{Schedule}} \tag{D.2}$$

where,

# FTE is Full Time Equivalent Personnel
Effort is the required person months
Schedule is $T_{DEV}$ in months

**2. Minimum Effort**

These same correlations are evident in the projections of the minimum required effort, Figure D-3. Intuitively, the COCOMO trend lines should project "higher" than the Simplified Software Equation. (Boeh81) states the Effort (MRM) is the number of man-months estimated for the specifications and main build of the life-cycle. The Simplified

Software Equation (Putn92) states Effort (MRM) is for the main build only. Due to the differences, the COCOMO effort projections, in theory, should be larger than the Simplified Software Equation effort projection. This is evident in every projection except the embedded mode.

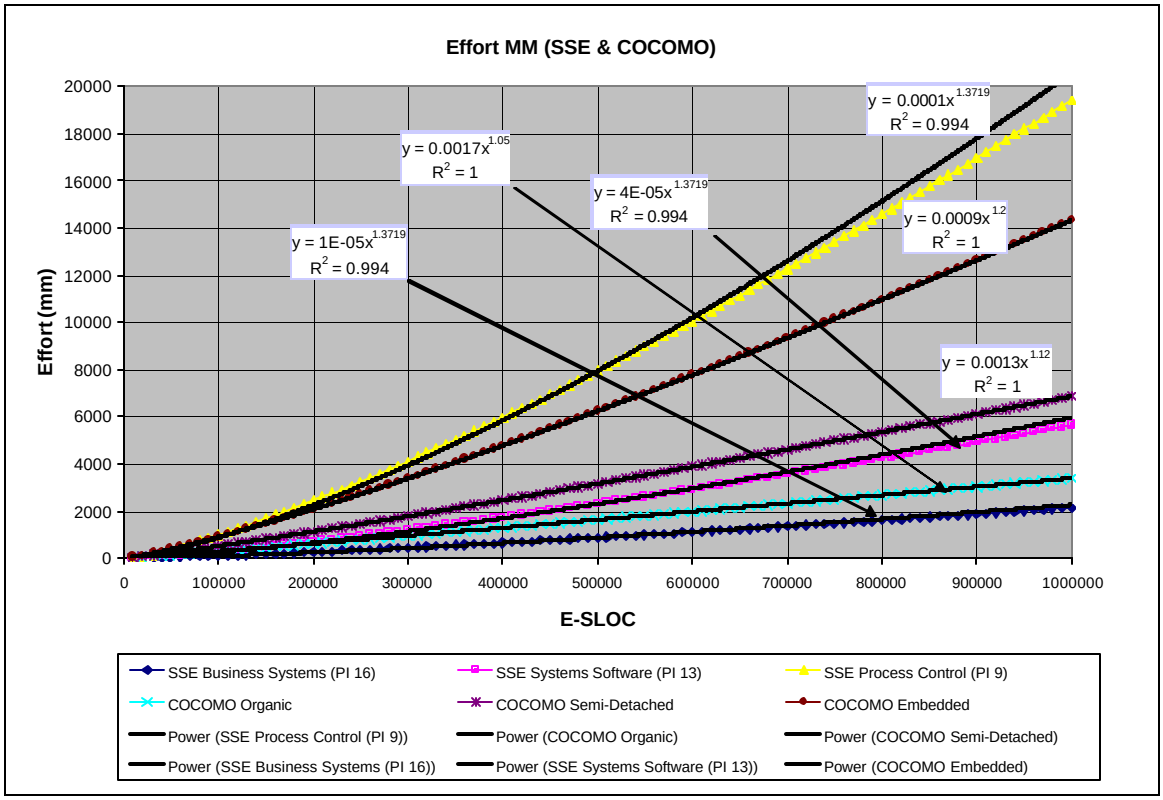



**Figure D-3. Baseline Effort Projections.**

Annex D-2 provides the supporting data for the development of Figure D-3. The three columns for the Simplified Software Equation's minimum effort were derived with the following set of equations. The first step is to select the Application Type for your specific needs. As illustrated, our baselines only consider three different *productivity parameters*: (28657, 13530, and 5186). Next, calculate the minimum development time, in months. Third, use the minimum development time to compute the development effort.

| Application Type (PP) | Minimum Time[68] ($t_{d_{\min}}$) | Development Effort[69,70] ($E$) |
|---|---|---|
| Business Systems (28657)<br>Systems Software (13530)<br>Process Control (5186) | $8.14(\frac{\text{SLOC}}{\text{PP}})^{.43}$ Months | $180Bt_d^3$ |

**Table D-3. Time and Effort of the Simplified Software Equation.**

| Table 2.1 | |
|---|---|
| **Size (SLOC)** | **B** |
| 5000 – 15000 | 0.16 |
| 15001 – 20000 | 0.18 |
| 20001 – 30000 | 0.28 |
| 30001 – 40000 | 0.34 |
| 40001 – 50000 | 0.37 |
| 50001 – 70000 | 0.39 |

**Table D-4. Table 2.1 from "Putn92".**

Basic COCOMO computes the required effort slightly different. All modes of the Basic COCOMO used fixed constants. Regardless of the software project, one of these three modes must accommodate the development. This is a potential limitation with the COCOMO equations[71]. The required development schedule can then be derive by supplying the required effort into the appropriate schedule equations; thus making the calculations a two-step process.

[68] Minimum time for development of the Main Build.

[69] $t_d$ is in years, you must take the minimum time and divide by 12.

[70] *B*, special skills factor, is determined from Table D-4.

[71] The Simplified Software Equation has the ability to adapt to evolving software development trends; more complex systems, larger systems, better tools. The Basic COCOMO is limited because there are not provisions to calibrate the equations to current software development practices.

| Mode | Effort[72] | Schedule |
|---|---|---|
| Organic | $PM = 2.4(KDSI)^{1.05}$ | $TDEV = 2.5(PM)^{0.38}$ |
| Semi-detached | $PM = 3.0(KDSI)^{1.12}$ | $TDEV = 2.5(PM)^{0.35}$ |
| Embedded | $PM = 3.6(KDSI)^{1.20}$ | $TDEV = 2.5(PM)^{0.32}$ |

**Table D-5. Effort and Schedule of the Basic COCOMO Equations.**

## D. VITÉPROJECT CALIBRATION

Duplication of the VitéProject calibration cannot occur without the proper configuration of the various simulation parameters. Most of the simulation parameters use the default values. However, several do change. Additionally, the calibration parameters are different than the parameters utilized in the creation of the SRM (Nogu00).

### 1. Probabilities

Section 0 of this Appendix illustrates the VitéProject simulation parameters utilized in the development of the SRM. This section, Section 0, documents the VitéProject simulation parameters used in the development of the calibration. The majority of the simulation settings of Section 0 were used in all the calibrations except the probabilities. The calibration maximized the stochastic behavior of the VitéProject[73].

The probabilities used by VitéProject were set as follows (Vite2.0):

[72] KDSI is thousands delivered source instructions.

[73] This is necessary because other probability settings produce a smaller standard deviation, thus increasing the number of voids in the simulation projections, Chapter V.

- *Functional Error Rate* (set to 1.0 high). Functional Error Rate is the probability that a sub activity will fail and generate rework within an activity. This probability is generally in the range 0.01 (low) to 0.10 (significant, but common). If the internal Error rate is greater than 0.20, more rework will be generated and the project may finish later. As with project errors, actors can take the following actions with project errors: rework, quick-fix, or ignore.
- *Project Error Rate* (set to 1.0 high). Project Error Rate is the probability that a sub activity will fail and generate rework for failure dependent activities. This probability is generally in the range 0.01 (low) to 0.10 (significant, but common). If the Project Error Rate is greater than 0.20, so much rework will be generated that the project may never finish. Actors can take the following actions with project errors: rework, quick-fix, or ignore.
- *Information Exchange* (set to 1.0 high). Information Exchange is the probability of generating a communication request while processing a sub activity. Generation is tried probabilistically each time any work, exception, or communication item completes. This probability is generally in the range 0.4 (for routine jobs with highly skilled workers) to 0.8 (default 0.5).
- *Noise* (set to 1.0 high). Noise is the probability that an actor is sent a noise item (a distraction from assigned activities) to process. Generation is tried probabilistically each time any work, exception, or communication item completes. This probability is generally in the range 0.01 (low, for a job with few distractions) to 0.1. A job with many significant disruptions may have an associated noise frequency of 0.2.

### 2. Visio Structure

Figure II-12. Organization Representation for VitéProject, served as the structure utilized in the calibration. However, this section provides additional information to aid future enhancements to this research.

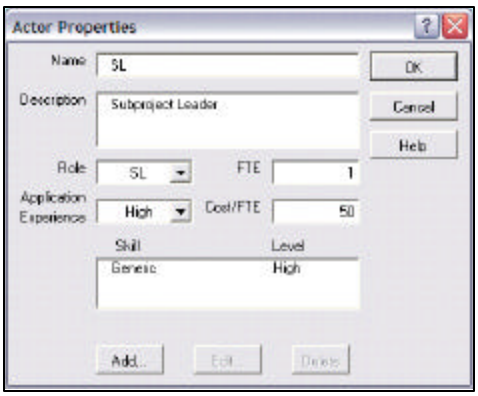



**Figure D-4. Actor Properties.**

Figure D-4 is representative of the *program manager* and *sub-project leader*. The FTE value remained one for all simulations. The use of the program manager and the sub-project leaders are for meetings, issue resolution, and general guidance.

Figure D-5 demonstrates the settings of the actual developers. The developers are the people assigned to do the work. The FTE's assigned as the developers are the total population available to conduct the work. For all of the simulation calibrations this value was set to two.

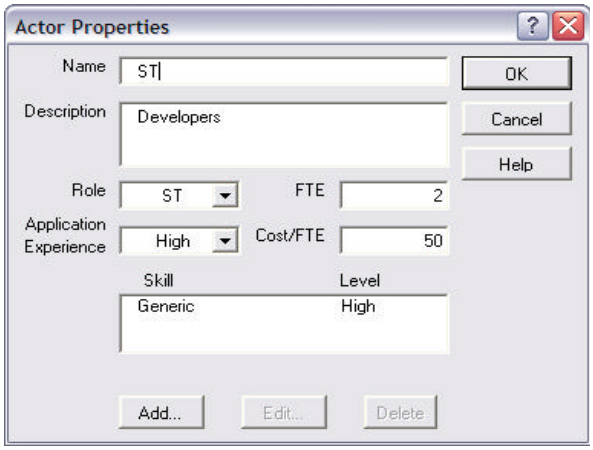



**Figure D-5. Developer Properties.**

Activities in the simulation require a specified effort for a specified amount of time, see Figure D-6. The value contained in the *work* field always receives a one for the

calibration. This requires additional explanation. The total amount of work volume required for an activity is calculated by

$$\text{duration} = \frac{\text{work volume}}{\text{FTE assigned}} \tag{D.3}$$

To adjust the activity duration, an analyst alters one of two parameters, or both. First, the value for the *work* field can change (numerator). Second, the duration is modified by adjusting the number of FTE assigned. For our calibration, the activity durations were adjusted by altering the FTE assigned[74].

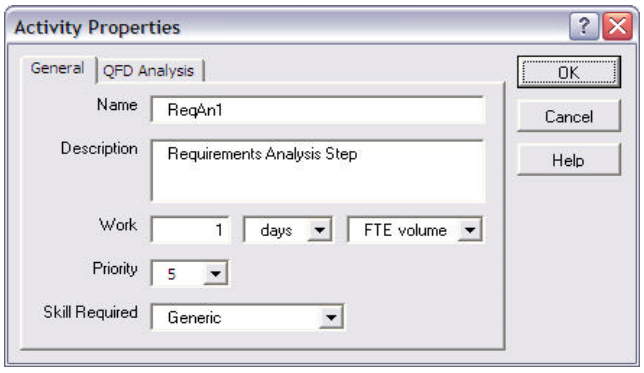



**Figure D-6. Activity Properties.**

Two additional points need to be emphasized to successfully replicate the calibration in VitéProject. To adjust the FTE assigned, an analyst must alter the assignment properties, Figure D-7. This value determines how many FTE's are assigned to work on the activity at any one time. For example, if you desire to have an activity with duration of two, assuming the work is assigned as one, the assignment properties (FTE) is set to a value of 0.5. Adjusting the VitéProject activities allow the simulator to emulate software development in the required ranges for our research. Recall that the benchmark trend lines span software development from 10K thru 1000K E-SLOC.

74 Changing the *work volume* is more convenient. However, the API has a bug that limits the ability for an activity duration to exceed 68. As a work-a-round, alter the activity durations by adjusting the FTE assigned.

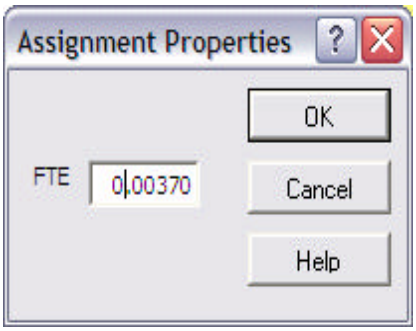



**Figure D-7. Assignment Properties.**

The distinguishing feature in the calibration is how to determine the appropriate activity duration for specific software this size? Casual analysis produced this correlation. A reasonable match was developed after several million simulation trials were examined. With the VitéProject parameters established as indicated, users can reference Table D-6.

| E-SLOC | Activity Duration | FTE Inverse |
|---|---|---|
| 6000 | 1 | 1.0000 |
| 30000 | 5 | 0.2000 |
| 60000 | 10 | 0.1000 |
| 90000 | 15 | 0.0667 |
| 120000 | 20 | 0.0500 |
| 150000 | 25 | 0.0400 |
| 180000 | 30 | 0.0333 |
| 210000 | 35 | 0.0286 |
| 240000 | 40 | 0.0250 |
| 270000 | 45 | 0.0222 |
| 300000 | 50 | 0.0200 |
| 330000 | 55 | 0.0182 |
| 360000 | 60 | 0.0167 |
| 390000 | 65 | 0.0154 |
| 420000 | 70 | 0.0143 |
| 450000 | 75 | 0.0133 |
| 480000 | 80 | 0.0125 |
| 510000 | 85 | 0.0118 |
| 540000 | 90 | 0.0111 |
| 570000 | 95 | 0.0105 |
| 600000 | 100 | 0.0100 |

**Table D-6. FTE Inverse.**

Table D-6 works as follows; an analyst determines the E-SLOC they wish to simulate, find the value on the chart. Next, set each activity duration to the corresponding value from the table. If the API is used and the desire is to have an activity duration greater than 68 (column two), then use the column FTE Inverse. FTE Inverse is the value you put in Figure D-7. Assignment Properties.

The final point requiring clarification is the work hours and work days per week. During our calibration, the work week was established to be five days and the work day to be eight hours. If the simulation is configured according to this appendix, calibration should occur with minimal confusion.

## E. ISSUES WITH VITÉPROJECT

Chapter V introduces issues with the VitéProject projections. Specifically, voids were demonstrated in the simulation projections, see Figure D-8. Interesting results were demonstrated in the simulation behavior for specific instances of E-SLOC. (Nogu00) used the Weibull equation to fit the simulation projections. Were these voids present in the original results? A series of experiments were set up to demonstrate how the voids were introduced.

The experiments involved fixing the activity durations and traversing through the 27 possible scenarios in VitéProject. To accurately account for the time projections, the simulator must represent all values within the upper and lower bounds. Any break in continuity will produce a discontinuous model.

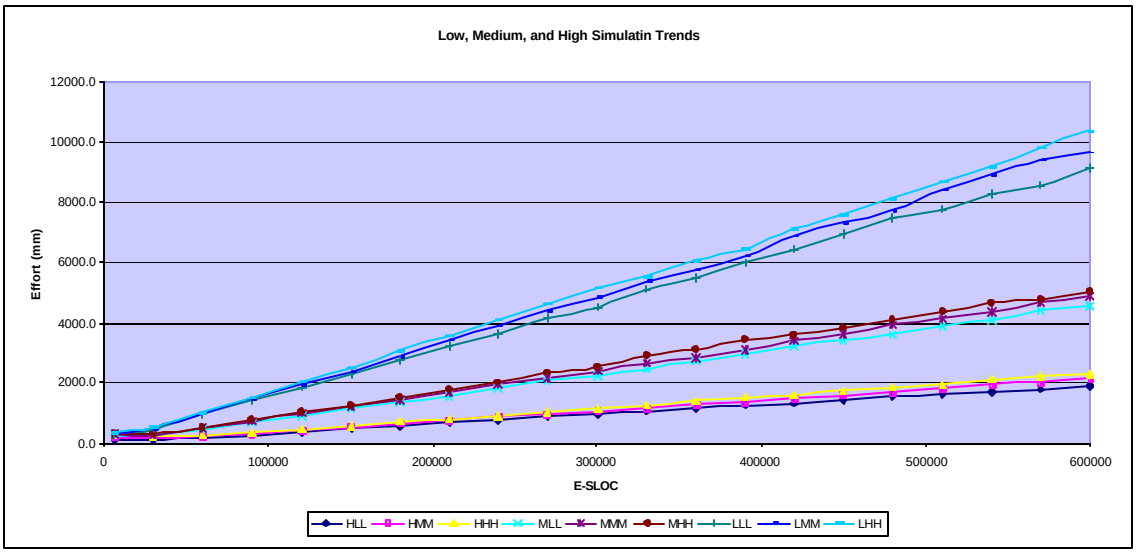



**Figure D-8. VitéProject Projections.**

(Nogu00) only exercised the simulation with *activity duration* of 1, 2.5, and 5. This research extended the *activity duration* to around 275. Ultimately, the number of the *activity duration* was reduced down to 100 in the calibrated product, Table D-6. Questions still remained as to why these voids were not discovered in the original SRM research; they were only discovered during our validation. Figure D-9 introduces our first clue.

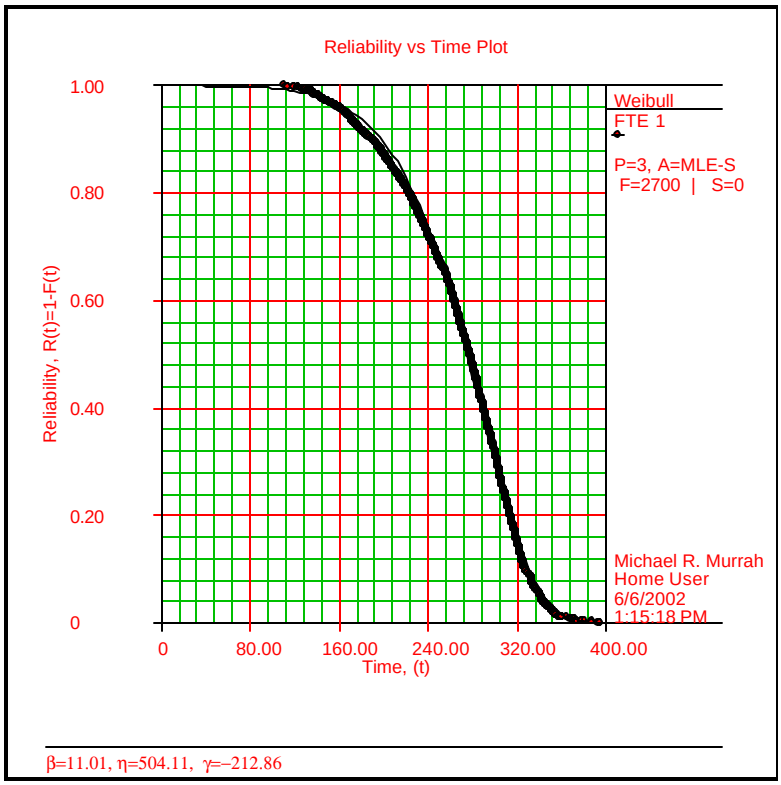



**Figure D-9. FTE = 1.**

When the simulation is configured for each activity having duration of one[75], a three-variable Weibull curve is an excellent distribution fit. The distribution has a lower bound of around 100 (scenario HLL) and an upper bound approximately 400 (scenario LHH). The simulation data for an activity duration of one (FTE = 1) produces continuous possibilities between the lower and upper bounds. There are no voids evident in the data. Figure D-10 begins to introduce some interesting behavior.

[75] When the SRM used an FTE = 1, a total of eight scenarios, each with 30 simulation executions, were generated: LLL, LLH, LHL, LHH, HLL, HLH, HHL, HHH. Figure D-9 introduces a Medium parameter, producing 27 scenarios, each with 100 simulation executions.

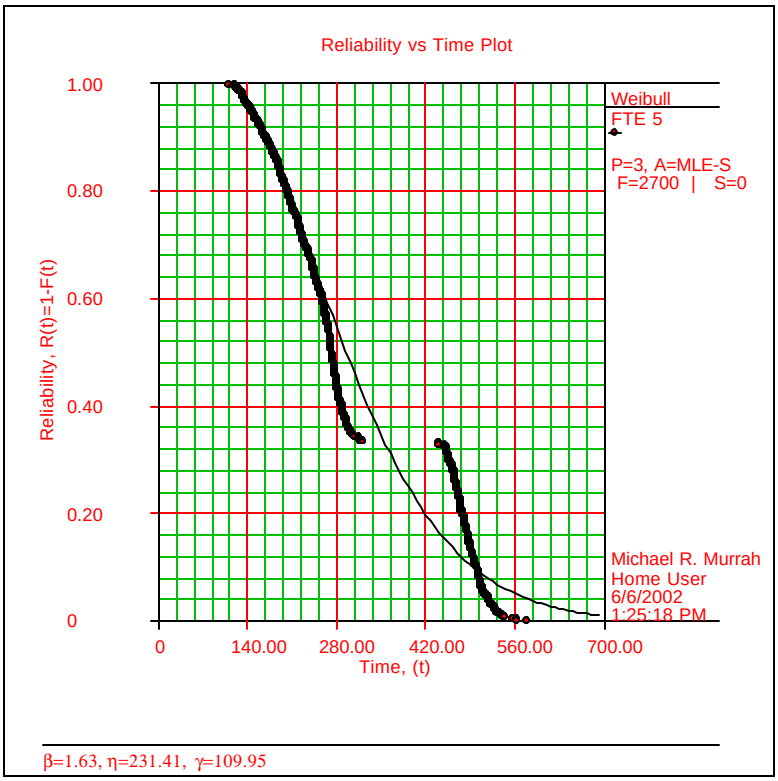



**Figure D-10. FTE = 5.**

Figure D-10 increases the individual activity durations from the one in Figure D-9 to five. Figure D-10 has a lower bound of approximately 120 and an upper bound of approximately 580. Interesting is the "break" or void between 310 and 430. Further investigation reveals this void occurs when a scenario traverses from a medium to a low efficiency scenario (MHH to LLL).

Again, why was this void not discovered in the creation of the SRM. The answer may not be fully known, but this dissertation contains evidence to support two possible theories.

- (Nogu00) documents using scenarios of xxH5. However, only four actual xxH5 scenarios were exercised (LLH5, LHH5, HLH5, HHH5). Using four values would have produced voids but these voids could have been discarded because all possible scenarios were not simulated (this research uses 27 scenarios).

- The second theory refers to the discrepancy in the SRM simulation data. Recall this research could not duplicate the results for xxHy, where y>0. Later, duplication became possible using Microsoft® Excel. If (Nogu00) had actually performed the xxHy simulations using VitéProject, the voids would have been evident.

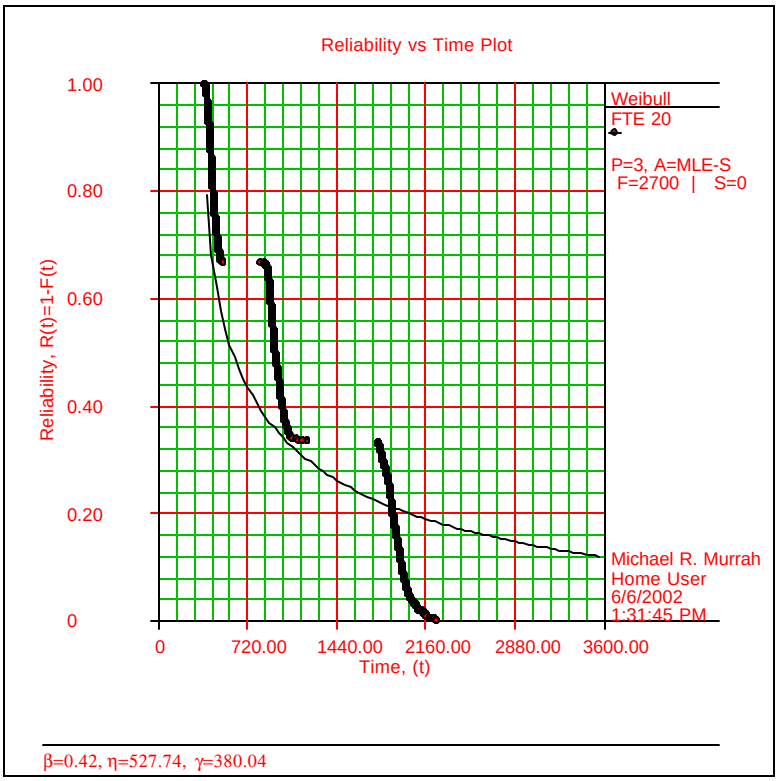



**Figure D-11. FTE = 20.**

Figure D-11 demonstrates the void propagates with higher activity durations. In this example, activity durations of 20 demonstrate two distinct voids. The voids are introduced each time the simulation changes *efficiency*. The ranges correspond to *high*, *medium*, and *low* efficiency respectively. *Recall that during the SRM validation attempts in Chapter IV, the SRM produced only two projections (low or high).*

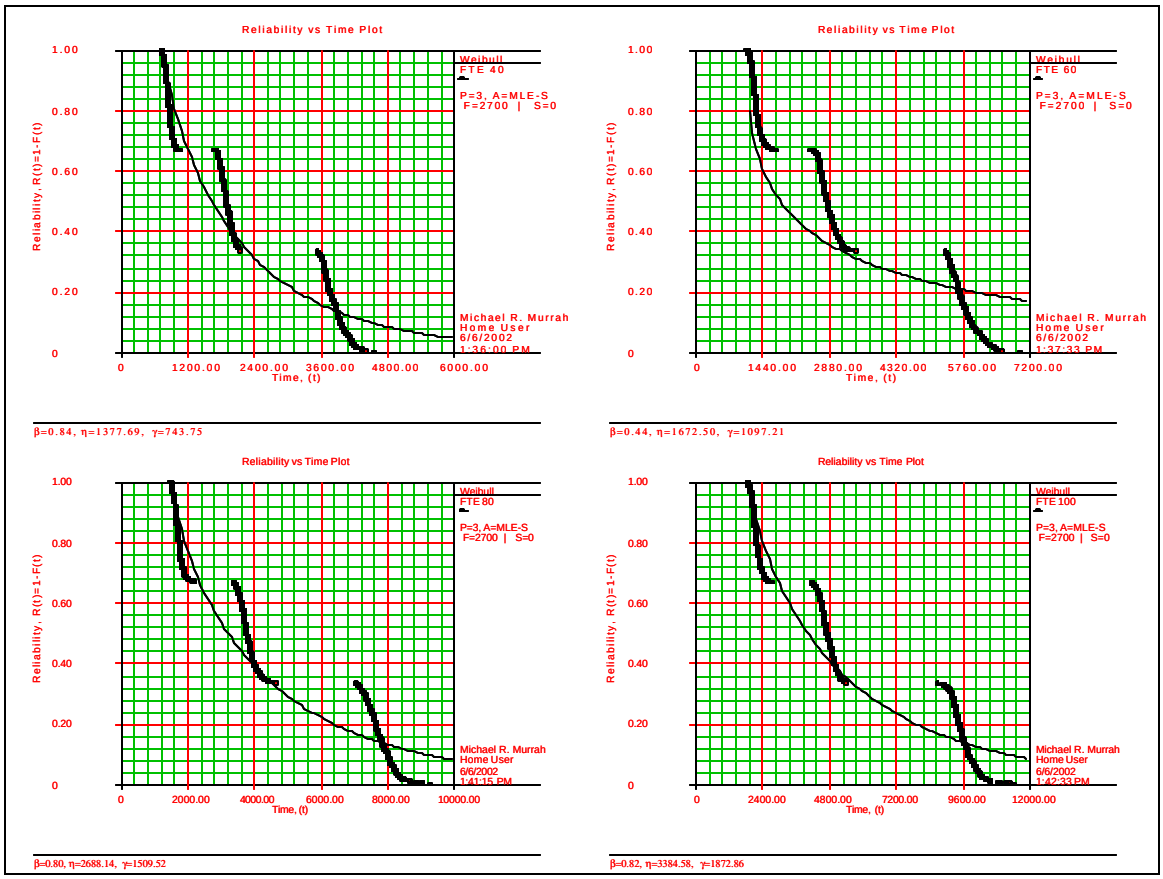



**Figure D-12. FTE = {40, 60, 80, and 100}.**

Figure D-12 demonstrates the propagation continues. From top left to bottom right: activity durations of 40, 60, 80, and 100 respectively. The patterns remain consistent however the scales are different.

The conclusion of this dissertation is that VitéProject, implemented according to (Nogu00) and this dissertation, is not suitable to simulate software development. The SRM model projects two distinct ranges of possible values. The dissertation demonstrates this is due to only using two of three efficiency ranges. However, if all three efficiency ranges would have been modeled, three distinct ranges would have been projected.

THIS PAGE INTENTIONALLY LEFT BLANK

# ANNEX D-1. DATA FILES FOR APPENDIX D

The following table is the support data for Figure D-3, Appendix D, Average Staff Baseline Projections.

| | | | SSE | SSE | SSE | | COCOMO | COCOMO | COCOMO | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Business Systems (PI 16) | Systems Software (PI 13) | Process Control (PI 9) | | Organic | Semi-Detached | Embedded | |
| **SLOC** | KSLOC | | Average Staff | Average Staff | Average Staff | | # FTE | # FTE | # FTE | |
| 10000 | 10 | | 0.446559 | 0.851494 | 1.942413 | | 3.081704 | 4.367056 | 6.2567052 | |
| 20000 | 20 | | 0.911838 | 1.73868 | 3.966248 | | 4.839065 | 7.233336 | 11.015042 | |
| 32000 | 32 | | 2.01021 | 3.833044 | 8.743875 | | 6.571184 | 10.18446 | 16.163971 | |
| 40000 | 40 | | 3.126149 | 5.960902 | 13.59791 | | 7.598573 | 11.98087 | 19.39218 | |
| 50000 | 50 | | 4.121688 | 7.859183 | 17.92824 | | 8.786591 | 14.09415 | 23.265116 | |
| 60000 | 60 | | 5.08199 | 9.690276 | 22.10529 | | 9.893898 | 16.0947 | 26.9971 | |
| 70000 | 70 | | 5.802405 | 11.06395 | 25.23891 | | 10.9383 | 18.00612 | 30.615806 | |
| 80000 | 80 | | 6.508504 | 12.41033 | 28.31024 | | 11.93171 | 19.84442 | 34.140283 | |
| 90000 | 90 | | 7.202318 | 13.73329 | 31.32815 | | 12.88258 | 21.62108 | 37.584396 | |
| 100000 | 100 | | 7.885401 | 15.03578 | 34.29938 | | 13.7972 | 23.34473 | 40.958655 | |
| 110000 | 110 | | 8.558969 | 16.32014 | 37.22922 | | 14.68039 | 25.02204 | 44.271285 | |
| 120000 | 120 | | 9.224007 | 17.58822 | 40.12196 | | 15.53595 | 26.65832 | 47.528881 | |
| 130000 | 130 | | 9.881322 | 18.84158 | 42.9811 | | 16.36696 | 28.25788 | 50.736846 | |
| 140000 | 140 | | 10.53159 | 20.0815 | 45.80958 | | 17.17593 | 29.82428 | 53.899678 | |
| 150000 | 150 | | 11.17538 | 21.30907 | 48.6099 | | 17.96496 | 31.36052 | 57.021175 | |
| 160000 | 160 | | 11.81318 | 22.52523 | 51.38418 | | 18.73584 | 32.86913 | 60.104584 | |
| 170000 | 170 | | 12.44543 | 23.73079 | 54.13427 | | 19.49006 | 34.35228 | 63.152713 | |
| 180000 | 180 | | 13.07248 | 24.92645 | 56.8618 | | 20.22895 | 35.81189 | 66.168007 | |
| 190000 | 190 | | 13.69468 | 26.11284 | 59.56818 | | 20.95364 | 37.24958 | 69.152619 | |
| 200000 | 200 | | 14.3123 | 27.29052 | 62.25468 | | 21.66514 | 38.66684 | 72.108451 | |
| 210000 | 210 | | 14.92562 | 28.45999 | 64.92244 | | 22.36432 | 40.06494 | 75.037203 | |
| 220000 | 220 | | 15.53486 | 29.62167 | 67.57246 | | 23.05197 | 41.44504 | 77.940394 | |
| 230000 | 230 | | 16.14023 | 30.77599 | 70.20567 | | 23.7288 | 42.80817 | 80.819397 | |
| 240000 | 240 | | 16.74193 | 31.9233 | 72.82289 | | 24.39543 | 44.15527 | 83.675452 | |
| 250000 | 250 | | 17.34012 | 33.06393 | 75.42488 | | 25.05243 | 45.48719 | 86.509687 | |
| 260000 | 260 | | 17.93498 | 34.19819 | 78.01234 | | 25.70032 | 46.80469 | 89.323132 | |
| 270000 | 270 | | 18.52664 | 35.32636 | 80.5859 | | 26.33958 | 48.10848 | 92.116732 | |
| 280000 | 280 | | 19.11524 | 36.4487 | 83.14615 | | 26.97061 | 49.39919 | 94.891354 | |
| 290000 | 290 | | 19.7009 | 37.56543 | 85.69362 | | 27.59383 | 50.67743 | 97.647799 | |
| 300000 | 300 | | 20.28374 | 38.67678 | 88.22883 | | 28.2096 | 51.94372 | 100.38681 | |
| 310000 | 310 | | 20.86387 | 39.78296 | 90.75222 | | 28.81824 | 53.19859 | 103.10906 | |
| 320000 | 320 | | 21.44138 | 40.88416 | 93.26425 | | 29.42006 | 54.44249 | 105.8152 | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 330000 | 330 | | 22.01637 | 41.98054 | 95.76531 | | 30.01536 | 55.67586 | 108.50583 | |
| 340000 | 340 | | 22.58893 | 43.07229 | 98.25578 | | 30.60439 | 56.89911 | 111.18149 | |
| 350000 | 350 | | 23.15913 | 44.15955 | 100.736 | | 31.18741 | 58.1126 | 113.84271 | |
| 360000 | 360 | | 23.72706 | 45.24246 | 103.2063 | | 31.76464 | 59.31671 | 116.48997 | |
| 370000 | 370 | | 24.29278 | 46.32117 | 105.6671 | | 32.3363 | 60.51174 | 119.12373 | |
| 380000 | 380 | | 24.85637 | 47.39581 | 108.1185 | | 32.90259 | 61.69803 | 121.74443 | |
| 390000 | 390 | | 25.41788 | 48.4665 | 110.561 | | 33.4637 | 62.87585 | 124.35247 | |
| 400000 | 400 | | 25.97738 | 49.53335 | 112.9946 | | 34.01982 | 64.04548 | 126.94823 | |
| 410000 | 410 | | 26.53493 | 50.59647 | 115.4198 | | 34.5711 | 65.20719 | 129.53207 | |
| 420000 | 420 | | 27.09057 | 51.65596 | 117.8367 | | 35.11771 | 66.36121 | 132.10435 | |
| 430000 | 430 | | 27.64437 | 52.71193 | 120.2456 | | 35.6598 | 67.50778 | 134.66538 | |
| 440000 | 440 | | 28.19636 | 53.76447 | 122.6466 | | 36.19751 | 68.64713 | 137.21547 | |
| 450000 | 450 | | 28.7466 | 54.81366 | 125.04 | | 36.73096 | 69.77945 | 139.75492 | |
| 460000 | 460 | | 29.29513 | 55.8596 | 127.426 | | 37.2603 | 70.90494 | 142.28401 | |
| 470000 | 470 | | 29.842 | 56.90235 | 129.8047 | | 37.78563 | 72.0238 | 144.80299 | |
| 480000 | 480 | | 30.38724 | 57.94201 | 132.1763 | | 38.30708 | 73.13621 | 147.31214 | |
| 490000 | 490 | | 30.93089 | 58.97863 | 134.5411 | | 38.82474 | 74.24232 | 149.81169 | |
| 500000 | 500 | | 31.47299 | 60.0123 | 136.899 | | 39.33874 | 75.34232 | 152.30186 | |
| 510000 | 510 | | 32.01357 | 61.04308 | 139.2504 | | 39.84916 | 76.43634 | 154.78289 | |
| 520000 | 520 | | 32.55267 | 62.07103 | 141.5954 | | 40.3561 | 77.52455 | 157.25498 | |
| 530000 | 530 | | 33.09032 | 63.09622 | 143.934 | | 40.85964 | 78.60708 | 159.71835 | |
| 540000 | 540 | | 33.62656 | 64.1187 | 146.2665 | | 41.35988 | 79.68406 | 162.17317 | |
| 550000 | 550 | | 34.1614 | 65.13854 | 148.5929 | | 41.8569 | 80.75564 | 164.61964 | |
| 560000 | 560 | | 34.69489 | 66.15578 | 150.9134 | | 42.35078 | 81.82192 | 167.05794 | |
| 570000 | 570 | | 35.22704 | 67.17048 | 153.2282 | | 42.84158 | 82.88305 | 169.48824 | |
| 580000 | 580 | | 35.75789 | 68.18269 | 155.5372 | | 43.32939 | 83.93912 | 171.91071 | |
| 590000 | 590 | | 36.28745 | 69.19247 | 157.8407 | | 43.81427 | 84.99024 | 174.32551 | |
| 600000 | 600 | | 36.81577 | 70.19985 | 160.1387 | | 44.2963 | 86.03654 | 176.73278 | |
| 610000 | 610 | | 37.34285 | 71.20488 | 162.4314 | | 44.77552 | 87.0781 | 179.13269 | |
| 620000 | 620 | | 37.86872 | 72.2076 | 164.7188 | | 45.25202 | 88.11502 | 181.52536 | |
| 630000 | 630 | | 38.39341 | 73.20807 | 167.001 | | 45.72584 | 89.14741 | 183.91095 | |
| 640000 | 640 | | 38.91693 | 74.20631 | 169.2782 | | 46.19704 | 90.17535 | 186.28957 | |
| 650000 | 650 | | 39.4393 | 75.20238 | 171.5504 | | 46.66568 | 91.19893 | 188.66137 | |
| 660000 | 660 | | 39.96056 | 76.1963 | 173.8177 | | 47.1318 | 92.21823 | 191.02647 | |
| 670000 | 670 | | 40.48071 | 77.18811 | 176.0802 | | 47.59547 | 93.23334 | 193.38498 | |
| 680000 | 680 | | 40.99977 | 78.17785 | 178.338 | | 48.05674 | 94.24434 | 195.73702 | |
| 690000 | 690 | | 41.51777 | 79.16556 | 180.5911 | | 48.51563 | 95.25131 | 198.0827 | |
| 700000 | 700 | | 42.03471 | 80.15127 | 182.8397 | | 48.97222 | 96.25431 | 200.42214 | |
| 710000 | 710 | | 42.55062 | 81.135 | 185.0838 | | 49.42653 | 97.25342 | 202.75543 | |
| 720000 | 720 | | 43.06552 | 82.1168 | 187.3235 | | 49.87862 | 98.24871 | 205.08269 | |
| 730000 | 730 | | 43.57942 | 83.09669 | 189.5588 | | 50.32852 | 99.24025 | 207.40401 | |
| 740000 | 740 | | 44.09233 | 84.07471 | 191.7898 | | 50.77627 | 100.2281 | 209.71948 | |
| 750000 | 750 | | 44.60427 | 85.05087 | 194.0166 | | 51.22192 | 101.2123 | 212.0292 | |
| 760000 | 760 | | 45.11526 | 86.02522 | 196.2393 | | 51.6655 | 102.193 | 214.33327 | |
| 770000 | 770 | | 45.62531 | 86.99777 | 198.4578 | | 52.10704 | 103.1701 | 216.63176 | |
| 780000 | 780 | | 46.13443 | 87.96855 | 200.6724 | | 52.54659 | 104.1439 | 218.92476 | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 790000 | 790 | | 46.64263 | 88.93759 | 202.8829 | | 52.98418 | 105.1142 | 221.21236 | |
| 800000 | 800 | | 47.14994 | 89.90492 | 205.0896 | | 53.41984 | 106.0812 | 223.49464 | |
| 810000 | 810 | | 47.65636 | 90.87056 | 207.2924 | | 53.8536 | 107.0449 | 225.77168 | |
| 820000 | 820 | | 48.16191 | 91.83453 | 209.4914 | | 54.28549 | 108.0054 | 228.04355 | |
| 830000 | 830 | | 48.66659 | 92.79685 | 211.6866 | | 54.71555 | 108.9627 | 230.31033 | |
| 840000 | 840 | | 49.17042 | 93.75755 | 213.8781 | | 55.14381 | 109.9168 | 232.57209 | |
| 850000 | 850 | | 49.67342 | 94.71666 | 216.066 | | 55.57029 | 110.8679 | 234.8289 | |
| 860000 | 860 | | 50.17558 | 95.67418 | 218.2503 | | 55.99503 | 111.8159 | 237.08082 | |
| 870000 | 870 | | 50.67693 | 96.63015 | 220.4311 | | 56.41804 | 112.761 | 239.32794 | |
| 880000 | 880 | | 51.17748 | 97.58458 | 222.6083 | | 56.83936 | 113.7031 | 241.57031 | |
| 890000 | 890 | | 51.67722 | 98.53749 | 224.7821 | | 57.25901 | 114.6423 | 243.80799 | |
| 900000 | 900 | | 52.17618 | 99.4889 | 226.9524 | | 57.67702 | 115.5786 | 246.04105 | |
| 910000 | 910 | | 52.67437 | 100.4388 | 229.1194 | | 58.09342 | 116.5121 | 248.26956 | |
| 920000 | 920 | | 53.17179 | 101.3873 | 231.283 | | 58.50821 | 117.4428 | 250.49356 | |
| 930000 | 930 | | 53.66846 | 102.3343 | 233.4434 | | 58.92144 | 118.3707 | 252.71311 | |
| 940000 | 940 | | 54.16437 | 103.28 | 235.6005 | | 59.33312 | 119.296 | 254.92828 | |
| 950000 | 950 | | 54.65955 | 104.2242 | 237.7544 | | 59.74328 | 120.2186 | 257.13912 | |
| 960000 | 960 | | 55.154 | 105.167 | 239.9051 | | 60.15193 | 121.1385 | 259.34568 | |
| 970000 | 970 | | 55.64773 | 106.1084 | 242.0527 | | 60.55909 | 122.0559 | 261.54801 | |
| 980000 | 980 | | 56.14075 | 107.0485 | 244.1972 | | 60.9648 | 122.9706 | 263.74617 | |
| 990000 | 990 | | 56.63306 | 107.9872 | 246.3386 | | 61.36906 | 123.8829 | 265.94021 | |
| 1000000 | 1000 | | 57.12468 | 108.9246 | 248.477 | | 61.7719 | 124.7926 | 268.13018 | |

THIS PAGE INTENTIONALLY LEFT BLANK

# ANNEX D-2. DATA FILES FOR APPENDIX D

The following table is the support data for Figure D-4, Appendix D, Baseline Effort Projections.

| | | SSE | SSE | SSE | COCOMO | COCOMO | COCOMO |
|---|---|---|---|---|---|---|---|
| | | Business Systems (PI 16) | Systems Software (PI 13) | Process Control (PI 9) | Organic | Semi-Detached | Embedded |
| **SLOC** | KSLOC | E (MM) | E (MM) | E (MM) | PM | PM | PM |
| 10000 | 10 | 2.311502 | 6.086222 | 20.9695 | 26.92844 | 39.5477 | 57.056155 |
| 20000 | 20 | 6.358806 | 16.74284 | 57.68584 | 55.75614 | 85.95573 | 131.08062 |
| 32000 | 32 | 16.68856 | 43.94124 | 151.3953 | 91.33111 | 145.5088 | 230.4 |
| 40000 | 40 | 29.37045 | 77.33287 | 266.443 | 115.4448 | 186.8222 | 301.14419 |
| 50000 | 50 | 42.62333 | 112.2279 | 386.6705 | 145.925 | 239.8654 | 393.61035 |
| 60000 | 60 | 56.84 | 149.6606 | 515.6414 | 176.7136 | 294.2054 | 489.87356 |
| 70000 | 70 | 69.34504 | 182.5866 | 629.0847 | 207.7611 | 349.648 | 589.4136 |
| 80000 | 80 | 82.38062 | 216.9095 | 747.3409 | 239.0318 | 406.0524 | 691.84767 |
| 90000 | 90 | 95.8985 | 252.5023 | 869.9724 | 270.4991 | 463.3113 | 796.88107 |
| 100000 | 100 | 109.8598 | 289.2627 | 996.627 | 302.1421 | 521.3402 | 904.27912 |
| 110000 | 110 | 124.2326 | 327.1064 | 1127.014 | 333.9439 | 580.0709 | 1013.85 |
| 120000 | 120 | 138.9897 | 365.9622 | 1260.888 | 365.8909 | 639.4466 | 1125.4339 |
| 130000 | 130 | 154.1082 | 405.7695 | 1398.04 | 397.9713 | 699.4196 | 1238.8951 |
| 140000 | 140 | 169.5681 | 446.4755 | 1538.289 | 430.1755 | 759.9494 | 1354.1169 |
| 150000 | 150 | 185.3517 | 488.0341 | 1681.475 | 462.495 | 821.0006 | 1470.9978 |
| 160000 | 160 | 201.4437 | 530.4045 | 1827.458 | 494.9225 | 882.5425 | 1589.4486 |
| 170000 | 170 | 217.8302 | 573.5503 | 1976.113 | 527.4516 | 944.548 | 1709.3902 |
| 180000 | 180 | 234.4987 | 617.4389 | 2127.327 | 560.0765 | 1006.993 | 1830.752 |
| 190000 | 190 | 251.4381 | 662.0405 | 2280.997 | 592.7923 | 1069.856 | 1953.4703 |
| 200000 | 200 | 268.6381 | 707.3284 | 2437.033 | 625.5942 | 1133.117 | 2077.4879 |
| 210000 | 210 | 286.0895 | 753.2781 | 2595.348 | 658.4784 | 1196.759 | 2202.7523 |
| 220000 | 220 | 303.7835 | 799.8668 | 2755.865 | 691.4409 | 1260.766 | 2329.2157 |
| 230000 | 230 | 321.7124 | 847.074 | 2918.512 | 724.4784 | 1325.124 | 2456.8345 |
| 240000 | 240 | 339.8689 | 894.8803 | 3083.225 | 757.5879 | 1389.818 | 2585.5682 |
| 250000 | 250 | 358.2462 | 943.2679 | 3249.939 | 790.7665 | 1454.836 | 2715.3794 |
| 260000 | 260 | 376.8379 | 992.2202 | 3418.6 | 824.0115 | 1520.167 | 2846.2334 |
| 270000 | 270 | 395.6382 | 1041.722 | 3589.152 | 857.3205 | 1585.801 | 2978.0982 |
| 280000 | 280 | 414.6415 | 1091.758 | 3761.547 | 890.6912 | 1651.727 | 3110.9436 |
| 290000 | 290 | 433.8428 | 1142.315 | 3935.737 | 924.1216 | 1717.936 | 3244.7415 |
| 300000 | 300 | 453.237 | 1193.38 | 4111.678 | 957.6097 | 1784.42 | 3379.4654 |
| 310000 | 310 | 472.8197 | 1244.942 | 4289.328 | 991.1537 | 1851.17 | 3515.0907 |
| 320000 | 320 | 492.5864 | 1296.988 | 4468.648 | 1024.752 | 1918.179 | 3651.5939 |
| 330000 | 330 | 512.5331 | 1349.508 | 4649.6 | 1058.403 | 1985.44 | 3788.9531 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 340000 | 340 | | 532.6559 | 1402.491 | 4832.15 | | 1092.104 | 2052.947 | 3927.1474 |
| 350000 | 350 | | 552.9511 | 1455.929 | 5016.264 | | 1125.856 | 2120.691 | 4066.1571 |
| 360000 | 360 | | 573.4152 | 1509.811 | 5201.91 | | 1159.655 | 2188.669 | 4205.9635 |
| 370000 | 370 | | 594.0448 | 1564.129 | 5389.058 | | 1193.502 | 2256.873 | 4346.5489 |
| 380000 | 380 | | 614.8367 | 1618.875 | 5577.679 | | 1227.394 | 2325.299 | 4487.8963 |
| 390000 | 390 | | 635.788 | 1674.04 | 5767.744 | | 1261.331 | 2393.942 | 4629.9897 |
| 400000 | 400 | | 656.8956 | 1729.617 | 5959.228 | | 1295.312 | 2462.796 | 4772.8138 |
| 410000 | 410 | | 678.1568 | 1785.598 | 6152.106 | | 1329.335 | 2531.857 | 4916.3538 |
| 420000 | 420 | | 699.569 | 1841.976 | 6346.353 | | 1363.399 | 2601.12 | 5060.5958 |
| 430000 | 430 | | 721.1295 | 1898.745 | 6541.946 | | 1397.504 | 2670.582 | 5205.5263 |
| 440000 | 440 | | 742.8359 | 1955.899 | 6738.863 | | 1431.649 | 2740.238 | 5351.1326 |
| 450000 | 450 | | 764.6859 | 2013.43 | 6937.081 | | 1465.833 | 2810.084 | 5497.4023 |
| 460000 | 460 | | 786.6772 | 2071.334 | 7136.582 | | 1500.054 | 2880.116 | 5644.3235 |
| 470000 | 470 | | 808.8075 | 2129.603 | 7337.344 | | 1534.313 | 2950.332 | 5791.885 |
| 480000 | 480 | | 831.0749 | 2188.233 | 7539.349 | | 1568.608 | 3020.727 | 5940.0758 |
| 490000 | 490 | | 853.4772 | 2247.219 | 7742.578 | | 1602.939 | 3091.298 | 6088.8854 |
| 500000 | 500 | | 876.0124 | 2306.555 | 7947.014 | | 1637.306 | 3162.042 | 6238.3036 |
| 510000 | 510 | | 898.6788 | 2366.235 | 8152.638 | | 1671.706 | 3232.956 | 6388.3207 |
| 520000 | 520 | | 921.4744 | 2426.257 | 8359.436 | | 1706.14 | 3304.038 | 6538.9274 |
| 530000 | 530 | | 944.3975 | 2486.614 | 8567.39 | | 1740.608 | 3375.283 | 6690.1144 |
| 540000 | 540 | | 967.4464 | 2547.302 | 8776.485 | | 1775.108 | 3446.69 | 6841.8731 |
| 550000 | 550 | | 990.6194 | 2608.317 | 8986.706 | | 1809.639 | 3518.256 | 6994.1949 |
| 560000 | 560 | | 1013.915 | 2669.654 | 9198.038 | | 1844.203 | 3589.978 | 7147.0716 |
| 570000 | 570 | | 1037.331 | 2731.31 | 9410.468 | | 1878.797 | 3661.855 | 7300.4953 |
| 580000 | 580 | | 1060.867 | 2793.28 | 9623.981 | | 1913.421 | 3733.882 | 7454.4584 |
| 590000 | 590 | | 1084.521 | 2855.562 | 9838.565 | | 1948.076 | 3806.059 | 7608.9532 |
| 600000 | 600 | | 1108.292 | 2918.15 | 10054.21 | | 1982.76 | 3878.383 | 7763.9727 |
| 610000 | 610 | | 1132.177 | 2981.041 | 10270.89 | | 2017.472 | 3950.851 | 7919.5098 |
| 620000 | 620 | | 1156.177 | 3044.232 | 10488.61 | | 2052.213 | 4023.462 | 8075.5578 |
| 630000 | 630 | | 1180.289 | 3107.719 | 10707.35 | | 2086.983 | 4096.214 | 8232.1099 |
| 640000 | 640 | | 1204.512 | 3171.499 | 10927.1 | | 2121.779 | 4169.105 | 8389.1598 |
| 650000 | 650 | | 1228.845 | 3235.569 | 11147.84 | | 2156.603 | 4242.132 | 8546.7014 |
| 660000 | 660 | | 1253.287 | 3299.925 | 11369.58 | | 2191.454 | 4315.295 | 8704.7284 |
| 670000 | 670 | | 1277.837 | 3364.565 | 11592.29 | | 2226.331 | 4388.59 | 8863.235 |
| 680000 | 680 | | 1302.493 | 3429.485 | 11815.96 | | 2261.234 | 4462.017 | 9022.2155 |
| 690000 | 690 | | 1327.255 | 3494.683 | 12040.6 | | 2296.163 | 4535.574 | 9181.6643 |
| 700000 | 700 | | 1352.121 | 3560.155 | 12266.17 | | 2331.118 | 4609.258 | 9341.5759 |
| 710000 | 710 | | 1377.09 | 3625.899 | 12492.69 | | 2366.097 | 4683.07 | 9501.9452 |
| 720000 | 720 | | 1402.161 | 3691.912 | 12720.13 | | 2401.101 | 4757.005 | 9662.7668 |
| 730000 | 730 | | 1427.333 | 3758.192 | 12948.49 | | 2436.129 | 4831.065 | 9824.0357 |
| 740000 | 740 | | 1452.606 | 3824.735 | 13177.76 | | 2471.181 | 4905.246 | 9985.7471 |
| 750000 | 750 | | 1477.978 | 3891.54 | 13407.93 | | 2506.257 | 4979.547 | 10147.896 |
| 760000 | 760 | | 1503.448 | 3958.603 | 13638.99 | | 2541.356 | 5053.968 | 10310.478 |
| 770000 | 770 | | 1529.016 | 4025.923 | 13870.93 | | 2576.478 | 5128.506 | 10473.489 |
| 780000 | 780 | | 1554.68 | 4093.497 | 14103.75 | | 2611.624 | 5203.16 | 10636.923 |
| 790000 | 790 | | 1580.44 | 4161.323 | 14337.44 | | 2646.791 | 5277.93 | 10800.777 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 800000 | 800 | | 1606.294 | 4229.398 | 14571.99 | | 2681.981 | 5352.813 | 10965.047 | |
| 810000 | 810 | | 1632.242 | 4297.72 | 14807.38 | | 2717.193 | 5427.808 | 11129.727 | |
| 820000 | 820 | | 1658.284 | 4366.287 | 15043.63 | | 2752.427 | 5502.915 | 11294.815 | |
| 830000 | 830 | | 1684.417 | 4435.097 | 15280.7 | | 2787.682 | 5578.131 | 11460.306 | |
| 840000 | 840 | | 1710.642 | 4504.149 | 15518.61 | | 2822.958 | 5653.457 | 11626.196 | |
| 850000 | 850 | | 1736.958 | 4573.439 | 15757.35 | | 2858.256 | 5728.89 | 11792.482 | |
| 860000 | 860 | | 1763.364 | 4642.965 | 15996.89 | | 2893.574 | 5804.429 | 11959.159 | |
| 870000 | 870 | | 1789.859 | 4712.727 | 16237.25 | | 2928.913 | 5880.074 | 12126.224 | |
| 880000 | 880 | | 1816.442 | 4782.721 | 16478.41 | | 2964.272 | 5955.824 | 12293.674 | |
| 890000 | 890 | | 1843.113 | 4852.947 | 16720.37 | | 2999.651 | 6031.677 | 12461.505 | |
| 900000 | 900 | | 1869.872 | 4923.402 | 16963.11 | | 3035.05 | 6107.632 | 12629.714 | |
| 910000 | 910 | | 1896.716 | 4994.084 | 17206.64 | | 3070.469 | 6183.689 | 12798.297 | |
| 920000 | 920 | | 1923.646 | 5064.992 | 17450.94 | | 3105.907 | 6259.846 | 12967.25 | |
| 930000 | 930 | | 1950.662 | 5136.123 | 17696.02 | | 3141.365 | 6336.102 | 13136.572 | |
| 940000 | 940 | | 1977.761 | 5207.477 | 17941.86 | | 3176.841 | 6412.457 | 13306.258 | |
| 950000 | 950 | | 2004.945 | 5279.051 | 18188.47 | | 3212.336 | 6488.909 | 13476.305 | |
| 960000 | 960 | | 2032.211 | 5350.844 | 18435.82 | | 3247.851 | 6565.458 | 13646.711 | |
| 970000 | 970 | | 2059.56 | 5422.855 | 18683.93 | | 3283.383 | 6642.103 | 13817.472 | |
| 980000 | 980 | | 2086.991 | 5495.081 | 18932.78 | | 3318.934 | 6718.842 | 13988.585 | |
| 990000 | 990 | | 2114.503 | 5567.521 | 19182.36 | | 3354.503 | 6795.676 | 14160.048 | |
| 1000000 | 1000 | | 2142.096 | 5640.173 | 19432.68 | | 3390.09 | 6872.603 | 14331.858 | |

THIS PAGE INTENTIONALLY LEFT BLANK

# E. MRM DEVELOPMENT DETAILS

This appendix provides direct support to Chapter VI. Section A presents an overview of the 1942 projects used in the development of the MRM. This section presents a series of illustrations for the: application type, productivity index ranges, requirements growth, phase effort and duration, application type effort, overrun data, and main build trend line data.

Section B contains information about the specific characteristics of the sixteen trend lines used to model the MRM. This section reviews the sixteen trend line categories and provides illustrations and data for: number of projects per application type, number of projects per productivity index category, average trend effort, and duration by phase. Annex E-1 is included for specific data pertaining to the primary language used in the applications.

Section C details additional trend specifics for the *efficiency* categories. Individual efficiency projections are illustrated for the entire subset of projects. Section D provides another perspective of the sixteen trend lines. This section illustrates the *functional complexity* trends.

Section E concludes this appendix by illustrating the derivation of the *alpha*, *beta*, and *gamma* input parameters to the three-variable Weibull function. Individual plots are provided for the probability and the cumulative distribution functions.

## A. PROJECT SUBSET OVERVIEW

Figure E-1 is reproduced from Chapter VI and details the ten available application types. Appendix A contains detailed descriptions of each application type. The lower half of the illustration presents the number of applications in the subset for each productivity index.

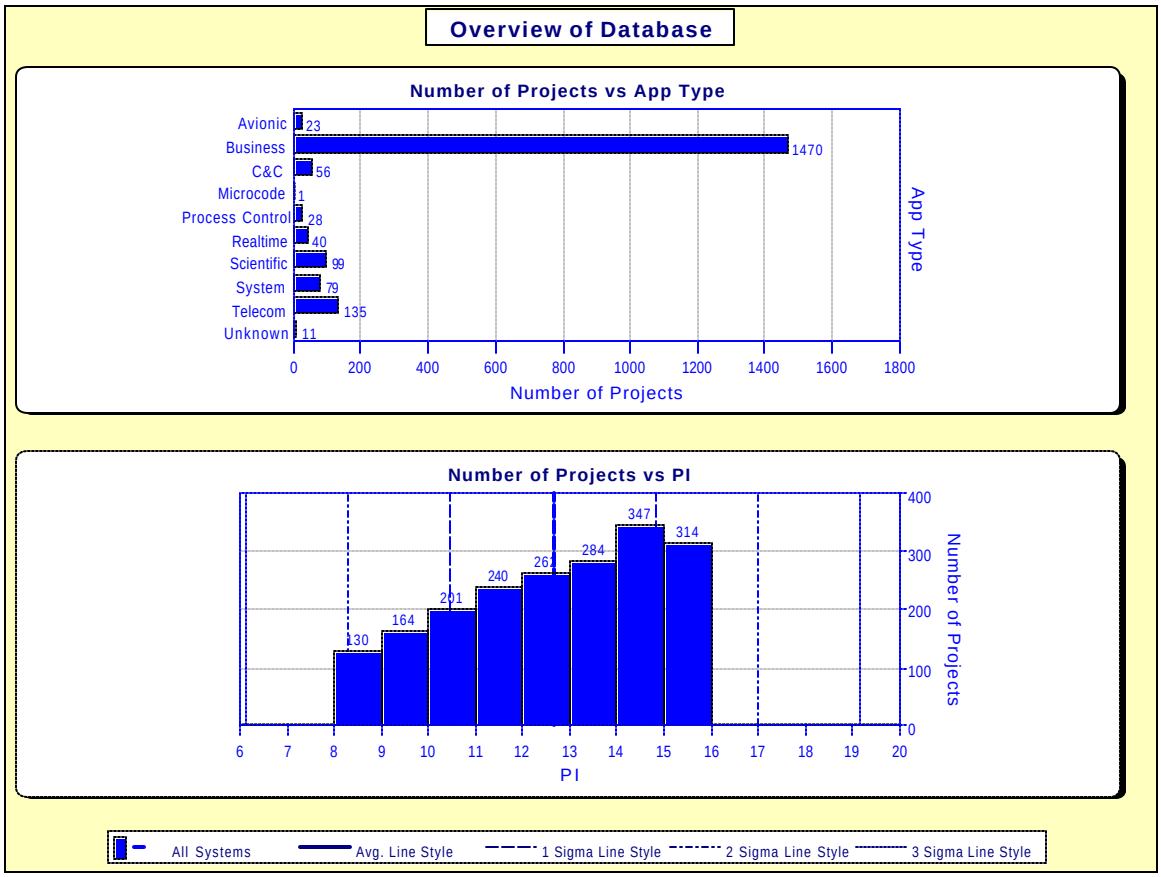



**Figure E-1. Overview of Project Subset.**

There exists a total of 1942 projects in the subset. The vast majority of these projects are business systems (1470). One microcode application is represented in the subset.

The average Productivity Index in the project subset is 12.65 with a 2.17 standard deviation. Eight different index ranges were provided by QSM®.

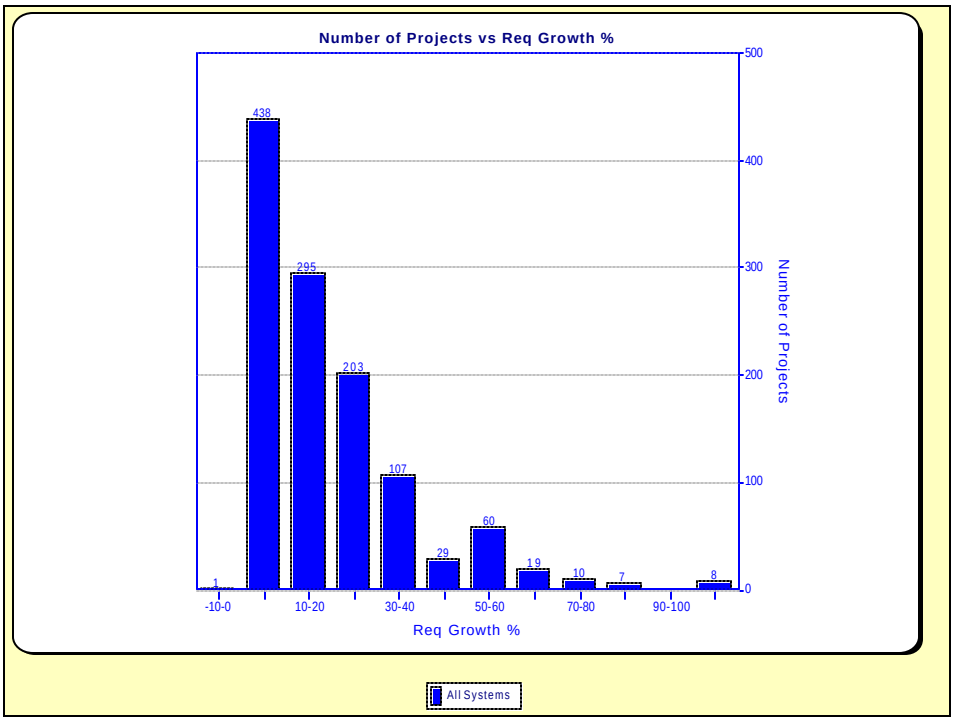



**Figure E-2. Number of Project vs. Requirements Growth.**

The project subset contains information documenting the volatility of the requirements during the project's development. However, not every project contains the requirements information; requirements are not a primary data field in the QSM® database. As mentioned in Chapter VI, over 40% of the subset would be discarded if requirement volatility became a trend line consideration.

Figure E-2 does provide some interesting insights though. Of the 1177 projects that provided requirements information, 88% experienced between zero and 40 percent increase in requirements. Only one project had their requirement's scope reduced, leaving 20% of the projects increasing the requirements upwards of 60 and 80 percent.

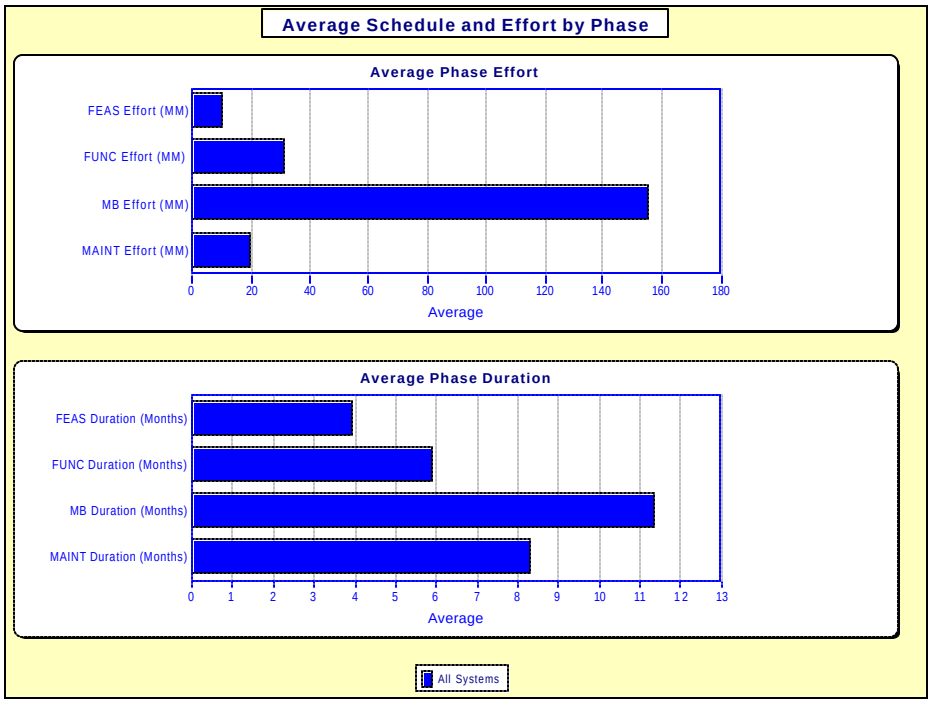



**Figure E-3. Effort and Duration by Phase.**

QSM® represents the software lifecycle in four phases, (Appendix A). Figure E-3 presents the average effort expended and average months to complete each phase. As with Figure E-2, information could be incomplete for some of the projects. The *main build* is the only phase of software development that each project must provide information. The main build data is the data that supports the trend line development of the MRM.

The *feasibility study* consumed about 4.5% of the total effort and 15% of the total time. The *functional design* required 14% of the total effort and 20% of the total time. Surprisingly, the *main build* consumed 72% of the total effort and only 37% of the total time. This is largely due to the implemented staffing profiles. The *maintenance* phase is a minimal effort actively that consumes more than a fourth of the total duration.

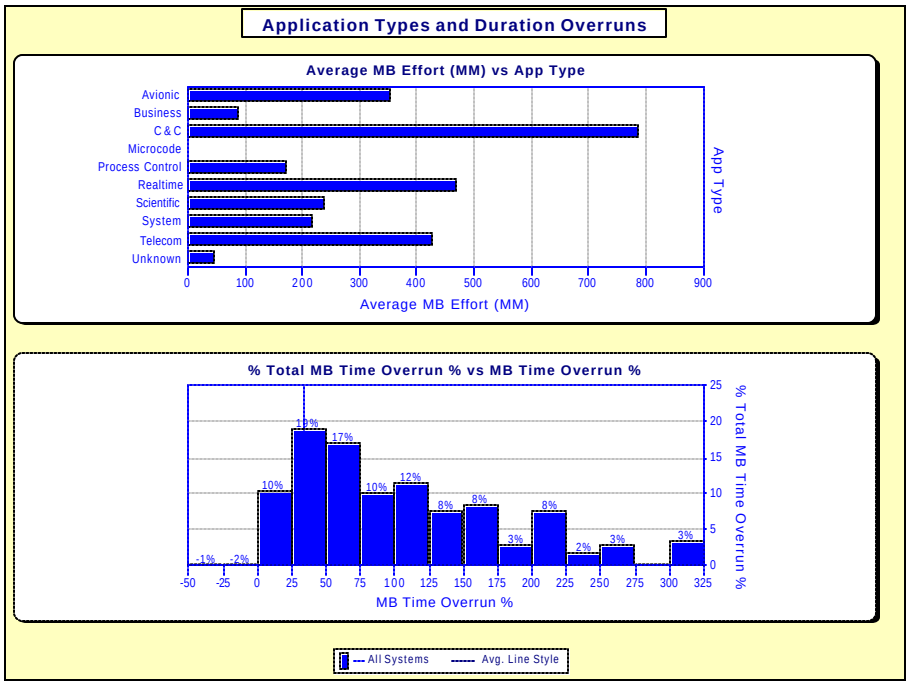



**Figure E-4. MB Effort and Develop Time Overruns.**

The top portion of Figure E-4 illustrates the average effort expended during the *main build* phase of the software development. The *command and control* application required, on average, the most development effort; however this illustration does not provide insight to the size of the project or the efficiency of the organization performing the development.

Three percent of the projects completed their *main build* ahead of their allocated schedules. Ten percent of the applications exceeded their *main build* developments between zero and 25%. Over 40% of the developments exceeded their original *main build* development schedules over 100%; three percent exceeded over 300%.

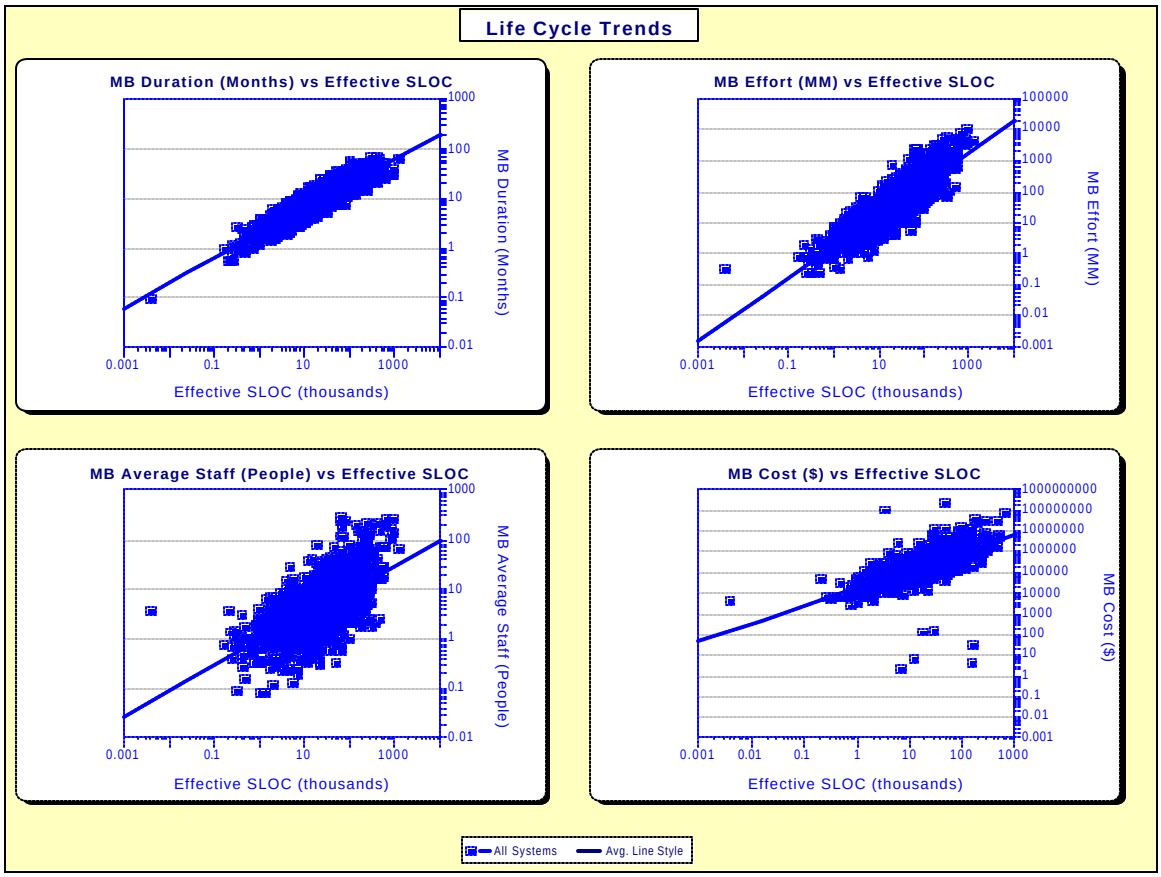



**Figure E-5. Life Cycle Trends.**

Figure E-5 provides an excellent consolidation of four measures. Beginning with the top-left, the trend for the indicated measure is projected against a specified functional size of E-SLOC. Continuing left to right then top to bottom, the same is provided for: *duration* (months), *effort* (man-months), *average staff*, and *cost* per E-SLOC. Table E-1 provides a summary of the power equations and the respective $R^2$ values.

| Measure | Power Equation ( $y = Bx^A$ ) | $R^2$ |
|---|---|---|
| Duration (months) | A=0.4994 B=0.0627 | 0.8041 |
| Effort (man-months) | A=1.0073 B=0.0016 | 0.7472 |
| Staff | A=0.5078 B=0.0270 | 0.3854 |
| Cost per E-SLOC | A=0.8585 B=46.583 | 0.4148 |

**Table E-1. Trend Line Equations for Main Build.**

## B. CATEGORY SPECIFICS

Table E-2 presents the trend line categories. There exist four groupings of the *application types*. There also exist four groups of *efficiencies*. Combining all the elements from these two groups of four, produces $4^2 = 16$ possible combinations; hence sixteen trend lines represent the foundation of the MRM model. The trend line is identified by the *functional complexity* and the *efficiency* range; i.e. [Type A (14/15)] indicates the trend line was produced from all of the *microcode, avionic,* and *real time systems* projects developed from an organization with a productivity index of 14 or 15 (high efficiency).

| **Functional Complexity Category** | **Application Type** | **Efficiency Ranges** |
|---|---|---|
| Type A | Microcode, Avionic, and Real Time Systems | PI = (8/9) |
| Type B | Command & Control and Process Control Systems | PI = (10/11) |
| Type C | Telecommunications, Systems Software, and Scientific Systems | PI = (12/13) |
| Type D | Business and Miscellaneous Systems | PI = (14/15) |

**Table E-2. Trend Line Categories.**

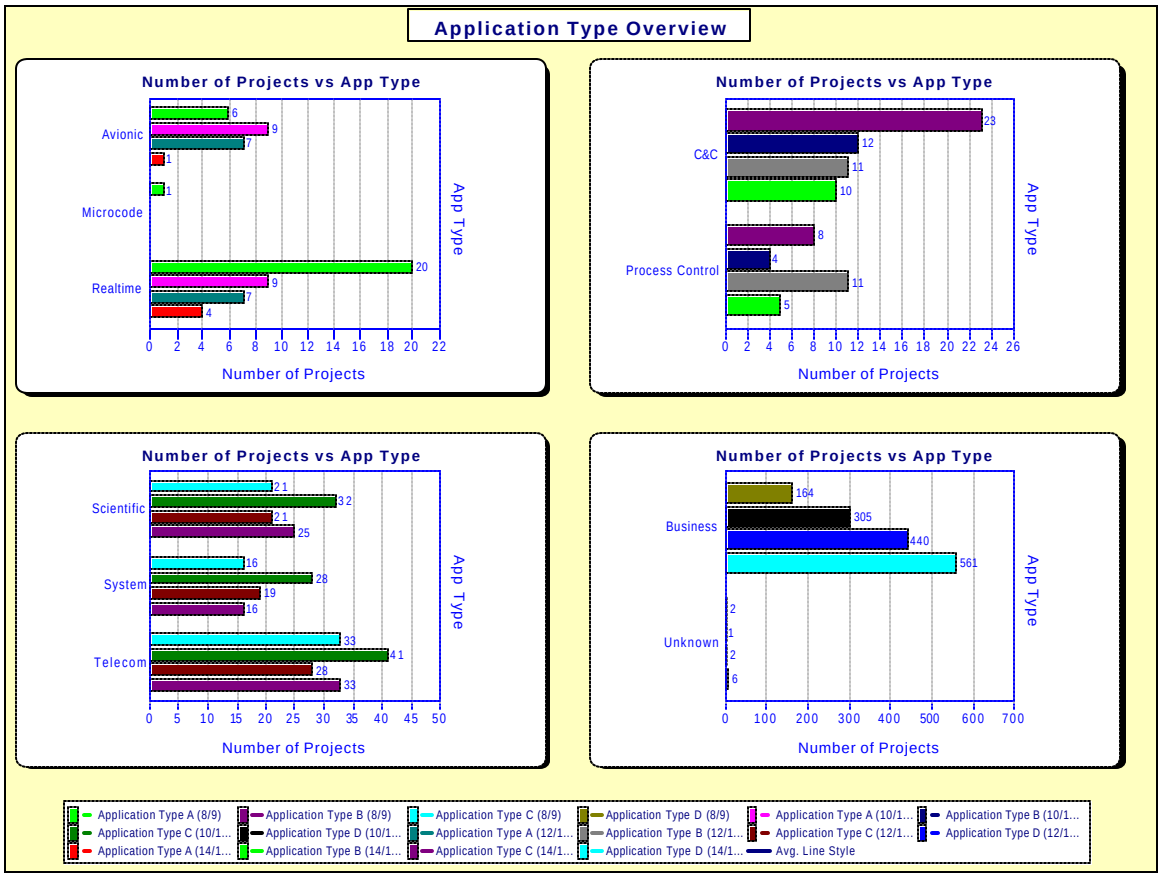



**Figure E-6. Number of Projects by Application Type.**

Read the illustration Figure E-6 in the standard convention left to right and top to bottom. Each quarter represents the four functional complexities (Type A, Type B, Type C, and Type D). Additionally, in each quarter, the identified application type contains *efficiency* information graphed from top to bottom (PI = (8/9), PI = (10/11), PI = (12/13), and PI = (14/15).

For example, for a functional complexity of Type C (bottom left), there are 41 *telecom* applications produced by an organization that has a *productivity index* of ten or eleven.

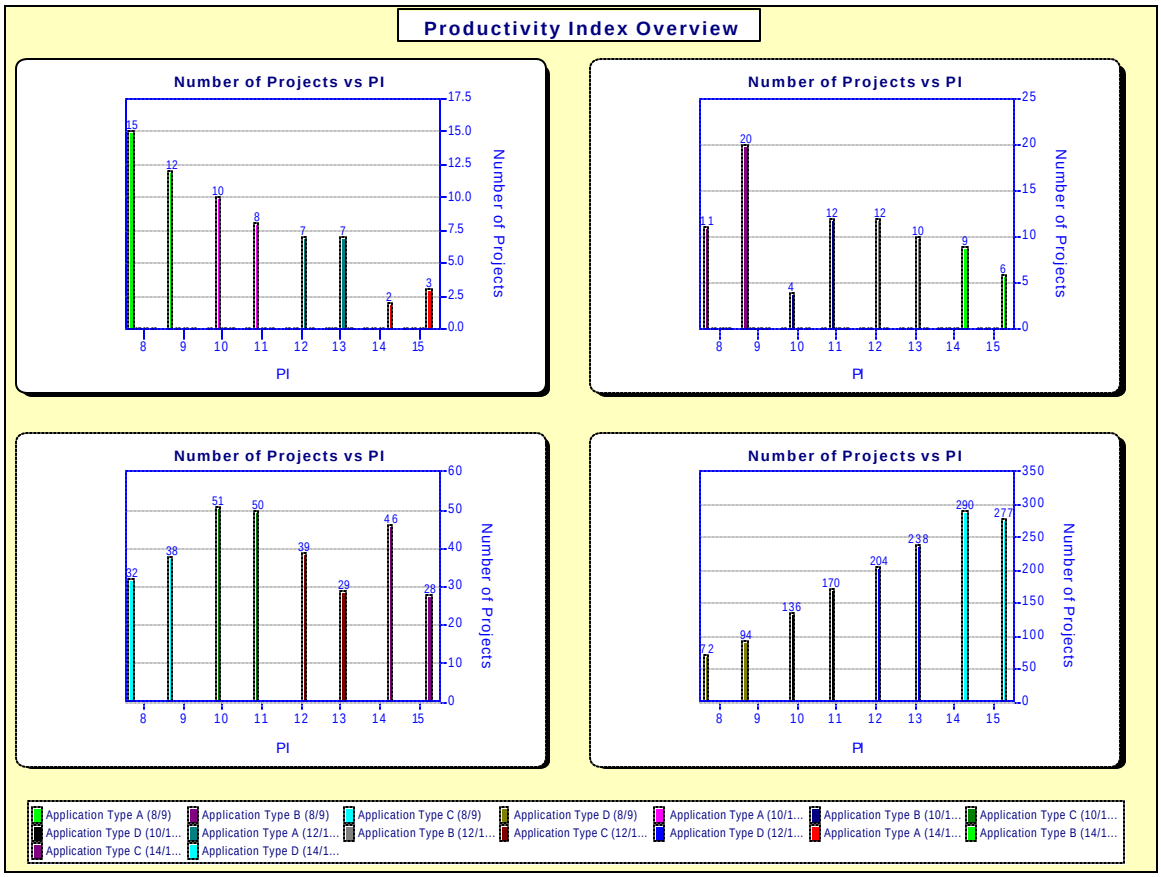



**Figure E-7. Number of Projects by Productivity Index.**

Following the standard conventions, Figure E-7 identifies how many projects in the subset as identified by their *productivity index*. For example, there are 170 projects (*functional complexity* D) developed by an organization with a *productivity index* of 11.

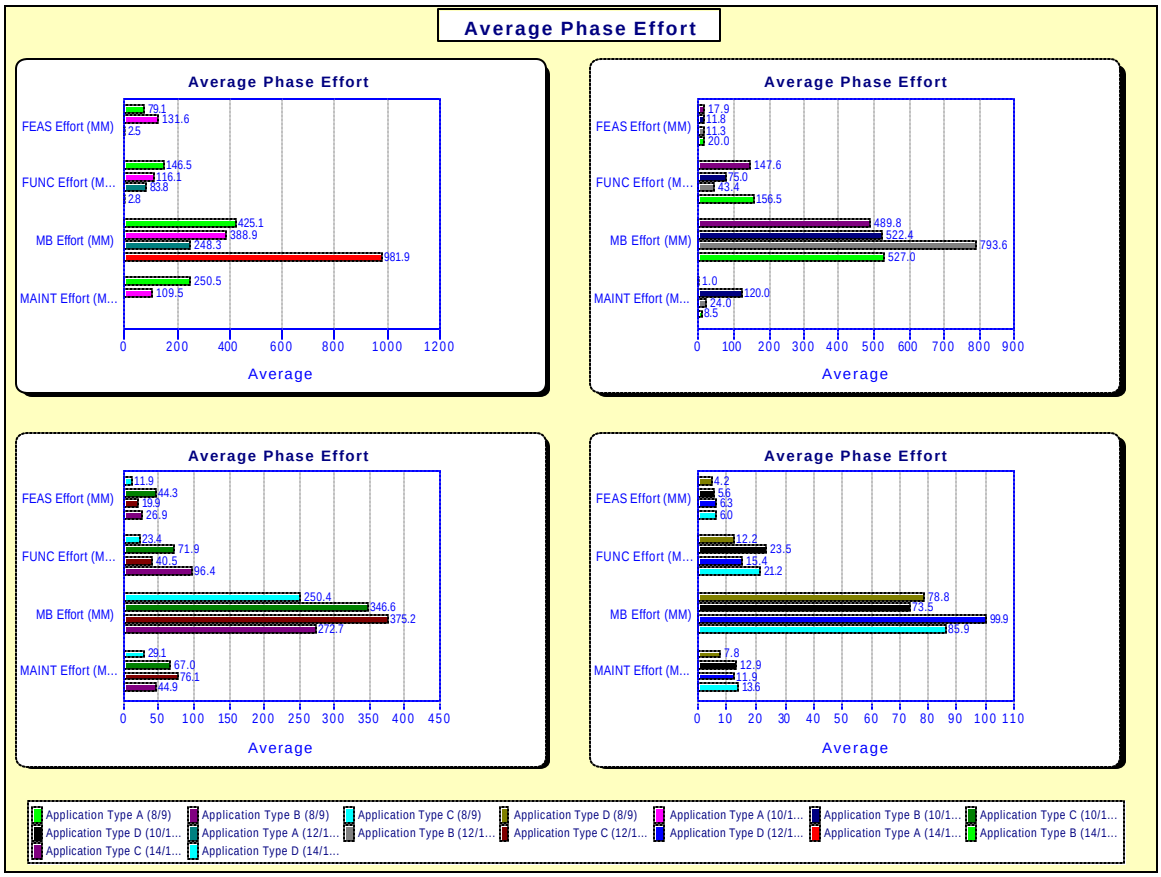



**Figure E-8. Average Phase Effort.**

Figure E-8 is the standard quad chart illustrating the average *effort* required for each phase. For example, in the bottom-left illustration (*functional complexity* C), an organization with a *productivity index* of eight or nine, took an average of about 250 man-months to complete the main build of the application.

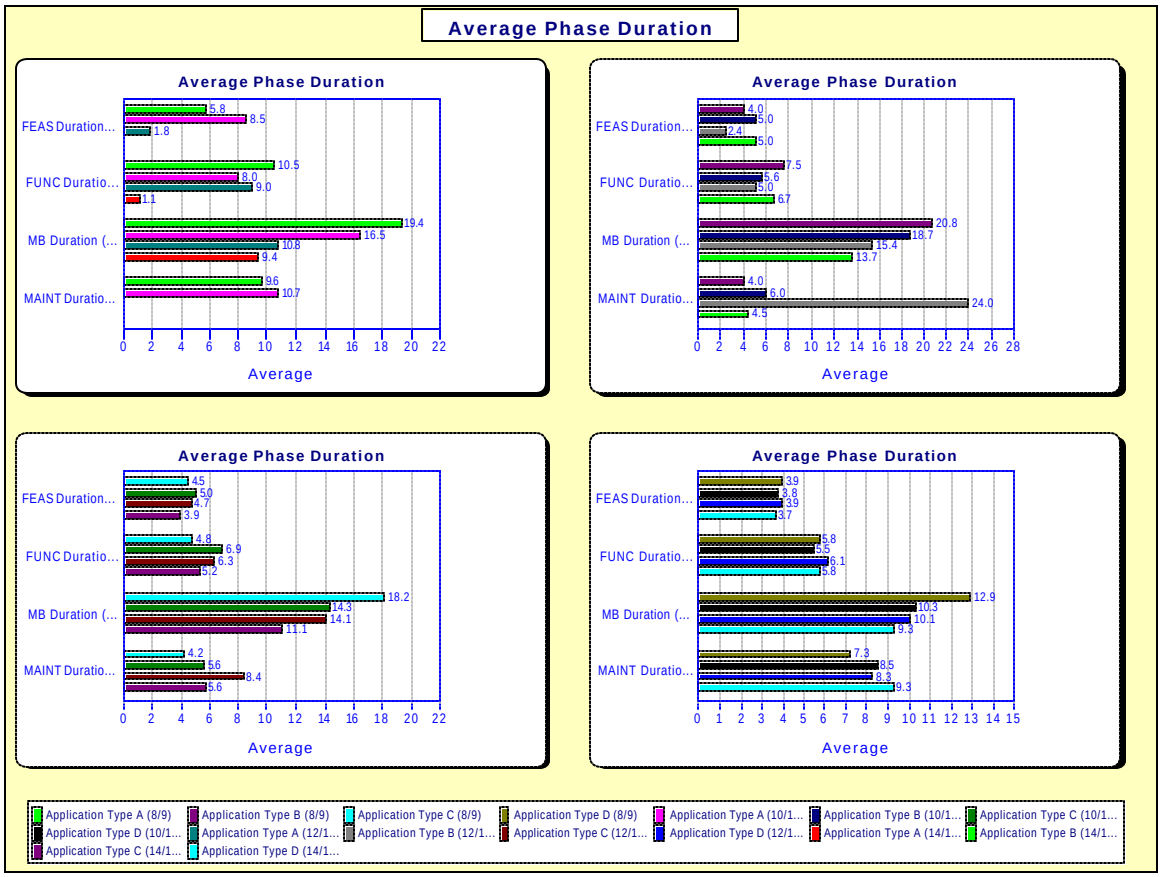



**Figure E-9. Average Phase Duration.**

Figure E-9 is a quad chart illustrating the average *duration* required for each phase. For example, in the bottom-left illustration (*functional complexity* C), an organization with a *productivity index* of eight or nine, took an average of 18.2 months to complete the *main build* of the application.

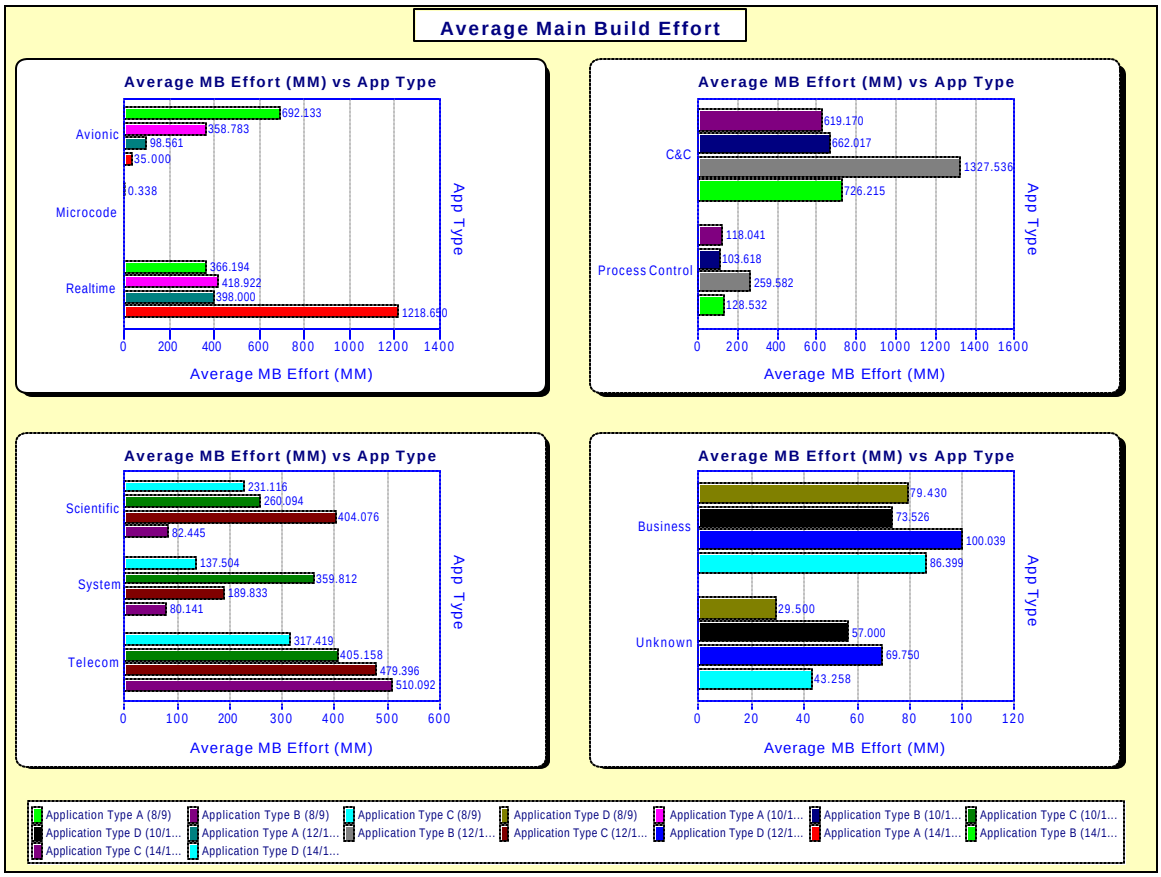



**Figure E-10. Main Build Effort by Application Type.**

Figure E-10 reviews the project subset just considering the *main build* phase. All ten of the application types are presented with their individual required *effort*, based on the recorded *efficiency*. A business system (Type D) required an organization with productivity of 12 or 13, an average of 100 man-months to complete the *main build*. Annex E-1 to this appendix contains a written description of the primary languages used for each of the application types.

## C. EFFICIENCY TRENDS

The next series of illustrations are the actual plots of the individual projects. Since the MRM is designed to project the required effort, all of the trend lines use the minimum effort as their dependent axis. The basic functional sizing unit (E-SLOC)

serves as the independent axis. Figure E-11 contains the *function complexities* and *efficiency* category as indicated.

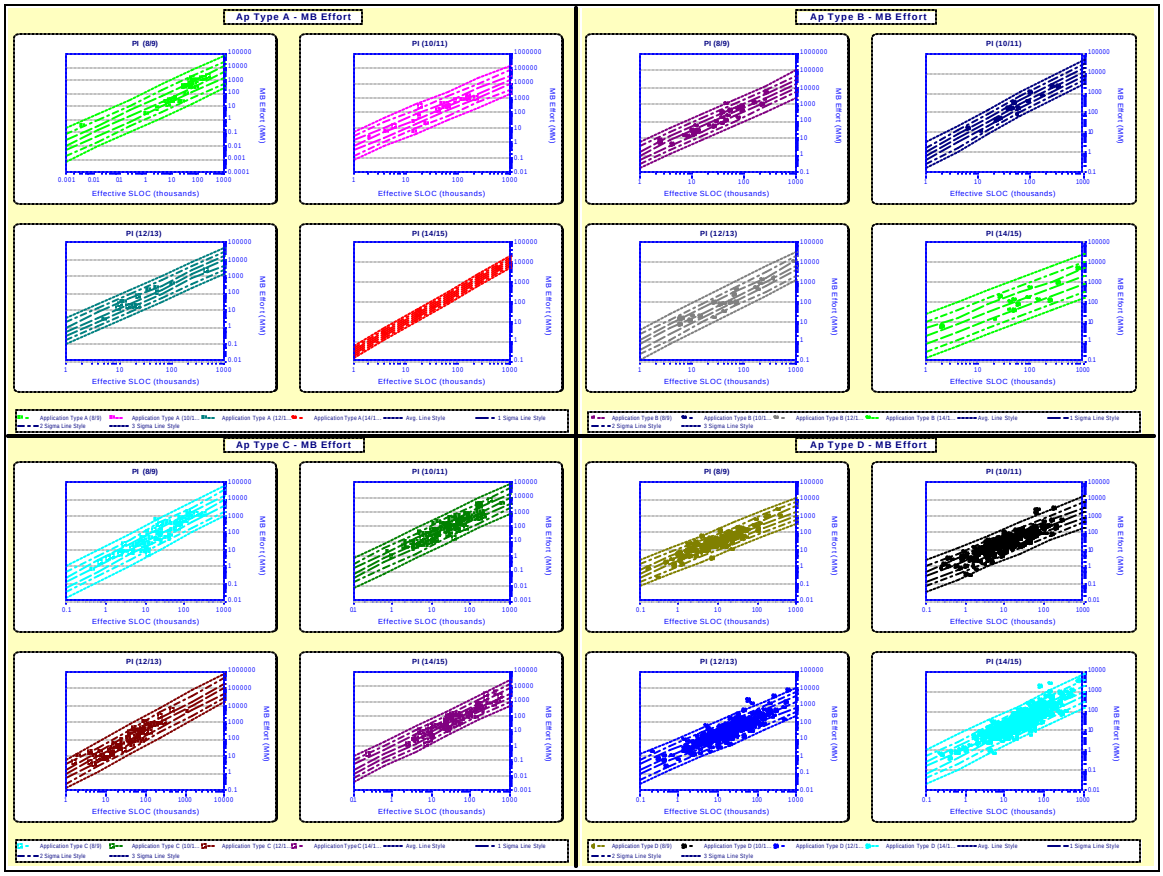



**Figure E-11. Effort Trends for Organizational Efficiency.**

## D. FUNCTIONAL COMPLEXITY TRENDS

Figure E-12 provides another view of the trend line data in Figure E-11. The trends are grouped differently to provide the reader another perspective on the data. Figure E-12 contains the *function complexities* and *efficiency* category as indicated.

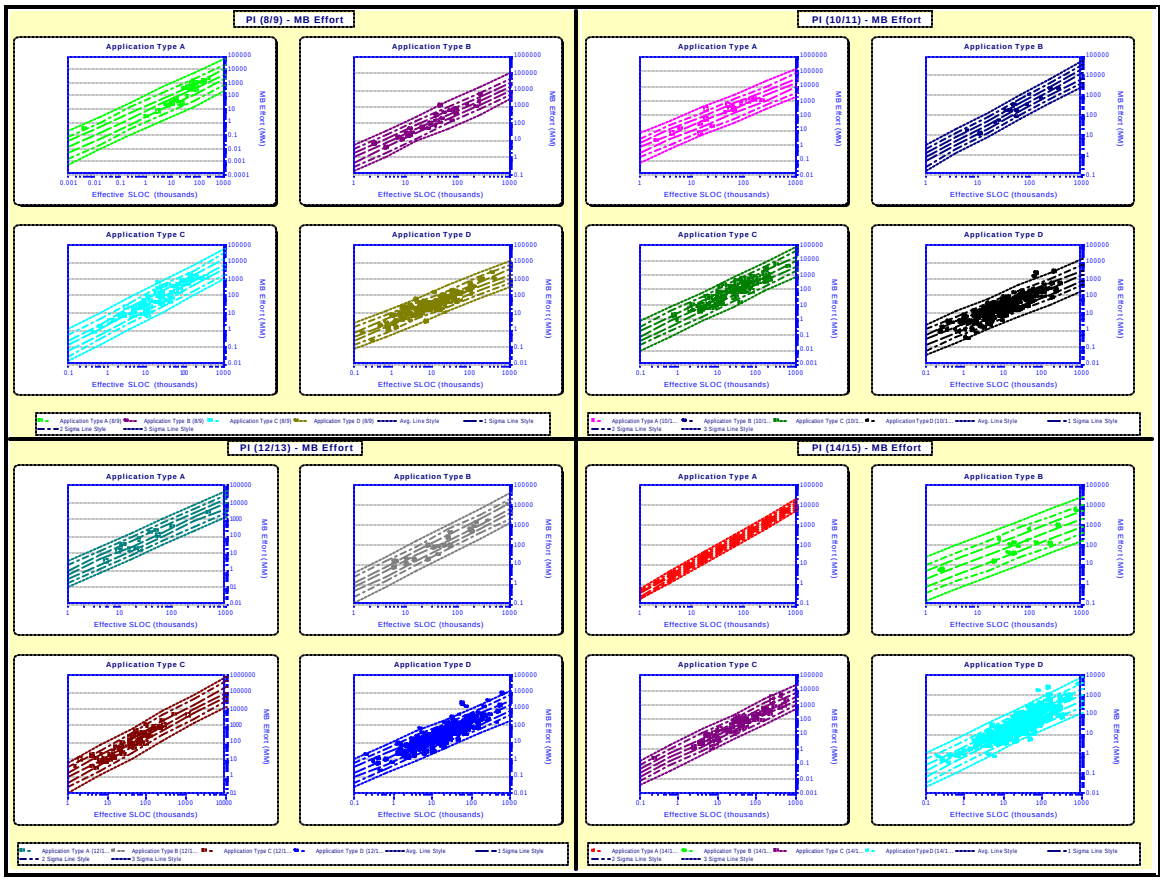



**Figure E-12. Effort Trends for Functional Complexity.**

## E. DISTRIBUTION ANALYSIS

Figure E-13 is a consolidated view of the sixteen projection lines used in the foundation of the MRM. An un-annotated version is found in Chapter VI. Table E-3, reproduced from Chapter VI, can serve as an ordered legend. The projection of all trend lines on a single scale demonstrates the importance of considering both the organization's *efficiency* and the *functional complexity* of the project.

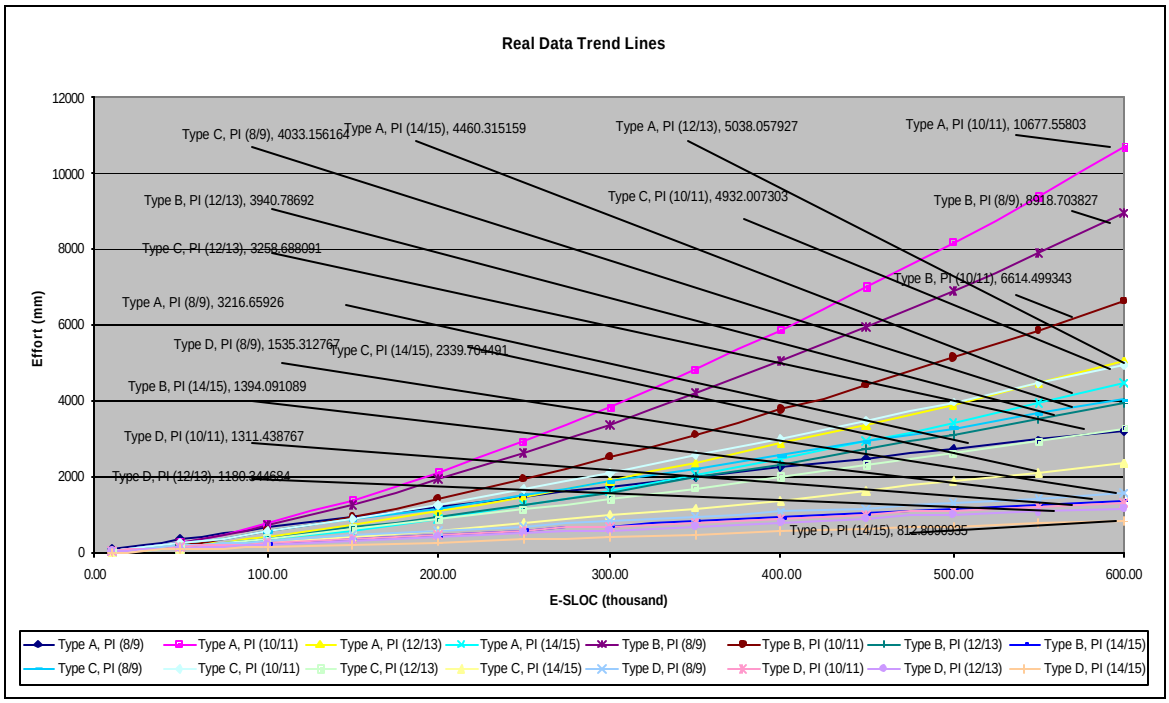



**Figure E-13. MRM Trend Lines.**

| Effort Required[76] | Trend Line Scenario | $R^2$ |
|---|---|---|
| 16 | Type A (10/11) | 0.8167 |
| 15 | Type B (8/9) | 0.8767 |
| 14 | Type B (10/11) | 0.8918 |
| 13 | Type A (12/13) | 0.9946 |
| 12 | Type C (10/11) | 0.9039 |
| 11 | Type A (14/15) | 0.9115 |
| 10 | Type C (8/9) | 0.9167 |
| 9 | Type B (12/13) | 0.7959 |
| 8 | Type C (12/13) | 0.8511 |
| 7 | Type A (8/9) | 0.8311 |
| 6 | Type C (14/15) | 0.8697 |
| 5 | Type D (8/9) | 0.8978 |
| 4 | Type B (14/15) | 0.8102 |
| 3 | Type D (10/11) | 0.7426 |
| 2 | Type D (12/13) | 0.7777 |
| 1 | Type D (14/15) | 0.7668 |

**Table E-3. Ordered Projection of Trend Line Data.**

[76] Effort Required ranges from high to low. The trend line Type A (10/11), required the most effort of all of the trends.

The final series of illustrations documents the origins of the *alpha, beta,* and *gamma*, derived for each of the sixteen foundation trend lines. There are a total of eight graphs; four sets of two. The illustrations are organized around the *functional complexity.* For example, the first two charts only consider the *functional complexity* of Type A (Microcode, Avionic, and Real Time) systems. The four plotted lines represent each of the efficiency categories for the indicated complexity.

The graph on the left provides the probability distribution function as given by the three-variable Weibull equation:

$$\text{pdf}: f(x; \gamma, \alpha, \beta) = \begin{cases} 0, & x < \gamma \\ (\alpha / \beta^{\alpha})(x-\gamma)^{\alpha-1} \exp(-((x-\gamma)/\beta)^{\alpha}), & x \geq \gamma \end{cases}$$

The graph on the right linearizes the three-variable Weibull cumulative distribution function:

$$\text{cdf}: F(x; \gamma, \alpha, \beta) = \begin{cases} 0, & x < \gamma \\ 1 - \exp(-((x-\gamma)/\beta)^{\alpha}), & x \geq \gamma \end{cases}$$

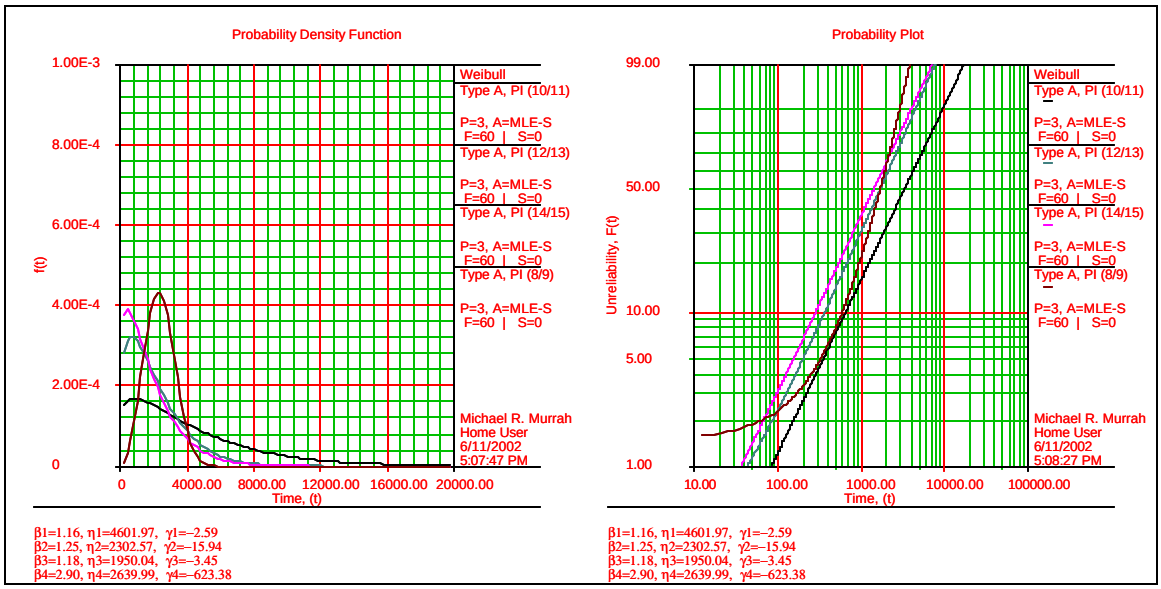



**Figure E-14. Function Complexity Type A Distribution.**

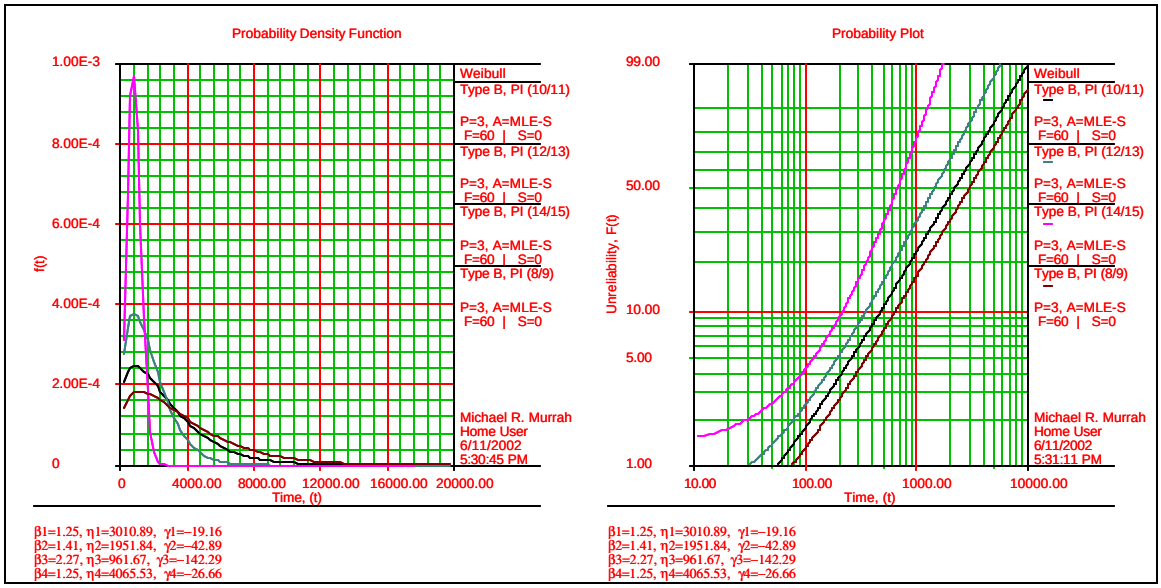



**Figure E-15. Function Complexity Type B Distribution.**

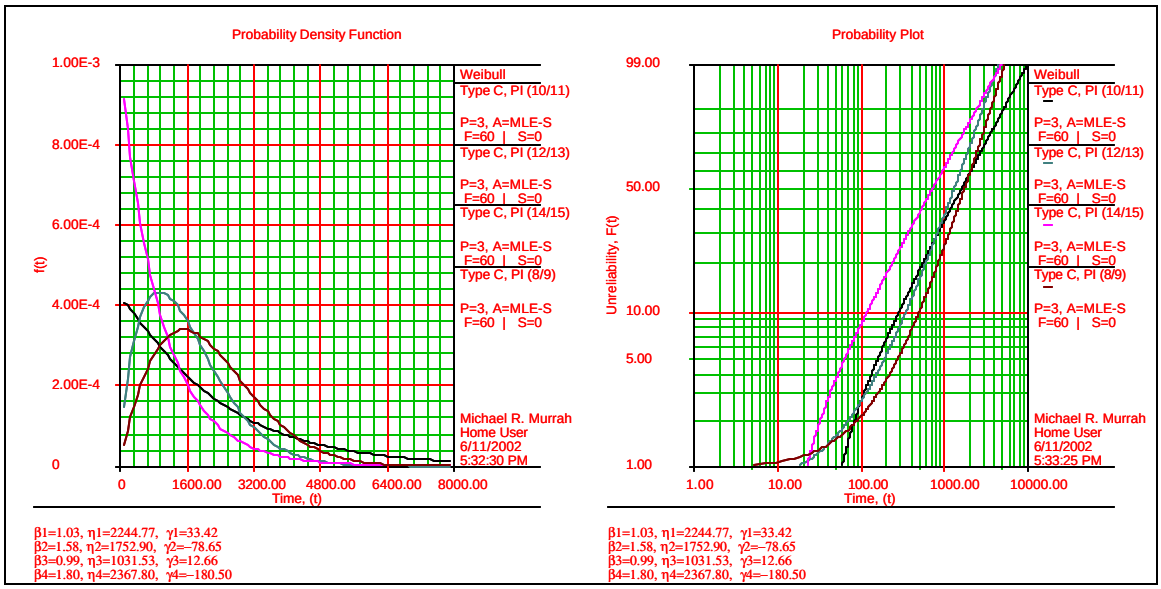



**Figure E-16. Function Complexity Type C Distribution.**

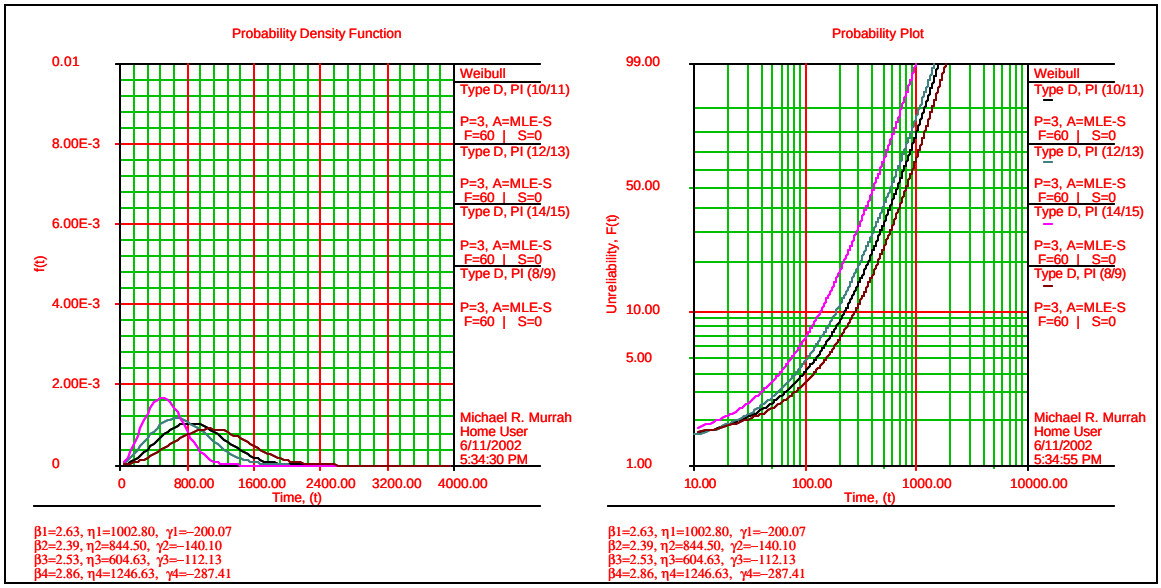



**Figure E-17. Function Complexity Type D Distribution.**

The nomenclature utilized by ReliaSoft's Weibull ++ 5.0 32 Bit (Pro) is different than the convention in this dissertation. The following table provides a quick translation for the nomenclature difference.

| Weibull ++ | Dissertation |
|---|---|
| ß | a |
| ? | ß |
| ? | ? |

**Table E-4. Nomenclature Conversion.**

THIS PAGE INTENTIONALLY LEFT BLANK

# ANNEX E-1. PRIMARY LANGUAGE BY APPLICATION TYPE

**Number of Projects vs Primary Lang**

**Data Set:** Application Type A (8/9)

| Primary Lang | Number of Projects |
|---|---|
| (Unknown) | 3 |
| ADA | 2 |
| ASSEMBLER | 7 |
| C | 5 |
| CMS-2 | 2 |
| CORAL | 1 |
| FORMS | - |
| FORTRAN | 2 |
| JOVIAL | 2 |
| KAREL | - |
| PASCAL | 2 |
| PLM | - |
| SDP PASCAL | 1 |
| SPL1 | - |

**Data Set:** Application Type A (10/11)

| Primary Lang | Number of Projects |
|---|---|
| (Unknown) | 5 |
| ADA | 7 |
| ASSEMBLER | - |
| C | 2 |
| CMS-2 | 1 |
| CORAL | - |
| FORMS | - |
| FORTRAN | 1 |
| JOVIAL | - |
| KAREL | - |
| PASCAL | - |
| PLM | 2 |
| SDP PASCAL | - |
| SPL1 | - |

**Data Set:** Application Type A (12/13)

| Primary Lang | Number of Projects |
|---|---|
| (Unknown) | 3 |
| ADA | 4 |
| ASSEMBLER | 1 |
| C | - |
| CMS-2 | 1 |
| CORAL | - |
| FORMS | - |
| FORTRAN | 1 |
| JOVIAL | 1 |
| KAREL | 1 |
| PASCAL | - |
| PLM | 2 |
| SDP PASCAL | - |
| SPL1 | - |

**Data Set:** Application Type A (14/15)

| Primary Lang | Number of Projects |
|---|---|
| (Unknown) | 2 |
| ADA | 1 |
| ASSEMBLER | - |

| | |
|---|---|
| C | - |
| CMS-2 | - |
| CORAL | - |
| FORMS | 1 |
| FORTRAN | - |
| JOVIAL | - |
| KAREL | - |
| PASCAL | - |
| PLM | - |
| SDP PASCAL | - |
| SPL1 | 1 |

**Number of Projects vs Primary Lang**

**Data Set:** Application Type B (8/9)

| Primary Lang | Number of Projects |
|---|---|
| (Unknown) | 9 |
| ADA | 6 |
| ASSEMBLER | 2 |
| C | 3 |
| C++ | - |
| CMS-2 | 5 |
| CORAL | 1 |
| FORTRAN | - |
| Higher Order Language | - |
| JOVIAL | 1 |
| PASCAL | 2 |
| PL/1 | 1 |
| VISUAL BASIC | 1 |

**Data Set:** Application Type B (10/11)

| Primary Lang | Number of Projects |
|---|---|
| (Unknown) | 6 |
| ADA | 3 |
| ASSEMBLER | 1 |
| C | 2 |
| C++ | - |
| CMS-2 | - |
| CORAL | - |
| FORTRAN | 2 |
| Higher Order Language | 1 |
| JOVIAL | - |
| PASCAL | 1 |
| PL/1 | - |
| VISUAL BASIC | - |

**Data Set:** Application Type B (12/13)

| Primary Lang | Number of Projects |
|---|---|
| (Unknown) | 7 |
| ADA | 7 |
| ASSEMBLER | 1 |
| C | 2 |
| C++ | 1 |
| CMS-2 | 1 |
| CORAL | 1 |
| FORTRAN | - |
| Higher Order Language | 1 |
| JOVIAL | - |
| PASCAL | 1 |
| PL/1 | - |
| VISUAL BASIC | - |

**Data Set:** Application Type B (14/15)

| Primary Lang | Number of Projects |
|---|---|
| (Unknown) | 4 |
| ADA | 1 |
| ASSEMBLER | 1 |

| | |
|---|---|
| C | 1 |
| C++ | 2 |
| CMS-2 | - |
| CORAL | - |
| FORTRAN | 2 |
| Higher Order Language | - |
| JOVIAL | 1 |
| PASCAL | 3 |
| PL/1 | - |
| VISUAL BASIC | - |

**Number of Projects vs Primary Lang**

**Data Set:** Application Type C (8/9)

| Primary Lang | Number of Projects |
|---|---|
| (Unknown) | 7 |
| ADA | 9 |
| ADS ONLINE | - |
| ALGOL | - |
| ASCII | - |
| ASSEMBLER | 6 |
| BASIC | 2 |
| C | 14 |
| C++ | 2 |
| CHILL | - |
| CMS-2 | 1 |
| COBOL | 5 |
| CORAL | 2 |
| CORAL 66 | 1 |
| ERSPL | - |
| FORTRAN | 11 |
| HTML | - |
| ISPF DIALOG | - |
| JAVA | - |
| JOVIAL | - |
| MATLAB | - |
| NATURAL | 1 |
| NEXPERT | - |
| Netexpert Events | - |
| PASCAL | 2 |
| PL/1 | 4 |
| PL/AS | 1 |
| PL/M | - |
| PL/S | - |
| PLM | - |
| PLUS | 1 |
| PLX | - |
| SDL | 1 |
| SL1 | - |
| SYSBUILDER | - |
| TAL | - |
| TSPL | - |
| UNIX SHELL | - |
| VARIOUS | - |

**Data Set:** Application Type C (10/11)

| Primary Lang | Number of Projects |
|---|---|
| (Unknown) | 26 |
| ADA | 5 |
| ADS ONLINE | 1 |
| ALGOL | 1 |
| ASCII | - |
| ASSEMBLER | 4 |
| BASIC | 4 |
| C | 18 |
| C++ | 6 |
| CHILL | 1 |
| CMS-2 | - |

| | |
|---|---|
| COBOL | 4 |
| CORAL | 1 |
| CORAL 66 | - |
| ERSPL | 1 |
| FORTRAN | 11 |
| HTML | - |
| ISPF DIALOG | - |
| JAVA | 1 |
| JOVIAL | - |
| MATLAB | - |
| NATURAL | - |
| NEXPERT | - |
| Netexpert Events | 1 |
| PASCAL | 2 |
| PL/1 | 3 |
| PL/AS | 1 |
| PL/M | 1 |
| PL/S | 1 |
| PLM | 2 |
| PLUS | - |
| PLX | 1 |
| SDL | - |
| SL1 | - |
| SYSBUILDER | - |
| TAL | 1 |
| TSPL | - |
| UNIX SHELL | 4 |
| VARIOUS | - |

**Data Set:** Application Type C (12/13)

| Primary Lang | Number of Projects |
|---|---|
| (Unknown) | 9 |
| ADA | 4 |
| ADS ONLINE | - |
| ALGOL | - |
| ASCII | 1 |
| ASSEMBLER | 4 |
| BASIC | 2 |
| C | 22 |
| C++ | 2 |
| CHILL | 1 |
| CMS-2 | - |
| COBOL | 5 |
| CORAL | 2 |
| CORAL 66 | - |
| ERSPL | - |
| FORTRAN | 4 |
| HTML | - |
| ISPF DIALOG | - |
| JAVA | - |
| JOVIAL | - |
| MATLAB | - |
| NATURAL | - |
| NEXPERT | - |
| Net expert Events | - |
| PASCAL | 3 |
| PL/1 | 3 |
| PL/AS | - |
| PL/M | 1 |
| PL/S | - |
| PLM | 1 |
| PLUS | 1 |
| PLX | - |
| SDL | - |
| SL1 | 1 |
| SYSBUILDER | 1 |
| TAL | - |
| TSPL | 1 |

| | |
|---|---|
| UNIX SHELL | - |
| VARIOUS | - |

**Data Set:** Application Type C (14/15)

| Primary Lang | Number of Projects |
|---|---|
| (Unknown) | 22 |
| ADA | 6 |
| ADS ONLINE | - |
| ALGOL | - |
| ASCII | - |
| ASSEMBLER | - |
| BASIC | - |
| C | 18 |
| C++ | 6 |
| CHILL | - |
| CMS-2 | - |
| COBOL | 4 |
| CORAL | 1 |
| CORAL 66 | - |
| ERSPL | - |
| FORTRAN | 5 |
| HTML | 1 |
| ISPF DIALOG | 1 |
| JAVA | - |
| JOVIAL | 1 |
| MATLAB | 1 |
| NATURAL | - |
| NEXPERT | 1 |
| Netexpert Events | - |
| PASCAL | 2 |
| PL/1 | 4 |
| PL/AS | - |
| PL/M | - |
| PL/S | - |
| PLM | - |
| PLUS | - |
| PLX | - |
| SDL | - |
| SL1 | - |
| SYSBUILDER | - |
| TAL | - |
| TSPL | - |
| UNIX SHELL | - |
| VARIOUS | 1 |

**Number of Projects vs Primary Lang**

**Data Set:** Application Type D (8/9)

| Primary Lang | Number of Projects |
|---|---|
| (Unknown) | 5 |
| ABAP | 1 |
| ACCESS | - |
| ADA | - |
| ADABAS | - |
| ADL | - |
| ADS ONLINE | 2 |
| AFOLDS | - |
| AM | - |
| AM NOT EST | - |
| APL | - |
| APS | 1 |
| AREV | - |
| ARTEMIS | - |
| ASSEMBLER | 1 |
| Application Master | - |
| BAL | - |
| BASIC | - |

| | |
|---|---|
| BUILDER | - |
| C | 4 |
| C++ | 1 |
| C/C++ | - |
| CBAS | - |
| CCP | 1 |
| CLIPPER | - |
| COBOL | 82 |
| COBOL(STR) | - |
| COBOL/ADS | 1 |
| COBOL/SQL | - |
| COLUMBUS | - |
| CONSTRUCT | - |
| CORAL | - |
| CREDIT | - |
| CRYSTAL | - |
| CSL | - |
| CSP | - |
| CULPRIT | - |
| Cold Fusion | - |
| DATABASIC | - |
| DATABUS | - |
| DATMAN | - |
| DBASE | - |
| DELPHI | - |
| DELTA | - |
| DIBOL | 1 |
| DSQL(TERA) | - |
| Dataflex | - |
| EAL | - |
| EASYTRIEVE | 2 |
| EXCEL | - |
| Essebase | - |
| FCL | - |
| FCS | - |
| FILEAID | - |
| FOCUS | 5 |
| FORMS | - |
| FORTRAN | - |
| FOXPRO | - |
| GEM | - |
| GENER/OL | 1 |
| HTML | - |
| IDEAL | - |
| IEF/COOL:GEN | - |
| INFO BASIC | - |
| INFORMIX | - |
| INVEX | - |
| IPDT | 1 |
| ISPF | - |
| ITX COBOL | - |
| JAVA | 1 |
| JCL/SQL | - |
| Java Script | 2 |
| JCL | 1 |
| LINC | - |
| LISP | 1 |
| LOTUS NOTE | - |
| MACRO | - |
| MACRO II | - |
| MAGNA | - |
| MANTIS | - |
| MAPPER | 1 |
| MICROFOCUS | - |
| NATURAL | 3 |
| NOMAD | 2 |
| ORACLE | 1 |
| ORACLE 2K | - |

| | |
|---|---|
| ORACLE DEV 2000 | - |
| ORACLE SQL | - |
| OTHER | - |
| OpenROAD | - |
| Oracle Forms | - |
| Oracle HR | - |
| PACBASE | 2 |
| PARADOX | - |
| PASCAL | 2 |
| PEOPLESOFT | - |
| PERL | 1 |
| PL/1 | 22 |
| PL/SQL | 5 |
| POWERBUILDER | 1 |
| POWERHOUSE | - |
| PRO IV | 1 |
| PROGRESS | - |
| QUICKBUILD | - |
| REXX | - |
| RPG | 1 |
| SAL | - |
| SAS | - |
| SCC | 1 |
| SCL | - |
| SCOBOL | 1 |
| SCREEN WRITER | - |
| SCRIPT | - |
| SIR | 1 |
| SL1 | - |
| SLOGAN | - |
| SMALLTALK | - |
| SMARTSTAR | - |
| SQL | 1 |
| SQL FORMS | - |
| SQL PLUS | - |
| SQL REPORT | - |
| SQL WINDOW | - |
| SQL/QMF | - |
| SUPERCALC | 1 |
| Source File | - |
| TELON | 1 |
| TIG | - |
| TRANSACT | - |
| Taskmate | - |
| Tuxedo | - |
| UFO | - |
| UNICODE | - |
| UNIF/PRONT | - |
| UNIFACE | - |
| UNIX | - |
| UNIX SHELL | - |
| USERCODE | - |
| Uniface 6.1 | 1 |
| VBScript | - |
| VISUAL BASIC | 3 |
| VM/COBOL | - |
| VMS | - |
| WEB Scripts | - |
| WIZARD | - |
| XGEN | - |

**Data Set:** Application Type D (10/11)

| Primary Lang | Number of Projects |
|---|---|
| (Unknown) | 10 |
| ABAP | 2 |
| ACCESS | - |
| ADA | - |
| ADABAS | - |

| | |
|---|---|
| ADL | - |
| ADS ONLINE | 4 |
| AFOLDS | - |
| AM | - |
| AM NOT EST | - |
| APL | - |
| APS | 3 |
| AREV | 1 |
| ARTEMIS | 1 |
| ASSEMBLER | 6 |
| Application Master | 3 |
| BAL | - |
| BASIC | 1 |
| BUILDER | - |
| C | 9 |
| C++ | 4 |
| C/C++ | - |
| CBAS | - |
| CCP | - |
| CLIPPER | 1 |
| COBOL | 138 |
| COBOL(STR) | - |
| COBOL/ADS | - |
| COBOL/SQL | 1 |
| COLUMBUS | - |
| CONSTRUCT | - |
| CORAL | - |
| CREDIT | - |
| CRYSTAL | - |
| CSL | - |
| CSP | 2 |
| CULPRIT | - |
| Cold Fusion | - |
| DATABASIC | - |
| DATABUS | 1 |
| DATMAN | - |
| DBASE | 2 |
| DELPHI | - |
| DELTA | 1 |
| DIBOL | - |
| DSQL(TERA) | - |
| Dataflex | 2 |
| EAL | - |
| EASYTRIEVE | 2 |
| EXCEL | 1 |
| Essebase | 1 |
| FCL | - |
| FCS | - |
| FILEAID | - |
| FOCUS | 11 |
| FORMS | - |
| FORTRAN | - |
| FOXPRO | - |
| GEM | - |
| GENER/OL | 5 |
| HTML | 4 |
| IDEAL | 3 |
| IEF/COOL:GEN | 2 |
| INFO BASIC | - |
| INFORMIX | 1 |
| INVEX | - |
| IPDT | - |
| ISPF | - |
| ITX COBOL | - |
| JAVA | 1 |
| JCL/SQL | - |
| Java Script | - |
| JCL | 1 |

| | |
|---|---|
| LINC | 3 |
| LISP | - |
| LOTUS NOTE | 1 |
| MACRO | 1 |
| MACRO II | - |
| MAGNA | - |
| MANTIS | 2 |
| MAPPER | - |
| MICROFOCUS | - |
| NATURAL | 6 |
| NOMAD | 2 |
| ORACLE | 2 |
| ORACLE 2K | - |
| ORACLE DEV 2000 | - |
| ORACLE SQL | 3 |
| OTHER | 1 |
| OpenROAD | - |
| Oracle Forms | 1 |
| Oracle HR | - |
| PACBASE | 1 |
| PARADOX | - |
| PASCAL | 2 |
| PEOPLESOFT | 1 |
| PERL | - |
| PL/1 | 23 |
| PL/SQL | - |
| POWERBUILDER | 1 |
| POWERHOUSE | - |
| PRO IV | 1 |
| PROGRESS | 1 |
| QUICKBUILD | - |
| REXX | - |
| RPG | 2 |
| SAL | - |
| SAS | 1 |
| SCC | - |
| SCL | 1 |
| SCOBOL | 1 |
| SCREEN WRITER | 1 |
| SCRIPT | - |
| SIR | - |
| SL1 | - |
| SLOGAN | - |
| SMALLTALK | - |
| SMARTSTAR | - |
| SQL | 7 |
| SQL FORMS | 1 |
| SQL PLUS | - |
| SQL REPORT | - |
| SQL WINDOW | - |
| SQL/QMF | - |
| SUPERCALC | - |
| Source File | 1 |
| TELON | 7 |
| TIG | - |
| TRANSACT | - |
| Taskmate | 1 |
| Tuxedo | - |
| UFO | - |
| UNICODE | - |
| UNIF/PRONT | 1 |
| UNIFACE | 1 |
| UNIX | - |
| UNIX SHELL | 1 |
| USERCODE | - |
| Uniface 6.1 | - |
| VBScript | - |
| VISUAL BASIC | 3 |

| | |
|---|---|
| VM/COBOL | - |
| VMS | - |
| WEB Scripts | - |
| WIZARD | - |
| XGEN | 1 |

**Data Set:** Application Type D (12/13)

| Primary Lang | Number of Projects |
|---|---|
| (Unknown) | 29 |
| ABAP | 2 |
| ACCESS | 3 |
| ADA | 1 |
| ADABAS | - |
| ADL | - |
| ADS ONLINE | 8 |
| AFOLDS | - |
| AM | 1 |
| AM NOT EST | - |
| APL | 1 |
| APS | 1 |
| AREV | 1 |
| ARTEMIS | - |
| ASSEMBLER | 7 |
| Application Master | 3 |
| BAL | - |
| BASIC | 4 |
| BUILDER | - |
| C | 4 |
| C++ | 9 |
| C/C++ | 1 |
| CBAS | - |
| CCP | - |
| CLIPPER | 2 |
| COBOL | 181 |
| COBOL(STR) | - |
| COBOL/ADS | 1 |
| COBOL/SQL | - |
| COLUMBUS | 1 |
| CONSTRUCT | 2 |
| CORAL | - |
| CREDIT | - |
| CRYSTAL | - |
| CSL | - |
| CSP | 2 |
| CULPRIT | 1 |
| Cold Fusion | - |
| DATABASIC | 1 |
| DATABUS | - |
| DATMAN | 1 |
| DBASE | 1 |
| DELPHI | - |
| DELTA | 3 |
| DIBOL | - |
| DSQL(TERA) | 1 |
| Dataflex | - |
| EAL | 1 |
| EASYTRIEVE | 1 |
| EXCEL | 1 |
| Essebase | - |
| FCL | - |
| FCS | 1 |
| FILEAID | 1 |
| FOCUS | 10 |
| FORMS | - |
| FORTRAN | 1 |
| FOXPRO | - |
| GEM | 1 |
| GENER/OL | 1 |

| | |
|---|---|
| HTML | 3 |
| IDEAL | 6 |
| IEF/COOL:GEN | - |
| INFO BASIC | 1 |
| INFORMIX | - |
| INVEX | - |
| IPDT | - |
| ISPF | 1 |
| ITX COBOL | - |
| JAVA | 2 |
| JCL/SQL | - |
| Java Script | - |
| JCL | - |
| LINC | 2 |
| LISP | - |
| LOTUS NOTE | - |
| MACRO | 1 |
| MACRO II | - |
| MAGNA | 1 |
| MANTIS | - |
| MAPPER | 1 |
| MICROFOCUS | 1 |
| NATURAL | 13 |
| NOMAD | - |
| ORACLE | 5 |
| ORACLE 2K | 1 |
| ORACLE DEV 2000 | 2 |
| ORACLE SQL | - |
| OTHER | - |
| OpenROAD | - |
| Oracle Forms | 1 |
| Oracle HR | - |
| PACBASE | 2 |
| PARADOX | 1 |
| PASCAL | 1 |
| PEOPLESOFT | - |
| PERL | - |
| PL/1 | 43 |
| PL/SQL | 3 |
| POWERBUILDER | 3 |
| POWERHOUSE | 1 |
| PRO IV | 3 |
| PROGRESS | 1 |
| QUICKBUILD | - |
| REXX | - |
| RPG | 4 |
| SAL | 2 |
| SAS | 1 |
| SCC | - |
| SCL | - |
| SCOBOL | 1 |
| SCREEN WRITER | - |
| SCRIPT | 1 |
| SIR | 2 |
| SL1 | 1 |
| SLOGAN | 2 |
| SMALLTALK | - |
| SMARTSTAR | - |
| SQL | 12 |
| SQL FORMS | 8 |
| SQL PLUS | - |
| SQL REPORT | 1 |
| SQL WINDOW | 1 |
| SQL/QMF | - |
| SUPERCALC | - |
| Source File | - |
| TELON | 6 |
| TIG | - |

| | |
|---|---|
| TRANSACT | - |
| Taskmate | 1 |
| Tuxedo | 1 |
| UFO | 1 |
| UNICODE | - |
| UNIF/PRONT | - |
| UNIFACE | 1 |
| UNIX | - |
| UNIX SHELL | 1 |
| USERCODE | 1 |
| Uniface 6.1 | - |
| VBScript | - |
| VISUAL BASIC | 5 |
| VM/COBOL | - |
| VMS | - |
| WEB Scripts | - |
| WIZARD | - |
| XGEN | - |

**Data Set:** Application Type D (14/15)

| Primary Lang | Number of Projects |
|---|---|
| (Unknown) | 39 |
| ABAP | - |
| ACCESS | 3 |
| ADA | 2 |
| ADABAS | 1 |
| ADL | 1 |
| ADS ONLINE | 10 |
| AFOLDS | 1 |
| AM | 2 |
| AM NOT EST | 1 |
| APL | - |
| APS | 1 |
| AREV | - |
| ARTEMIS | 1 |
| ASSEMBLER | 8 |
| Application Master | 9 |
| BAL | 1 |
| BASIC | 6 |
| BUILDER | 1 |
| C | 11 |
| C++ | 5 |
| C/C++ | - |
| CBAS | 1 |
| CCP | - |
| CLIPPER | 3 |
| COBOL | 232 |
| COBOL(STR) | 1 |
| COBOL/ADS | - |
| COBOL/SQL | - |
| COLUMBUS | 1 |
| CONSTRUCT | 2 |
| CORAL | 1 |
| CREDIT | 1 |
| CRYSTAL | 1 |
| CSL | 1 |
| CSP | 1 |
| CULPRIT | - |
| Cold Fusion | 1 |
| DATABASIC | - |
| DATABUS | - |
| DATMAN | - |
| DBASE | - |
| DELPHI | 2 |
| DELTA | 2 |
| DIBOL | - |
| DSQL(TERA) | - |
| Dataflex | - |

| | |
|---|---|
| EAL | - |
| EASYTRIEVE | 1 |
| EXCEL | 1 |
| Essebase | - |
| FCL | 1 |
| FCS | - |
| FILEAID | - |
| FOCUS | 7 |
| FORMS | 2 |
| FORTRAN | 1 |
| FOXPRO | 1 |
| GEM | 1 |
| GENER/OL | 4 |
| HTML | 3 |
| IDEAL | 11 |
| IEF/COOL:GEN | 7 |
| INFO BASIC | - |
| INFORMIX | 3 |
| INVEX | 1 |
| IPDT | - |
| ISPF | - |
| ITX COBOL | 1 |
| JAVA | 2 |
| JCL/SQL | 1 |
| Java Script | 1 |
| JCL | - |
| LINC | 1 |
| LISP | - |
| LOTUS NOTE | - |
| MACRO | - |
| MACRO II | 1 |
| MAGNA | - |
| MANTIS | 3 |
| MAPPER | 1 |
| MICROFOCUS | - |
| NATURAL | 19 |
| NOMAD | 1 |
| ORACLE | 2 |
| ORACLE 2K | 3 |
| ORACLE DEV 2000 | 5 |
| ORACLE SQL | - |
| OTHER | - |
| OpenROAD | 1 |
| Oracle Forms | - |
| Oracle HR | 1 |
| PACBASE | 3 |
| PARADOX | - |
| PASCAL | 3 |
| PEOPLESOFT | 1 |
| PERL | - |
| PL/1 | 33 |
| PL/SQL | 4 |
| POWERBUILDER | 5 |
| POWERHOUSE | - |
| PRO IV | - |
| PROGRESS | - |
| QUICKBUILD | 1 |
| REXX | 1 |
| RPG | 7 |
| SAL | - |
| SAS | - |
| SCC | - |
| SCL | - |
| SCOBOL | 1 |
| SCREEN WRITER | 3 |
| SCRIPT | - |
| SIR | 1 |
| SL1 | - |

| | |
|---|---|
| SLOGAN | 2 |
| SMALLTALK | 5 |
| SMARTSTAR | 1 |
| SQL | 8 |
| SQL FORMS | 9 |
| SQL PLUS | 2 |
| SQL REPORT | - |
| SQL WINDOW | 1 |
| SQL/QMF | 1 |
| SUPERCALC | - |
| Source File | - |
| TELON | 8 |
| TIG | 1 |
| TRANSACT | 1 |
| Taskmate | - |
| Tuxedo | 2 |
| UFO | 1 |
| UNICODE | 1 |
| UNIF/PRONT | - |
| UNIFACE | - |
| UNIX | 1 |
| UNIX SHELL | - |
| USERCODE | - |
| Uniface 6.1 | 1 |
| VBScript | 2 |
| VISUAL BASIC | 11 |
| VM/COBOL | 1 |
| VMS | 1 |
| WEB Scripts | 2 |
| WIZARD | 2 |
| XGEN | - |

# F. MRM VALIDATION

This appendix provides direct support to Chapter VII. It is divided into two primary sections and contains data relevant to the validation of the MRM. The first section details the methodology implemented to ensure that the three test models are projecting information based on the same assumptions. The last section of the appendix details the atomic level of validation of the MRM.

## A. EQUALIZING THE MODELS

Chapter IV introduces the fact that the Basic COCOMO and Simplified Software Equation do not represent the same phases of the software lifecycle. The Basic COCOMO equation considers the time / effort required to produce the *specifications* and the *main build* (Boeh83); the Simplified Software Equation considers only the *main build* (Putn92). The MRM also only considers the *main build*. In order to properly represent the performance of each of these models, a suitable technique had to be devised to normalize the data to a common denominator. The following is an extract of a conversation with Lawrence Putnam, Sr. regarding equalizing the models.

> Murrah: Boehm's Tdev goes to the FOC but starts after the Requirements Analysis. It seems your model's (Software Equation) Td begins after the Functional Design Specifications and Boehm's before the Functional Design Specifications. Does that mean (Boehm's Tdev) = (Putnam's Tdmin + (Tdmin/3))?
>
> Putnam: Yes, approximately. [probably as good as you can do. I would do it that way.]

The previous equation only accounts for the differences in the development time, which worked fine for the SRM validation in Chapter IV and Appendix C. However, for the MRM validation, a method is needed to account for the differences in required effort. (Putn92) provides a partial technique to normalize the models. Table F-1 is an extension from (Putn92) and provides the suitable values for Equation (F.1).

$$E_{\text{func}} = \text{Fr}(E) \text{ man-months} \quad \text{(F.1)}$$

where,

$E_{\text{func}}$ = effort in the functional design phase
Fr = is the appropriate fraction from Table F-1
E = effort of the *main build*

| Size (K-SLOC) | Fraction | Factor |
|---|---|---|
| 0 | err | 0.000000 |
| 100 <= 15K | 0.60 | 0.625000 |
| 15K <= 20K | 0.50 | 0.666667 |
| 20K <= 25K | 0.35 | 0.740741 |
| 25K <= 30K | 0.28 | 0.781250 |
| 30K <= 40K | 0.20 | 0.833333 |
| 40K <= 50K | 0.16 | 0.862069 |
| 50K <= 70K | 0.18 | 0.847458 |
| 70K and avove | 0.20 | 0.833333 |

**Table F-1. Functional Design Conversion Factor.**

The column "Factor" in Table F-1 is an extension developed from this research and it is used to calculate the appropriate conversion. This conversion is based on the assumption in Equation (F.2).

$$\text{COCOMO}_{effort} = \text{SLIM}(Effort_{mainbuild} + Effort_{func}) \quad \text{(F.2)}$$

where,

| | |
|---|---|
| COCOMO[77] effort = | $\text{PM} = 2.4(\text{KDSI})^{1.05}$<br>$\text{PM} = 3.0(\text{KDSI})^{1.12}$<br>$\text{PM} = 3.6(\text{KDSI})^{1.20}$ |
| SSE Effort[78] mainbuild = | $E = 180\ B\ t_d^3$ |
| SSE Effort func = | $E_{\text{func}} = \text{Fr}(E)$ man-months |

77 The appropriate equation is applied: organic, semi-detached, or embedded.

78 The $B$ value is the special skills factor in (Putn92). The $t_d$ value is the minimum time, demonstrated in Appendix C.

The following example converts *effort*, derived using Basic COCOMO, to the equivalent *effort* of the SSE and MRM projections. Basic COCOMO calculates 1500 person-months are required for a 50K E-SLOC job. Look up the 50K in Table F-1 and multiply the "Factor" column by the COCOMO effort, as in Equation (F.3).

$$\text{Adjusted COCOMO} = \text{COCOMO(PM)} * \text{Factor} \qquad \text{(F.3)}$$

where,

Adjusted COCOMO = COCOMO's projection for just the main build
COCOMO(PM) = the original COCOMO effort projection
Factor = value obtained from Table F-1

The value obtained from the table is 0.862069. Multiplied by the original COCOMO effort of 1500, yields an adjusted COCOMO effort of 1293.1035 person-months (Main Build Only). The adjusted COCOMO effort is equivalent to the effort derived from the SSE and MRM models. This process was repeated for every value projected by the COCOMO equations.

## B. ATOMIC LEVEL OF VALIDATION

This last section of the appendix demonstrates the performance of each of the test models against eight different application types. These applications are the same projects presented in validation levels one and two during Chapter VII. As noted in Chapter VII, the atomic level of detail provides the "true" interpretation of each model's performance. Table F-2 provides comparison statistics pertaining to each model's overall performance and Table F-3 provides the percent of overall accuracy.

| | Rated First | Rated Second | Rated Third |
|---|---|---|---|
| **Real Time** | MRM | COCOMO | SSE |
| **Avionic** | MRM | COCOMO | SSE |
| **Command & Control** | COCOMO | MRM | SSE |
| **Process Control** | COCOMO | MRM | SSE |
| **Telecommunication** | MRM | SSE | COCOMO |
| **Systems Software** | MRM | COCOMO | SSE |
| **Scientific Software** | SSE | MRM | COCOMO |
| **Business Systems** | COCOMO | SSE | MRM |

**Table F-2. Projection Summary Table.**

| | First | Second | Third |
|---|---|---|---|
| **Modified Risk Model** | 50% | 37.5% | 12.5% |
| **Basic COCOMO** | 37.5% | 37.5% | 25% |
| **Simplified Software Equation** | 12.5% | 25% | 62.5% |

**Table F-3. Projection Summary Percentages.**

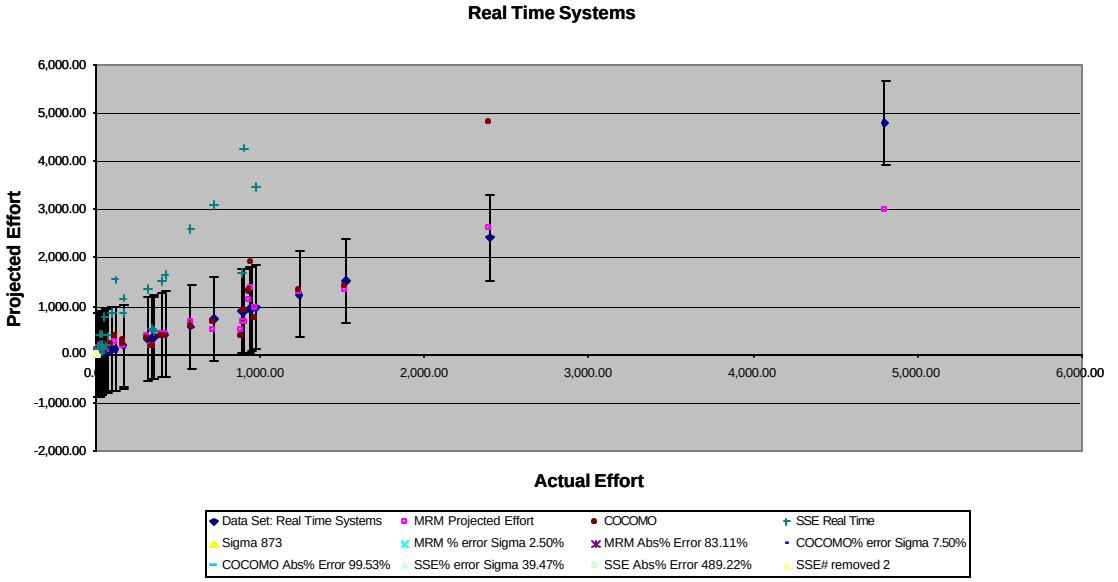
Each of the remaining sub-sections in this appendix follows the same format: description, scatter plot, summary table, quad charts. The quad charts are provided for each of the three test models.

- Description: a description is provided to overview of the type of application being demonstrated for the analysis. Since these applications types are contained in the QSM® database, the data dictionary (Appendix A) is used to provide the definitions.
- Scatter Plot: Scatter plots are provided for each of the application types. The vertical and horizontal axis are scaled the same to represent a linear representation of the actual data. The standard deviation of the original data's trend line provides insight to the variability in the data. Also, each model's performance is projected to illustrate accuracy of the model.

- Summary Table: The summary table presents information in the same format illustrated in the summary tables of Chapter VII. The evaluation criteria remain the same as well.
- Quad Charts: The quad charts illustrate the each model's performance on the actual error, absolute error, under and over projection error.

## 1. Real Time Systems

**Real Time.** Software that must operate close to the processing limits of the CPU. This is interrupt driven software and is often written in C, Ada or Assembly language. Typical examples are military systems like radar, signal processors, missile guidance systems, etc. Figure F-1 is a scatter plot of the real time systems in the project subset.




| | | | | | Actual Error (wt 5) | | | | Absolute Error (wt 4) | | | | Balance (wt 3) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data Set: Real Time Systems | N | 5% | Sigma | Correl | Mean | Std | Rank | Score | Mean | Std | Rank | Score | % Under | Rank | Score |
| MRM | 40 | 2 | 873 | 0.9478 | 69.24% | 108.31% | 1 | 5 | 83.11% | 97.78% | 1 | 4 | 26.32% | 1 | 3 |
| COCOMO | 40 | 2 | 873 | 0.9605 | 87.46% | 126.26% | 2 | 10 | 99.53% | 116.72% | 2 | 8 | 23.68% | 2 | 6 |
| SSE | 40 | 2 | 873 | 0.9582 | 489.22% | 374.31% | 3 | 15 | 489.22% | 374.31% | 3 | 12 | 0.00% | 3 | 9 |

| | | Under Estimation (wt 2) | | | | | Over Estimation (wt 1) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SSE required 2 projects removed. | | N-Under | Mean | Std | Rank | Score | N-Over | Mean | Std | Rank | Score | Total |
| | MRM | 10 | -26.35% | 20.07% | 2 | 4 | 28 | 103.38% | 106.53% | 1 | 1 | 17 |
| | COCOMO | 9 | -25.47% | 22.17% | 3 | 6 | 29 | 122.51% | 124.71% | 2 | 2 | 32 |
| | SSE | 0 | n/a | n/a | 1 | 2 | 36 | 489.22% | 374.31% | 3 | 3 | 41 |

**Figure F-1. Real Time Systems.**

Real Time systems contained 40 projects. Each model had a total of two projects removed from the validation. The average standard deviation is 873 man-months. Each model achieved a correlation coefficient around 95%. A total of two projects were removed from consideration for the Simplified Software Equation. Compared with the other two models, the MRM placed first followed by the Basic COCOMO and the Simplified Software Equation respectively.

### *a. Modified Risk Model*

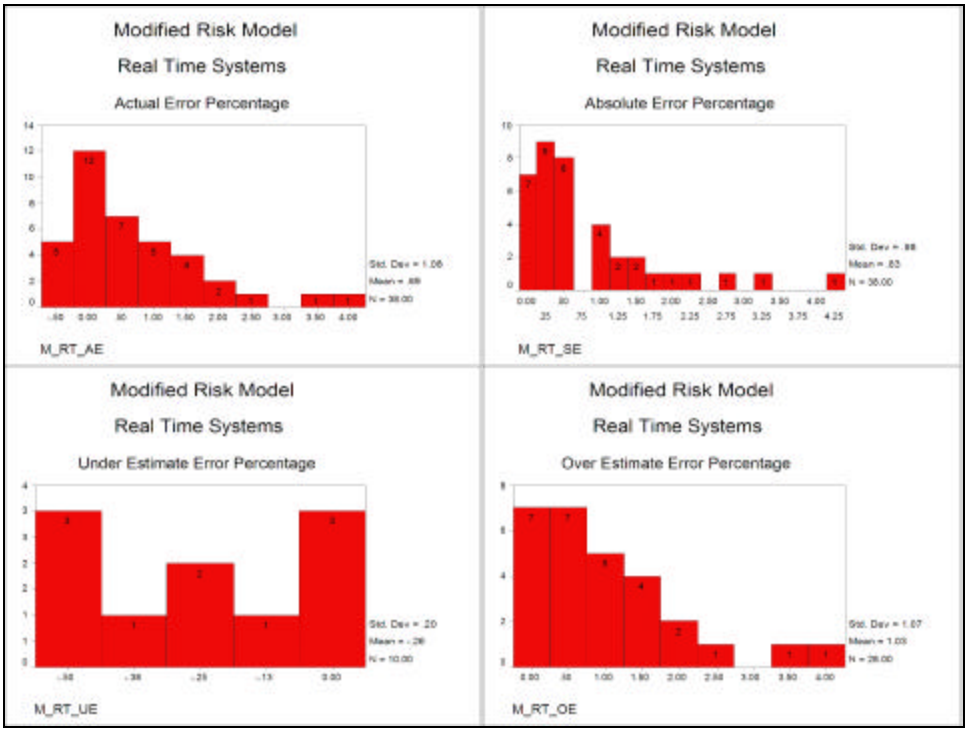



**Figure F-2. MRM Performance on Real Time Systems.**

The Modified Risk Model projected 31.58% of all of the *real time* applications within 25% of the actual value. The MRM ranked 1st in overall model performance for *real time* systems:

- Actual Error – The mean error is 0.69 or an average of 69% from the actual value. The average error has a standard deviation of 108%. Twelve projects, or 31.5%, are projected within 25% of the actual value. The MRM ranked 1 of 3 for average actual error.

- Absolute Error – The MRM projected the actual project performance with an absolute error of 83% ± 98%. The MRM ranked 1 of 3 for average absolute error.
- Under Estimate – The MRM projected 10 out of 38 projects under the actual value, 26.3%. When the MRM projects short, it does so with an average error of 26%. The majority (100%) of the under estimates occur between zero and 50 percent. The MRM ranked 1 of 3 for balance and 2 of 3 for average under-estimation error.
- Over Estimation – The MRM over estimated 28 projects. Fifty percent of the over-estimates were between zero and 50 percent. The MRM ranked 1 of 3 for average over-estimation error.

### *b. Basic COCOMO Model*

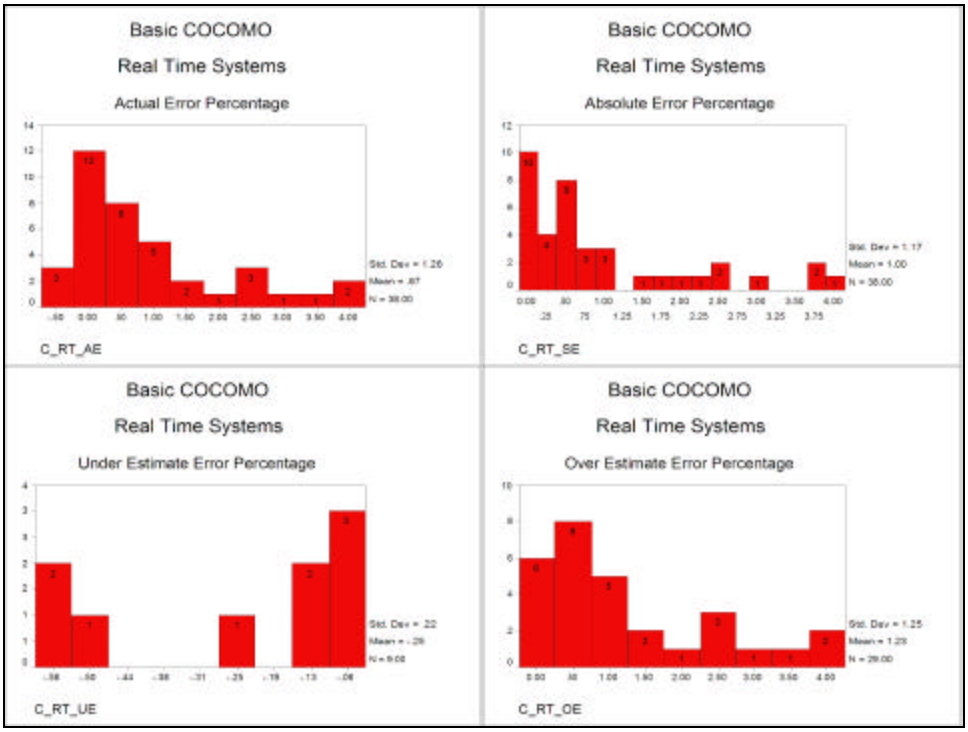



**Figure F-3. COCOMO Performance on Real Time Systems.**

The Basic COCOMO ranked 2nd in overall model performance for *real time* systems:

- Actual Error – The mean error is 0.87 or an average of 87% from the actual value. The average error has a standard deviation of 126%. Twelve

projects, or 31.5%, are projected within 25% of the actual value. The Basic COCOMO ranked 2 of 3 for average actual error.

- Absolute Error – The Basic COCOMO projected the actual project performance with an absolute error of 100% ± 117%. The Basic COCOMO ranked 2 of 3 for average absolute error.
- Under Estimate – The Basic COCOMO projected 9 out of 38 projects under the actual value, 23.6%. When the Basic COCOMO projects short, it does so with an average error of 25%. The majority (77%) of the under estimates occur between zero and 50 percent. The Basic COCOMO ranked 2 of 3 for balance and 3 of 3 for average under-estimation error.
- Over Estimation – The Basic COCOMO over estimated 29 projects. Forty-eight percent of the over-estimates were between zero and 50 percent. The Basic COCOMO ranked 2 of 3 for average over-estimation error.

### *c. Simplified Software Equation*

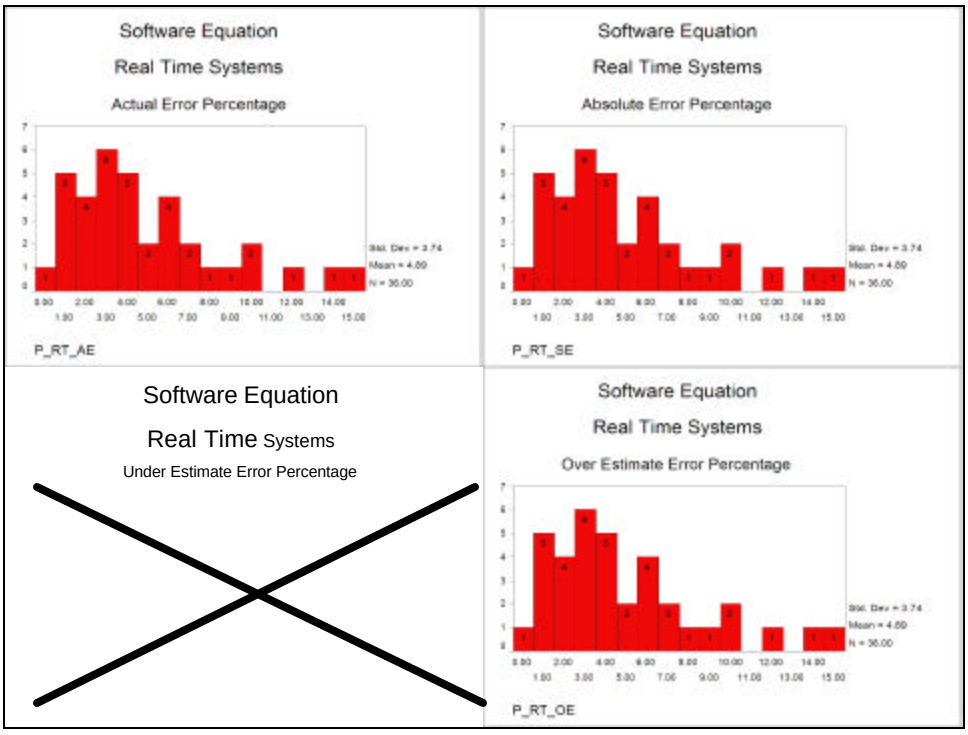


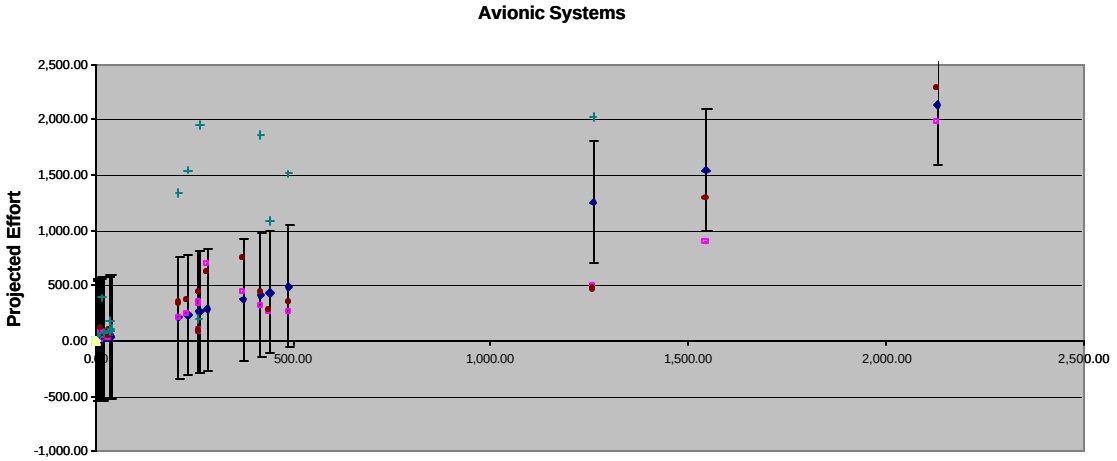

**Figure F-4. SSE Performance on Real Time Systems.**

The Simplified Software Equation ranked 3rd in overall model performance for *real time* systems:

- Actual Error – The mean error is 4.89 or an average of 489% from the actual value. The average error has a standard deviation of 374%. One project is projected within 50% of the actual value. The SSE ranked 3 of 3 for average actual error.
- Absolute Error – The SSE projected the actual project performance with an absolute error of 489% ± 374%. The SSE ranked 3 of 3 for average absolute error.
- Under Estimate – The SSE did not under-estimate any projects. The SSE ranked 3 of 3 for balance and 1 of 3 for average under-estimation error.
- Over Estimation – The SSE over estimated 36 projects. One project, 2.7%, was over-estimated within 50 percent. The SSE ranked 3 of 3 for average over-estimation error.

### 2. Avionic Systems

**Avionics.** Software that is on-board and controls the flight and operation of the aircraft. Figure F-5 is a scatter plot of the avionic systems in the project subset.




| | | | | | Actual Error (wt 5) | | | | Absolute Error (wt 4) | | | | Balance (wt 3) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data Set: Avionic Systems | N | 5% | Sigma | Correl | Mean | Std | Rank | Score | Mean | Std | Rank | Score | % Under | Rank | Score |
| MRM | 23 | 1 | 552 | 0.9096 | 12.03% | 79.31% | 1 | 5 | 52.22% | 59.87% | 1 | 4 | 50.00% | 1 | 3 |
| COCOMO | 23 | 1 | 552 | 0.9120 | 55.28% | 101.14% | 2 | 10 | 76.21% | 89.34% | 2 | 8 | 27.27% | 2 | 6 |
| SSE | 23 | 1 | 552 | 0.9038 | 382.25% | 284.02% | 3 | 15 | 384.76% | 280.42% | 3 | 12 | 5.26% | 3 | 9 |

| | | Under Estimation (wt 2) | | | | | Over Estimation (wt 1) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SSE required 3 projects removed. | | N-Under | Mean | Std | Rank | Score | N-Over | Mean | Std | Rank | Score | Total |
| | MRM | 11 | -40.19% | 37.99% | 3 | 6 | 11 | 64.25% | 75.93% | 1 | 1 | 19 |
| | COCOMO | 6 | -38.37% | 21.78% | 2 | 4 | 16 | 90.40% | 101.14% | 2 | 2 | 30 |
| | SSE | 1 | -23.86% | n/a | 1 | 2 | 18 | 404.81% | 274.18% | 3 | 3 | 41 |

**Figure F-5. Avionic Systems.**

Avionic systems contained 23 projects. Each model had a total of one project removed from the validation. The average standard deviation is 552 man-months. Each model achieved a correlation coefficient around 90%. A total of three projects were removed from consideration for the Simplified Software Equation. Compared with the other two models, overall the MRM placed first followed by the Basic COCOMO and the Simplified Software Equation respectively.

### *a. Modified Risk Model*

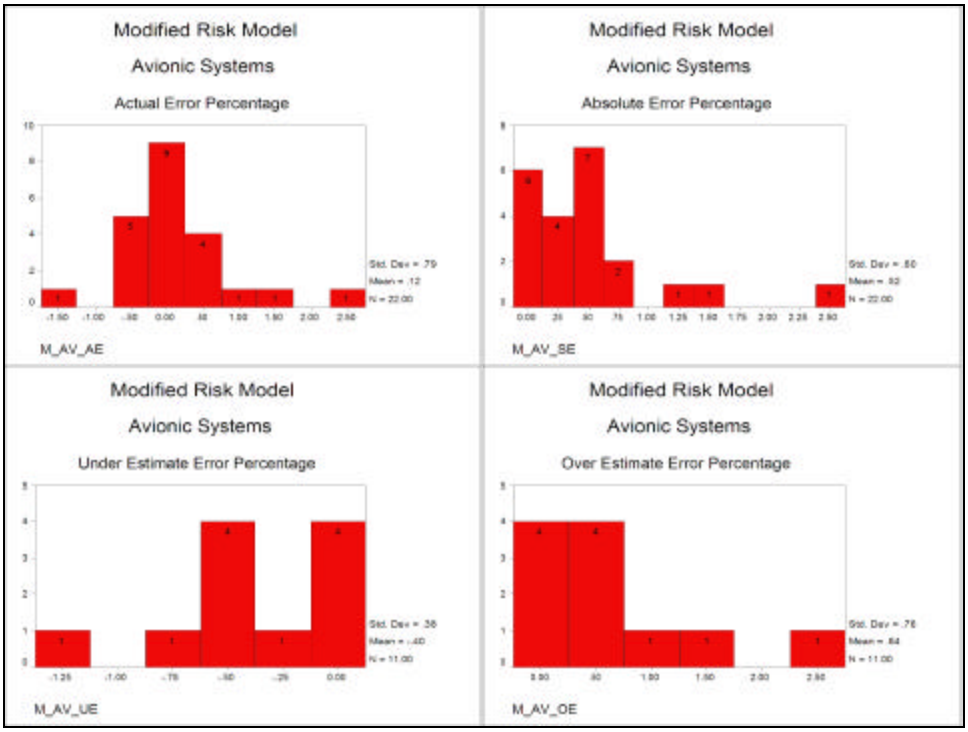



**Figure F-6. MRM Performance on Avionic Systems.**

The Modified Risk Model projected 40.91% of all of the *avionic* applications within 25% of the actual value. The MRM ranked 1st in overall model performance for *avionic* systems:

- Actual Error – The mean error is 0.12 or an average of 12% from the actual value. The average error has a standard deviation of 79%. Nine projects, or 41%, are projected within 25% of the actual value. The MRM ranked 1 of 3 for average actual error.
- Absolute Error – The MRM projected the actual project performance with an absolute error of 52% ± 60%. The MRM ranked 1 of 3 for average absolute error.
- Under Estimate – The MRM projected 11 out of 22 projects under the actual value, 50%. When the MRM projects short, it does so with an average error of 40%. The majority (82%) of the under estimates occur between zero and 50 percent. The MRM ranked 1 of 3 for balance and 3 of 3 for average under-estimation error.

- Over Estimation – The MRM over estimated 11 projects. Seventy-three percent of the over-estimates were between zero and 50 percent. The MRM ranked 1 of 3 for average over-estimation error.

### *b. Basic COCOMO Model*

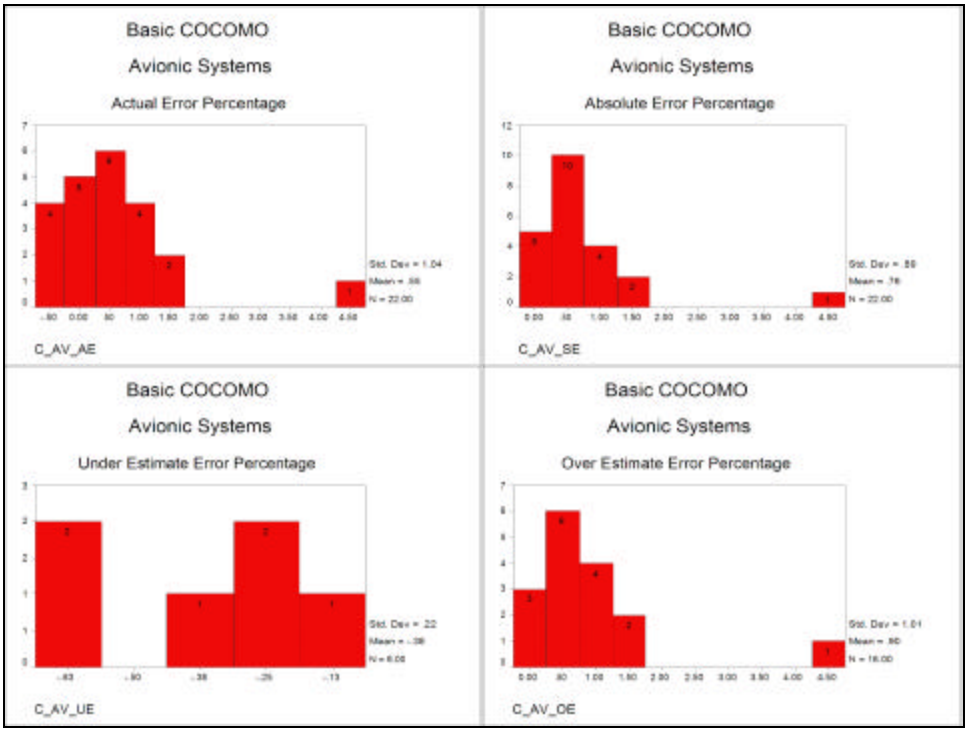



**Figure F-7. COCOMO Performance on Avionic Systems.**

The Basic COCOMO ranked 2nd in overall model performance for *avionic* systems:

- Actual Error – The mean error is 0.55 or an average of 55% from the actual value. The average error has a standard deviation of 104%. Five projects, or 23%, are projected within 25% of the actual value. The Basic COCOMO ranked 2 of 3 for average actual error.
- Absolute Error – The Basic COCOMO projected the actual project performance with an absolute error of 76% ± 89%. The Basic COCOMO ranked 2 of 3 for average absolute error.
- Under Estimate – The Basic COCOMO projected 6 out of 22 projects under the actual value, 27%. When the Basic COCOMO projects short, it does so with an average error of 38%. The majority (67%) of the under

estimates occur between zero and 50 percent. The Basic COCOMO ranked 2 of 3 for balance and 2 of 3 for average under-estimation error.

- Over Estimation – The Basic COCOMO over estimated 16 projects. Fifty-six percent of the over-estimates were between zero and 50 percent. The Basic COCOMO ranked 2 of 3 for average over-estimation error.

### *c. Simplified Software Equation*

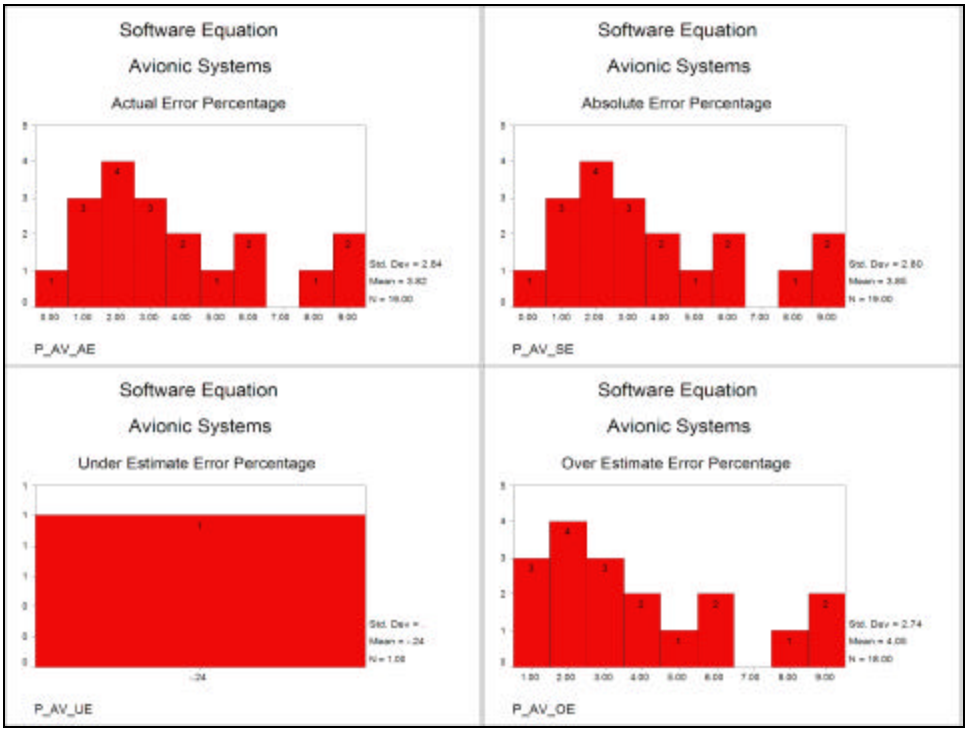



**Figure F-8. SSE Performance on Avionic Systems.**

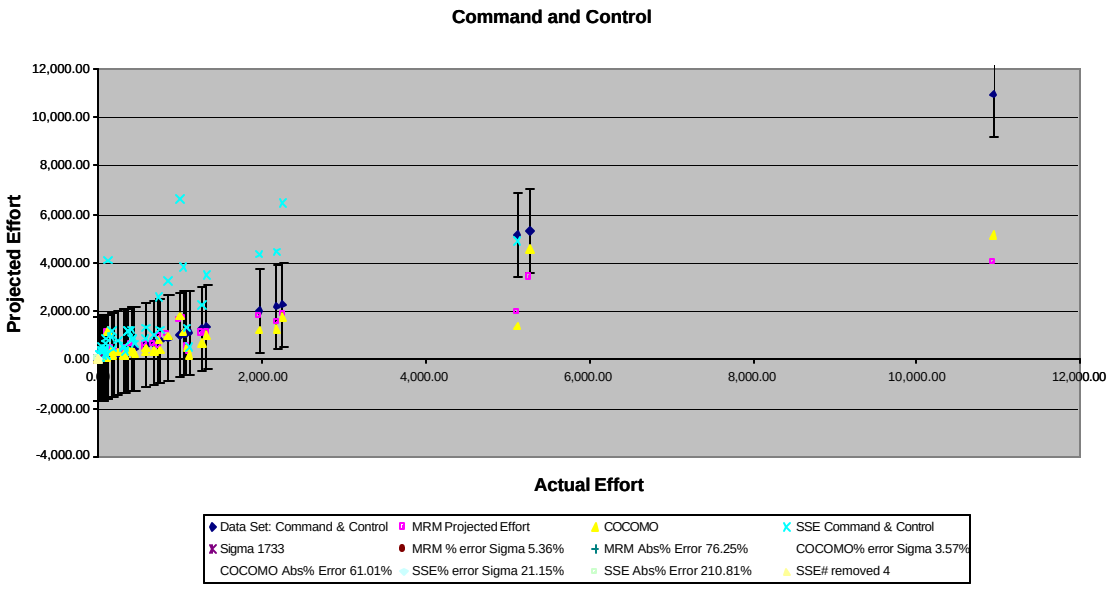
The Simplified Software Equation ranked 3rd in overall model performance for *avionic* systems:

- Actual Error – The mean error is 3.82 or an average of 382% from the actual value. The average error has a standard deviation of 284%. One project is projected within 25% of the actual value. The SSE ranked 3 of 3 for average actual error.
- Absolute Error – The SSE projected the actual project performance with an absolute error of 385% ± 280%. The SSE ranked 3 of 3 for average absolute error.

- Under Estimate – The SSE projected one of 19 projects under the actual value, 5.2%. When the SSE projects short, it does so with an average error of 24%. The SSE ranked 3 of 3 for balance and 1 of 3 for average under-estimation error.
- Over Estimation – The SSE over estimated 18 projects. Two projects were over-estimated within 50 percent. The SSE ranked 3 of 3 for average over-estimation error.

### 3. Command and Control Systems

**Command & Control.** Software that allows humans to manage a dynamic situation and respond in human real-time. Examples are battle field command systems, telephone network control systems, government disaster response systems, military intelligence systems, electric utility power control systems. Figure F-9 is a scatter plot of the command and control systems in the project subset.




| | | | | | Actual Error (wt 5) | | | | Absolute Error (wt 4) | | | | Balance (wt 3) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data Set: Command & Control | N | 5% | Sigma | Correl | Mean | Std | Rank | Score | Mean | Std | Rank | Score | % Under | Rank | Score |
| MRM | 56 | 3 | 1,733 | 0.8968 | 45.10% | 97.24% | 2 | 10 | 76.25% | 74.85% | 2 | 8 | 43.40% | 2 | 6 |
| COCOMO | 56 | 3 | 1,733 | 0.9056 | 26.62% | 92.84% | 1 | 5 | 61.01% | 74.49% | 1 | 4 | 47.17% | 1 | 3 |
| SSE | 56 | 3 | 1,733 | 0.9089 | 208.24% | 211.39% | 3 | 15 | 208.24% | 208.77% | 3 | 12 | 4.08% | 3 | 9 |

| | Under Estimation (wt 2) | | | | | Over Estimation (wt 1) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| SSE required 4 projects removed. | N-Under | Mean | Std | Rank | Score | N-Over | Mean | Std | Rank | Score | Total |
| MRM | 23 | -35.89% | 20.20% | 2 | 4 | 30 | 107.20% | 86.33% | 2 | 2 | 30 |
| COCOMO | 25 | -36.45% | 20.50% | 3 | 6 | 28 | 82.94% | 96.20% | 1 | 1 | 19 |
| SSE | 2 | -31.44% | 37.39% | 1 | 2 | 47 | 218.44% | 209.74% | 3 | 3 | 41 |

**Figure F-9. Command and Control Systems.**

Command and Control systems contained 56 projects. Each model had a total of three projects removed from the validation. The average standard deviation is 1733 man-months. Each model achieved a correlation coefficient around 90%. A total of four projects were removed from consideration for the Simplified Software Equation. Compared with the other two models, overall the Basic COCOMO placed first followed by the MRM and the Simplified Software Equation respectively.

### *a. Modified Risk Model*

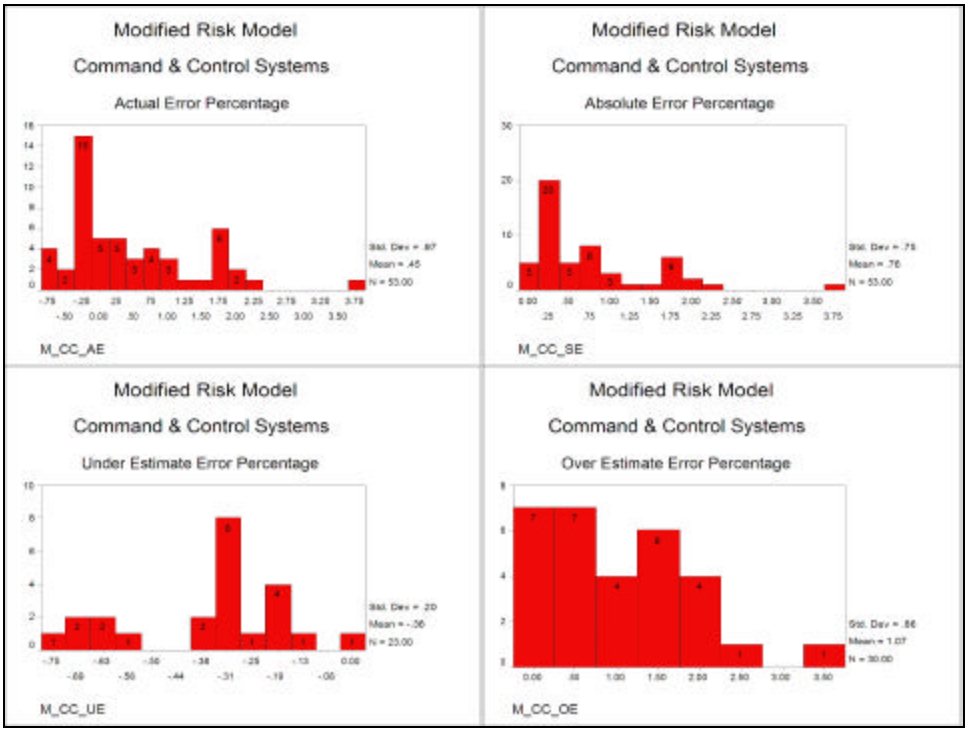



**Figure F-10. MRM Performance on Command & Control Systems.**

The Modified Risk Model projected 24.53% of all of the *command & control* applications within 25% of the actual value. The MRM ranked 2nd in overall model performance for *command and control* systems:

- Actual Error – The mean error is 0.45 or an average of 45% from the actual value. The average error has a standard deviation of 97%. Twenty-eight projects, or 53%, are projected within 25% of the actual value. The MRM ranked 2 of 3 for average actual error.

- Absolute Error – The MRM projected the actual project performance with an absolute error of 76% ± 75%. The MRM ranked 2 of 3 for average absolute error.
- Under Estimate – The MRM projected 23 out of 53 projects under the actual value, 43%. When the MRM projects short, it does so with an average error of 36%. The majority (74%) of the under estimates occur between zero and 50 percent. The MRM ranked 2 of 3 for balance and 2 of 3 for average under-estimation error.
- Over Estimation – The MRM over estimated 30 projects. Forty-seven percent of the over-estimates were between zero and 50 percent. The MRM ranked 2 of 3 for average over-estimation error.

### *b. Basic COCOMO Model*

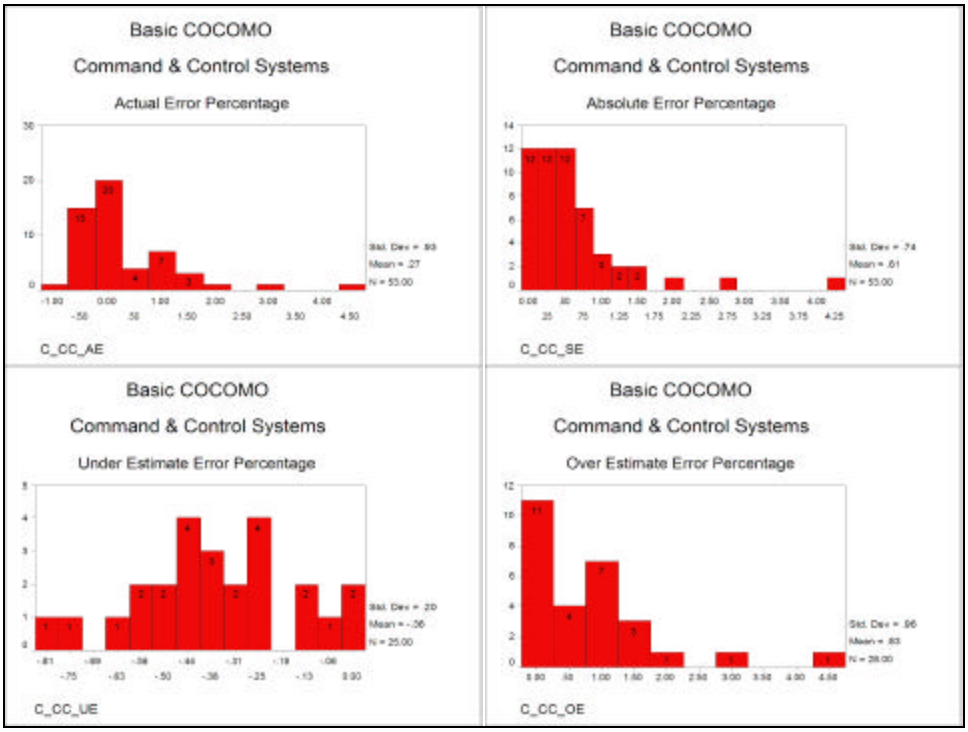



**Figure F-11. COCOMO Performance on Command & Control Systems.**

The Basic COCOMO ranked 1 in overall model performance for *command and control* systems:

- Actual Error – The mean error is 0.27 or an average of 27% from the actual value. The average error has a standard deviation of 93%. Thirty-

six projects, or 68%, are projected within 25% of the actual value. The Basic COCOMO ranked 1 of 3 for average actual error.

- Absolute Error – The Basic COCOMO projected the actual project performance with an absolute error of 61% ± 74%. The Basic COCOMO ranked 1 of 3 for average absolute error.
- Under Estimate – The Basic COCOMO projected 25 out of 53 projects under the actual value, 47%. When the Basic COCOMO projects short, it does so with an average error of 36%. The majority (80%) of the under estimates occur between zero and 50 percent. The Basic COCOMO ranked 1 of 3 for balance and 3 of 3 for average under-estimation error.
- Over Estimation – The Basic COCOMO over estimated 28 projects. Fifty-four percent of the over-estimates were between zero and 50 percent. The Basic COCOMO ranked 1 of 3 for average over-estimation error.

### *c. Simplified Software Equation*

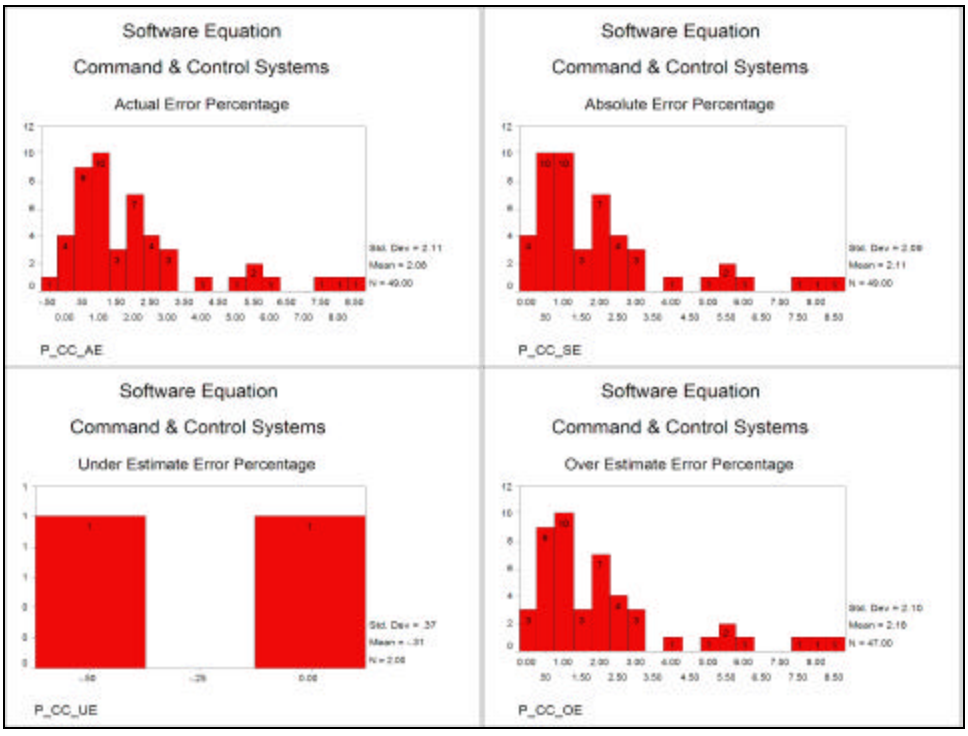


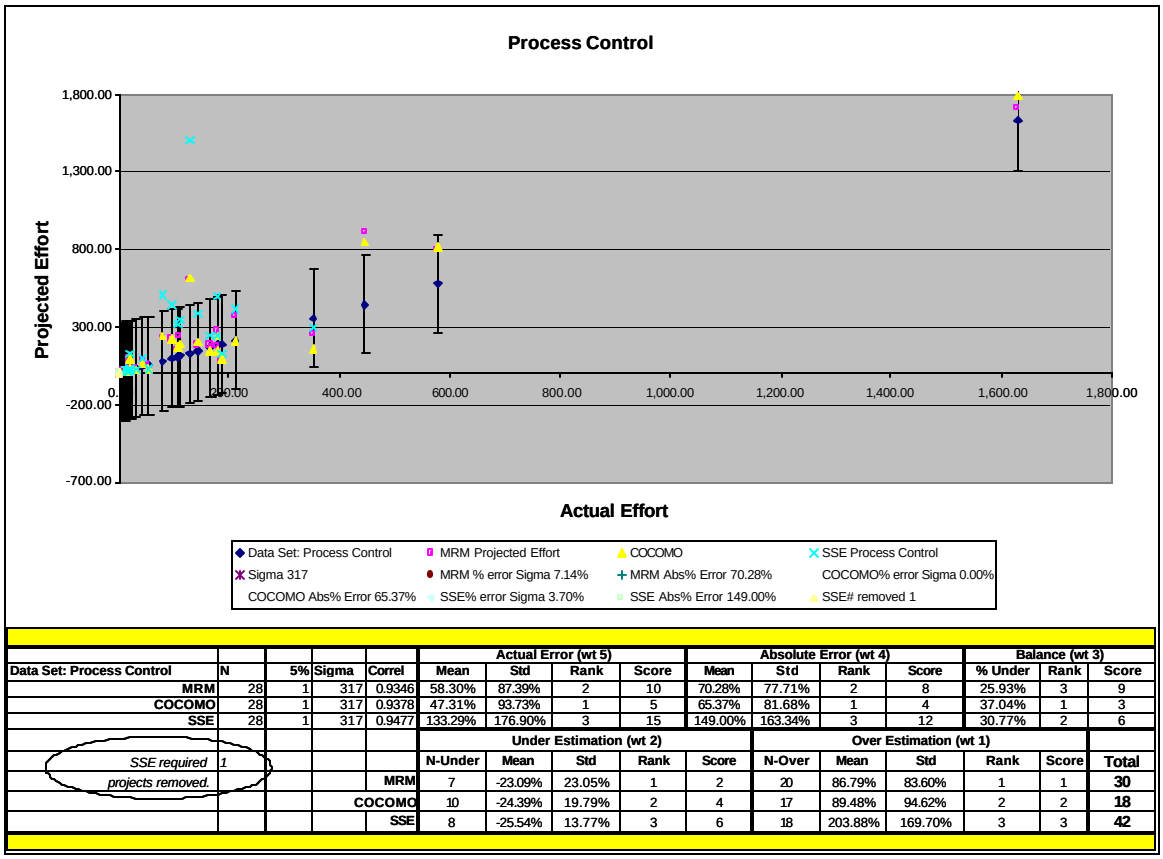

**Figure F-12. SSE Performance on Command & Control Systems.**

The Simplified Software Equation ranked 3rd in overall model performance for *command and control* systems:

- Actual Error – The mean error is 2.08 or an average of 208% from the actual value. The average error has a standard deviation of 211%. Fourteen projects are projected within 25% of the actual value. The SSE ranked 3 of 3 for average actual error.
- Absolute Error – The SSE projected the actual project performance with an absolute error of 211% ± 209%. The SSE ranked 3 of 3 for average absolute error.
- Under Estimate – The SSE projected two of 49 projects under the actual value, 4%. When the SSE projects short, it does so with an average error of 31%. The SSE ranked 3 of 3 for balance and 1 of 3 for average under-estimation error.
- Over Estimation – The SSE over estimated 47 projects. Twelve projects were over-estimated within 50 percent. The SSE ranked 3 of 3 for average over-estimation error.

### 4. Process Control

**Process Control.** Software that controls an automated system. Examples are software that runs a nuclear power plant or software that runs a petro-chemical plant. Figure F-13 is a scatter plot of the process control systems in the project subset.




| | | | | | Actual Error (wt 5) | | | | Absolute Error (wt 4) | | | | Balance (wt 3) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data Set: Process Control | N | 5% | Sigma | Correl | Mean | Std | Rank | Score | Mean | Std | Rank | Score | % Under | Rank | Score |
| MRM | 28 | 1 | 317 | 0.9346 | 58.30% | 87.39% | 2 | 10 | 70.28% | 77.71% | 2 | 8 | 25.93% | 3 | 9 |
| COCOMO | 28 | 1 | 317 | 0.9378 | 47.31% | 93.73% | 1 | 5 | 65.37% | 81.68% | 1 | 4 | 37.04% | 1 | 3 |
| SSE | 28 | 1 | 317 | 0.9477 | 133.29% | 176.90% | 3 | 15 | 149.00% | 163.34% | 3 | 12 | 30.77% | 2 | 6 |

| | Under Estimation (wt 2) | | | | | Over Estimation (wt 1) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| SSE required 1 projects removed. | N-Under | Mean | Std | Rank | Score | N-Over | Mean | Std | Rank | Score | Total |
| MRM | 7 | -23.09% | 23.05% | 1 | 2 | 20 | 86.79% | 83.60% | 1 | 1 | 30 |
| COCOMO | 10 | -24.39% | 19.79% | 2 | 4 | 17 | 89.48% | 94.62% | 2 | 2 | 18 |
| SSE | 8 | -25.54% | 13.77% | 3 | 6 | 18 | 203.88% | 169.70% | 3 | 3 | 42 |

**Figure F-13. Process Control Systems.**

Process Control systems contained 28 projects. Each model had one project removed from the validation. The average standard deviation is 317 man-months. Each model achieved a correlation coefficient around 93%. A total of one project was removed from consideration for the Simplified Software Equation. Compared with the other two models, overall the Basic COCOMO placed first followed by the MRM and the Simplified Software Equation respectively.

### *a. Modified Risk Model*

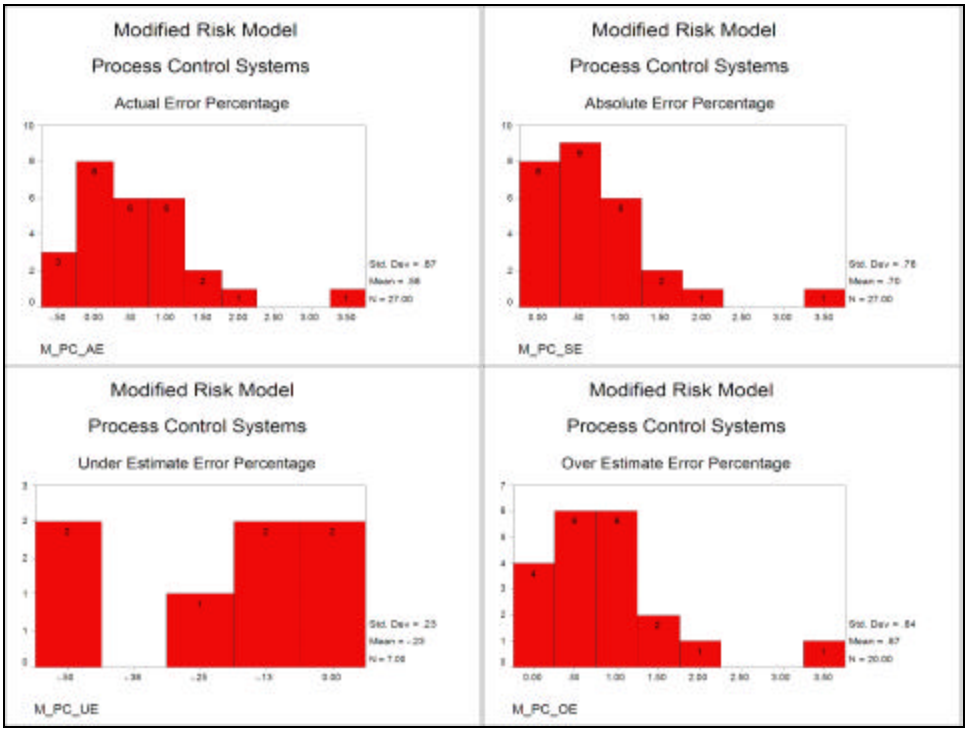



**Figure F-14. MRM Performance on Process Control Systems.**

The Modified Risk Model projected 29.63% of all of the *process control* applications within 25% of the actual value. The MRM ranked 2nd in overall model performance for *process control* systems:

- Actual Error – The mean error is 0.58 or an average of 58% from the actual value. The average error has a standard deviation of 87%. Eight projects, or 30%, are projected within 25% of the actual value. The MRM ranked 2 of 3 for average actual error.
- Absolute Error – The MRM projected the actual project performance with an absolute error of 70% ± 78%. The MRM ranked 2 of 3 for average absolute error.
- Under Estimate – The MRM projected 7 out of 27 projects under the actual value, 26%. When the MRM projects short, it does so with an average error of 23%. The majority (71%) of the under estimates occur between zero and 50 percent. The MRM ranked 3 of 3 for balance and 1 of 3 for average under-estimation error.

- Over Estimation – The MRM over estimated 20 projects. Fifty percent of the over-estimates were between zero and 50 percent. The MRM ranked 1 of 3 for average over-estimation error.

### *b. Basic COCOMO Model*

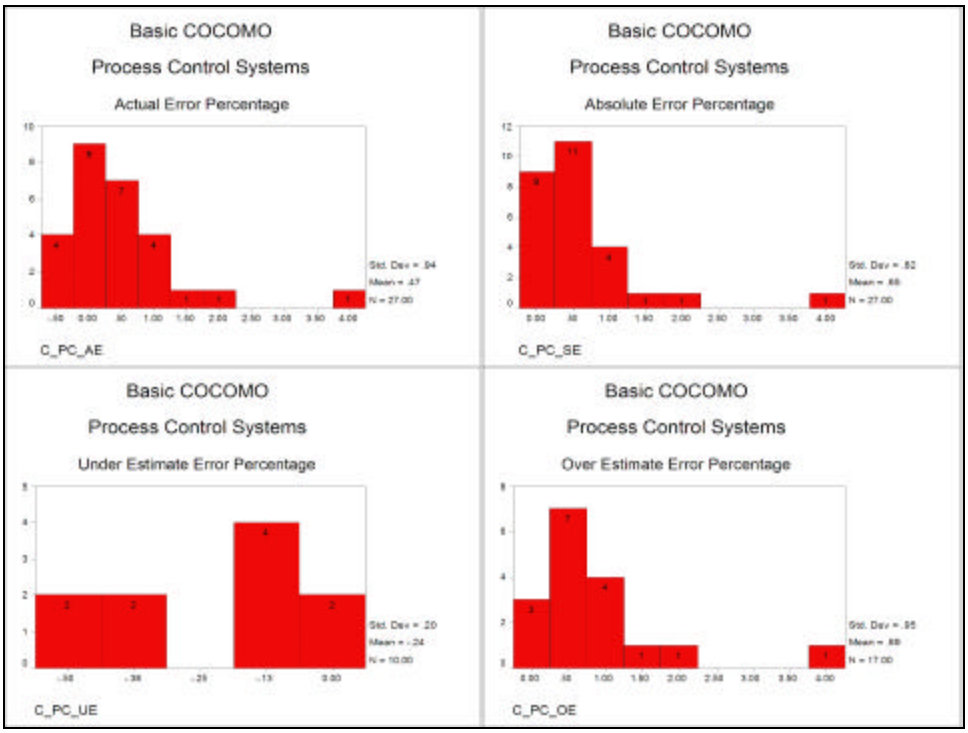



**Figure F-15. COCOMO Performance on Process Control Systems.**

The Basic COCOMO ranked 1st in overall model performance for *process control* systems:

- Actual Error – The mean error is 0.47 or an average of 47% from the actual value. The average error has a standard deviation of 94%. Nine projects, or 33%, are projected within 25% of the actual value. The Basic COCOMO ranked 1 of 3 for average actual error.
- Absolute Error – The Basic COCOMO projected the actual project performance with an absolute error of 65% ± 82%. The Basic COCOMO ranked 1 of 3 for average absolute error.
- Under Estimate – The Basic COCOMO projected 10 out of 27 projects under the actual value, 37%. When the Basic COCOMO projects short, it

does so with an average error of 24%. The majority (80%) of the under estimates occur between zero and 50 percent. The Basic COCOMO ranked 1 of 3 for balance and 2 of 3 for average under-estimation error.

- Over Estimation – The Basic COCOMO over estimated 17 projects. Fifty-nine percent of the over-estimates were between zero and 50 percent. The Basic COCOMO ranked 2 of 3 for average over-estimation error.

### *c. Simplified Software Equation*

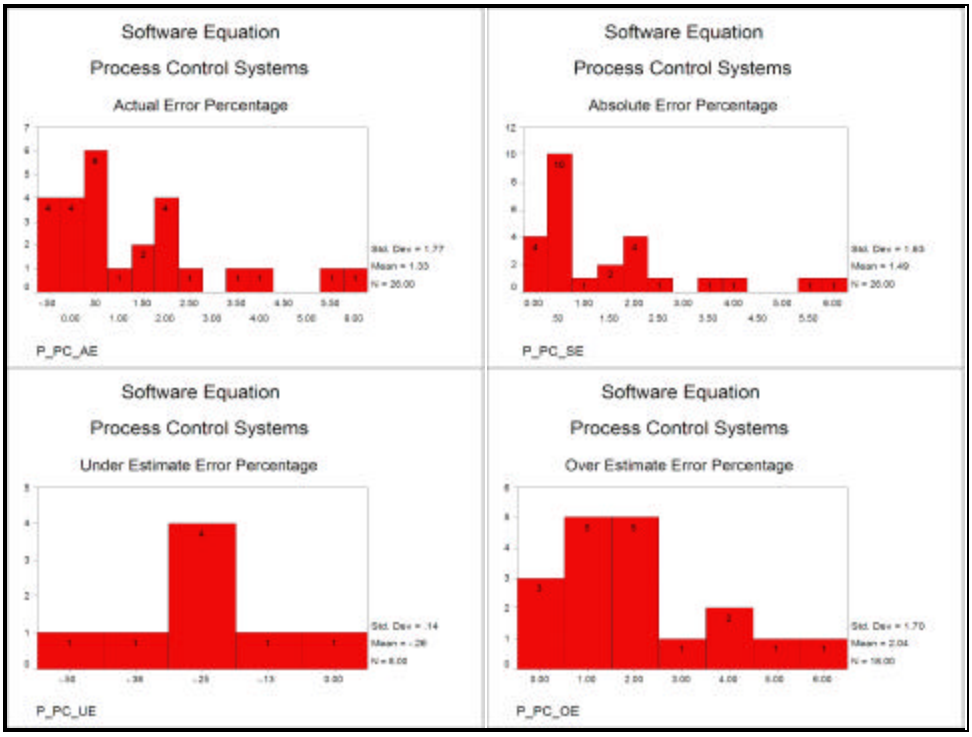



**Figure F-16. SSE Performance on Process Control Systems.**

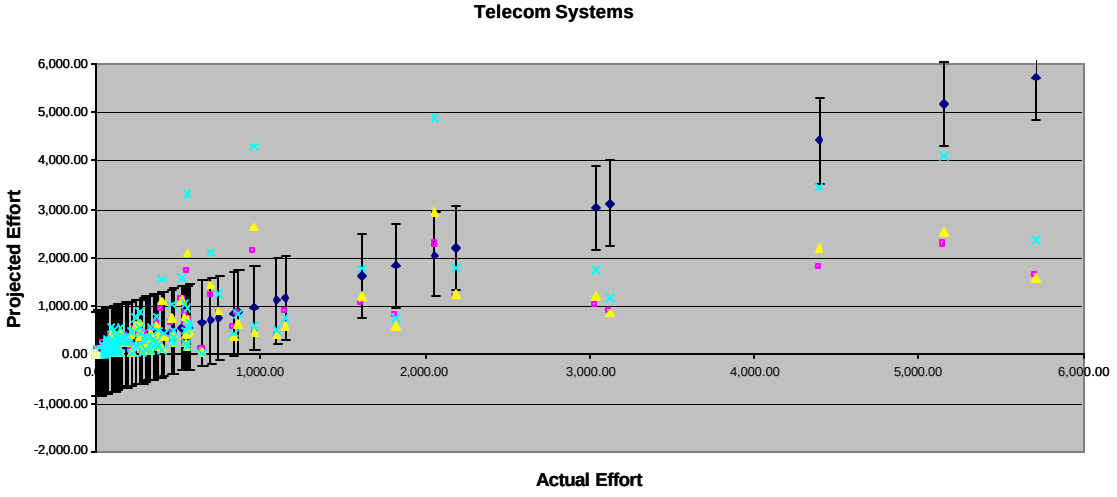
The Simplified Software Equation ranked 3rd in overall model performance for *process control* systems:

- Actual Error – The mean error is 1.33 or an average of 133% from the actual value. The average error has a standard deviation of 177%. Four projects, or 15%, are projected within 25% of the actual value. The SSE ranked 3 of 3 for average actual error.
- Absolute Error – The SSE projected the actual project performance with an absolute error of 149% ± 163%. The SSE ranked 3 of 3 for average absolute error.

- Under Estimate – The SSE projected 8 of 26 projects under the actual value, 31%. When the SSE projects short, it does so with an average error of 26%. The SSE ranked 2 of 3 for balance and 3 of 3 for average under-estimation error.
- Over Estimation – The SSE over estimated 18 projects. One project was over-estimated within 50 percent. The SSE ranked 3 of 3 for average over-estimation error.

## 5. Telecommunication Systems

**Telecommunications.** Software that facilitates the transmission of information from one physical location to another. Examples are telephone switches, transmission systems, modem communication products, fax communication products, satellite communications products. Figure F-17 is a scatter plot of the telecommunication systems in the project subset.




| | | | | | Actual Error (wt 5) | | | | Absolute Error (wt 4) | | | | Balance (wt 3) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data Set: Telecom Systems | N | 5% | Sigma | Correl | Mean | Std | Rank | Score | Mean | Std | Rank | Score | % Under | Rank | Score |
| MRM | 135 | 7 | 873 | 0.7625 | 27.20% | 81.40% | 2 | 10 | 64.44% | 56.45% | 1 | 4 | 45.31% | 3 | 9 |
| COCOMO | 135 | 7 | 873 | 0.7235 | 27.62% | 85.53% | 3 | 15 | 67.49% | 59.11% | 2 | 8 | 46.88% | 2 | 6 |
| SSE | 135 | 7 | 873 | 0.7107 | 22.47% | 90.73% | 1 | 5 | 69.10% | 62.65% | 3 | 12 | 50.00% | 1 | 3 |

| | | Under Estimation (wt 2) | | | | | Over Estimation (wt 1) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SSE required 8 projects removed. | | N-Under | Mean | Std | Rank | Score | N-Over | Mean | Std | Rank | Score | Total |
| | MRM | 58 | -41.09% | 25.76% | 1 | 2 | 70 | 83.78% | 66.92% | 1 | 1 | 26 |
| | COCOMO | 60 | -42.52% | 22.75% | 2 | 4 | 68 | 89.51% | 71.52% | 2 | 2 | 35 |
| | SSE | 60 | -46.63% | 25.71% | 3 | 6 | 60 | 91.56% | 78.93% | 3 | 3 | 29 |

**Figure F-17. Telecommunication Systems.**

Telecommunications systems contained 135 projects. Each model had a total of seven projects removed from the validation. The average standard deviation is 873 man-months. Each model achieved a correlation coefficient around 73%. A total of eight projects were removed from consideration for the Simplified Software Equation. Compared with the other two models, overall the MRM placed first followed by the Simplified Software Equation and the Basic COCOMO respectively.

### *a. Modified Risk Model*

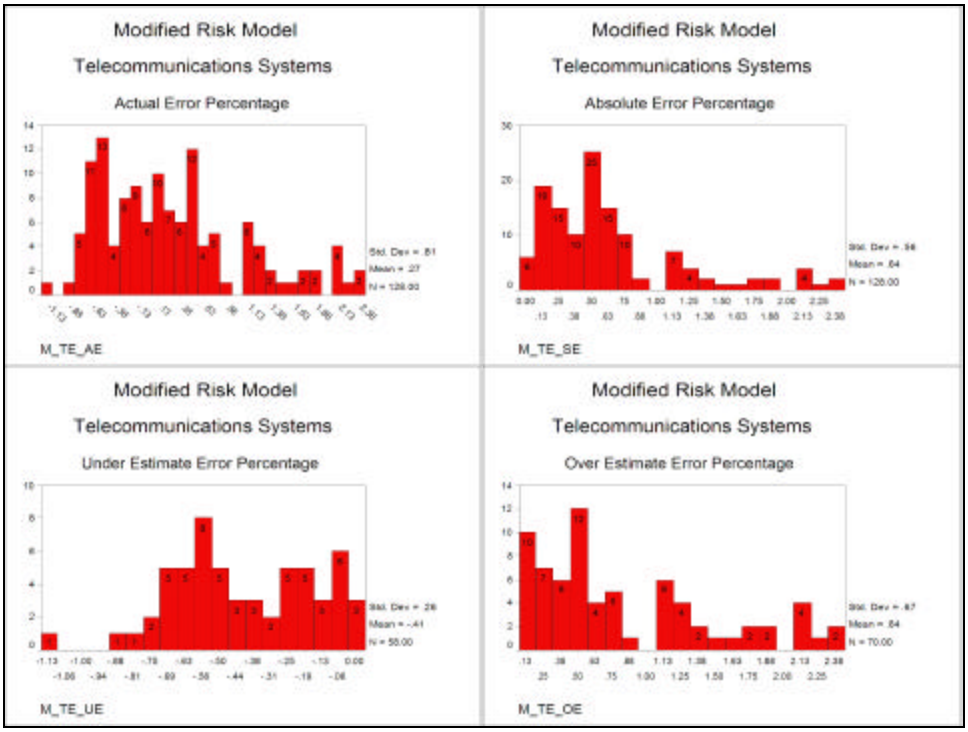



**Figure F-18. MRM Performance on Telecommunication Systems.**

The Modified Risk Model projected 28.13% of all of the *telecommunication* applications within 25% of the actual value. The MRM ranked 1st in overall model performance for *telecommunication* systems:

- Actual Error – The mean error is 0.27 or an average of 27% from the actual value. The average error has a standard deviation of 81%. Forty-

one projects, or 32%, are projected within 25% of the actual value. The MRM ranked 2 of 3 for average actual error.

- Absolute Error – The MRM projected the actual project performance with an absolute error of 64% ± 56%. The MRM ranked 1 of 3 for average absolute error.
- Under Estimate – The MRM projected 58 out of 128 projects under the actual value, 45%. When the MRM projects short, it does so with an average error of 41%. The majority (60%) of the under estimates occur between zero and 50 percent. The MRM ranked 3 of 3 for balance and 1 of 3 for average under-estimation error.
- Over Estimation – The MRM over estimated 70 projects. Fifty percent of the over-estimates were between zero and 50 percent. The MRM ranked 1 of 3 for average over-estimation error.

### *b. Basic COCOMO Model*

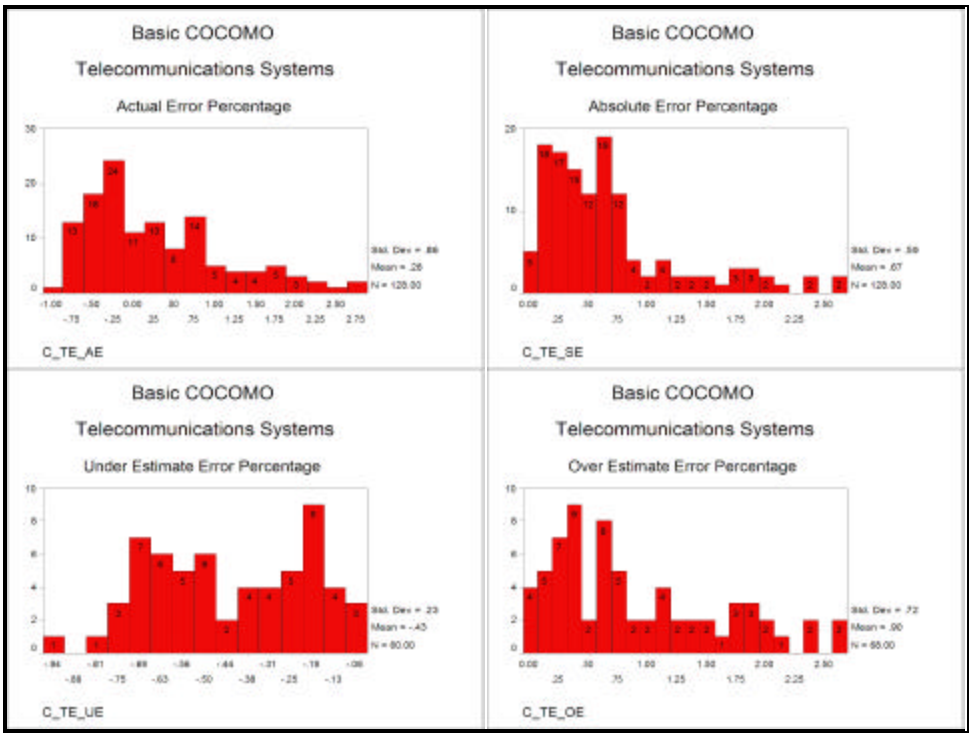



**Figure F-19. COCOMO Performance on Telecommunication Systems.**

The Basic COCOMO ranked 3rd in overall model performance for *telecommunication* systems:

- Actual Error – The mean error is 0.28 or an average of 28% from the actual value. The average error has a standard deviation of 86%. Forty-eight projects, or 38%, are projected within 25% of the actual value. The Basic COCOMO ranked 3 of 3 for average actual error.
- Absolute Error – The Basic COCOMO projected the actual project performance with an absolute error of 67% ± 59%. The Basic COCOMO ranked 2 of 3 for average absolute error.
- Under Estimate – The Basic COCOMO projected 60 out of 128 projects under the actual value, 47%. When the Basic COCOMO projects short, it does so with an average error of 43%. The majority (62%) of the under estimates occur between zero and 50 percent. The Basic COCOMO ranked 2 of 3 for balance and 2 of 3 for average under-estimation error.
- Over Estimation – The Basic COCOMO over estimated 68 projects. Forty percent of the over-estimates were between zero and 50 percent. The Basic COCOMO ranked 2 of 3 for average over-estimation error.

### *c. Simplified Software Equation*

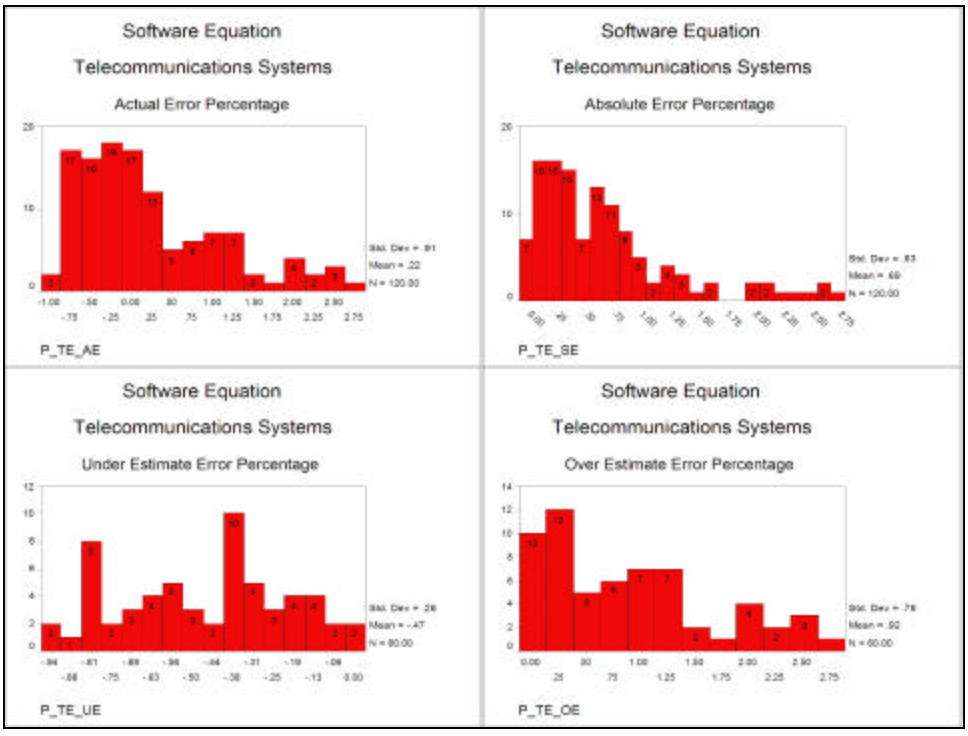


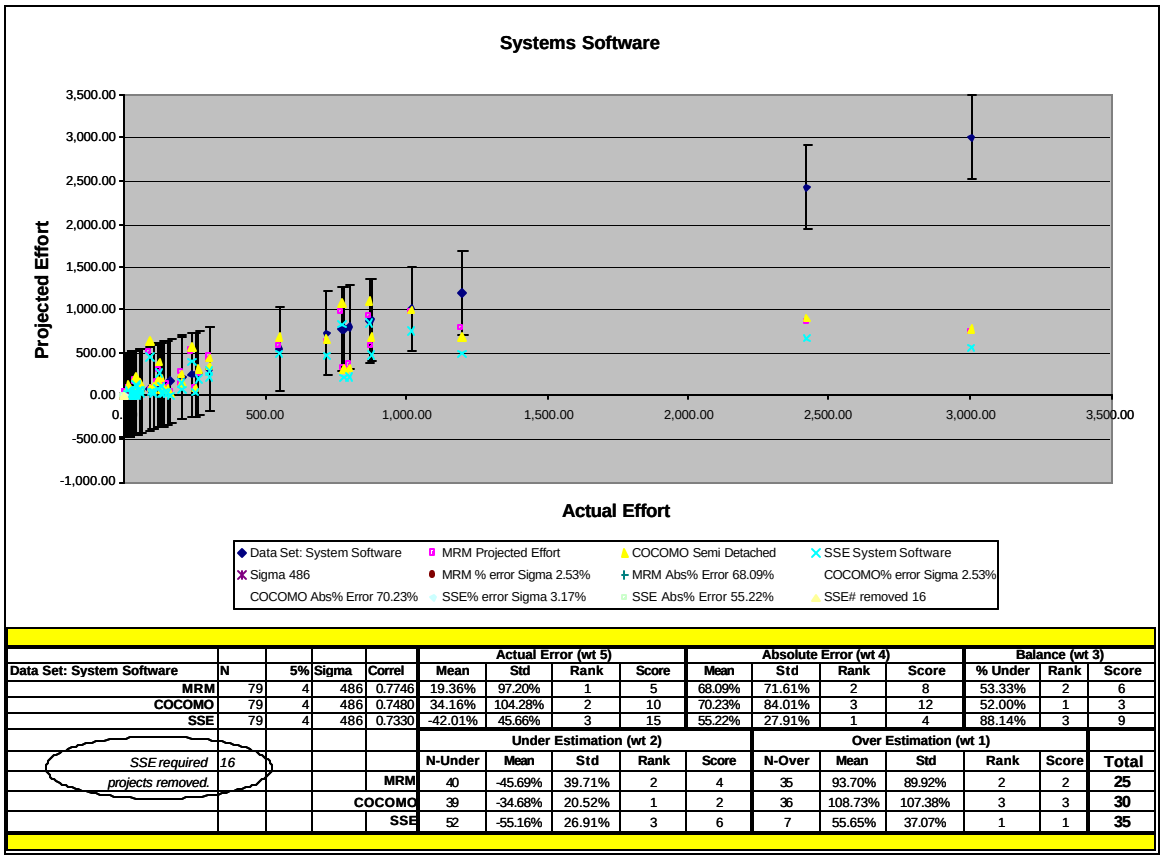

**Figure F-20. SSE Performance on Telecommunication Systems.**

The Simplified Software Equation ranked 2nd in overall model performance for *telecommunication* systems:

- Actual Error – The mean error is .22 or an average of 22% from the actual value. The average error has a standard deviation of 91%. Sixty-six projects, 55%, are projected within 25% of the actual value. The SSE ranked 1 of 3 for average actual error.
- Absolute Error – The SSE projected the actual project performance with an absolute error of 69% ± 63%. The SSE ranked 3 of 3 for average absolute error.
- Under Estimate – The SSE projected 60 of 120 projects under the actual value, 50%. When the SSE projects short, it does so with an average error of 47%. The SSE ranked 1 of 3 for balance and 3 of 3 for average under-estimation error.
- Over Estimation – The SSE over estimated 60 projects. Twenty-two projects were over-estimated within 50 percent. The SSE ranked 3 of 3 for average over-estimation error.

### 6. Systems Software

**System Software.** Layers of software that sit between the hardware and applications programs. Examples are operating systems (DOS, UNIX, VMS, etc.), GUI's (graphical user interfaces – Windows, Xwindows etc.), Executives or Database Management systems, Network products, and Image processing products. Figure F-21 is a scatter plot of systems software in the project subset.




| | | | | | Actual Error (wt 5) | | | | Absolute Error (wt 4) | | | | Balance (wt 3) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data Set: System Software | N | 5% | Sigma | Correl | Mean | Std | Rank | Score | Mean | Std | Rank | Score | % Under | Rank | Score |
| MRM | 79 | 4 | 486 | 0.7746 | 19.36% | 97.20% | 1 | 5 | 68.09% | 71.61% | 2 | 8 | 53.33% | 2 | 6 |
| COCOMO | 79 | 4 | 486 | 0.7480 | 34.16% | 104.28% | 2 | 10 | 70.23% | 84.01% | 3 | 12 | 52.00% | 1 | 3 |
| SSE | 79 | 4 | 486 | 0.7330 | -42.01% | 45.66% | 3 | 15 | 55.22% | 27.91% | 1 | 4 | 88.14% | 3 | 9 |

| | Under Estimation (wt 2) | | | | | Over Estimation (wt 1) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| SSE required 16 projects removed. | N-Under | Mean | Std | Rank | Score | N-Over | Mean | Std | Rank | Score | Total |
| MRM | 40 | -45.69% | 39.71% | 2 | 4 | 35 | 93.70% | 89.92% | 2 | 2 | 25 |
| COCOMO | 39 | -34.68% | 20.52% | 1 | 2 | 36 | 108.73% | 107.38% | 3 | 3 | 30 |
| SSE | 52 | -55.16% | 26.91% | 3 | 6 | 7 | 55.65% | 37.07% | 1 | 1 | 35 |

**Figure F-21. Systems Software.**

Systems Software contained 79 projects. Each model had a total of four projects removed from the validation. The average standard deviation is 486 man-months. Each model achieved a correlation coefficient around 75%. A total of sixteen projects were removed from consideration for the Simplified Software Equation. Compared with the other two models, overall the MRM placed first followed by the Basic COCOMO and the Simplified Software Equation respectively.

### *a. Modified Risk Model*

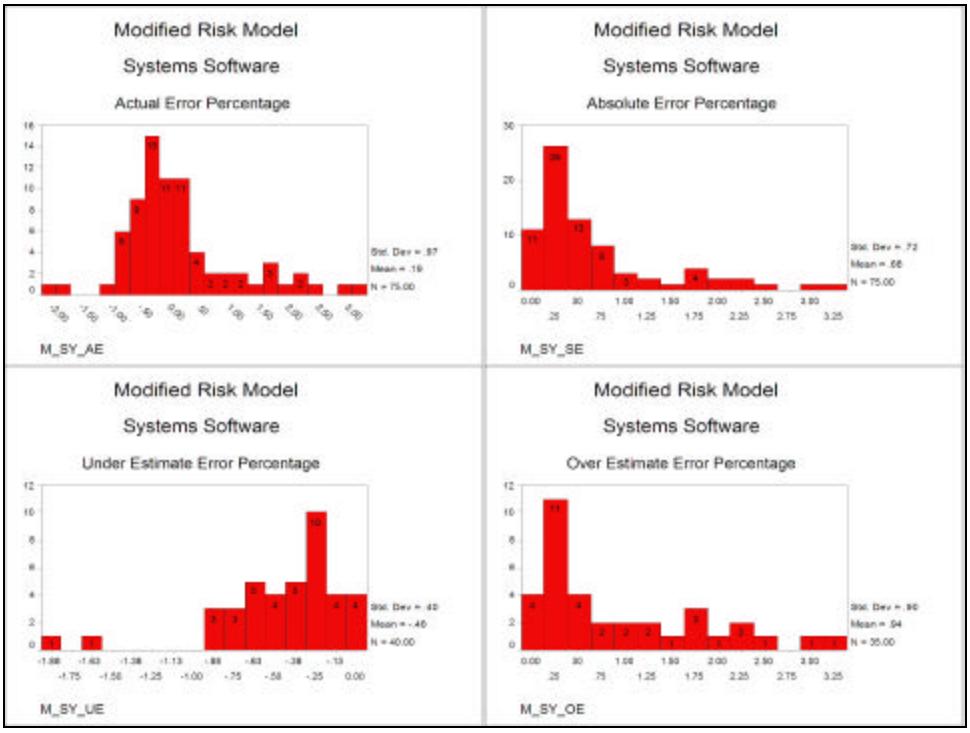



**Figure F-22. MRM Performance on Systems Software.**

The Modified Risk Model projected 25.33% of all of the *systems* applications within 25% of the actual value. The MRM ranked 1st in overall model performance for *systems* software:

- Actual Error – The mean error is 0.19 or an average of 19% from the actual value. The average error has a standard deviation of 97%. Thirty-nine projects, or 52%, are projected within 25% of the actual value. The MRM ranked 1 of 3 for average actual error.
- Absolute Error – The MRM projected the actual project performance with an absolute error of 68% ± 72%. The MRM ranked 2 of 3 for average absolute error.
- Under Estimate – The MRM projected 40 out of 75 projects under the actual value, 53%. When the MRM projects short, it does so with an average error of 46%. The majority (70%) of the under estimates occur

between zero and 50 percent. The MRM ranked 2 of 3 for balance and 2 of 3 for average under-estimation error.

- Over Estimation – The MRM over estimated 35 projects. Fifty-four percent of the over-estimates were between zero and 50 percent. The MRM ranked 2 of 3 for average over-estimation error.

### *b. Basic COCOMO Model*

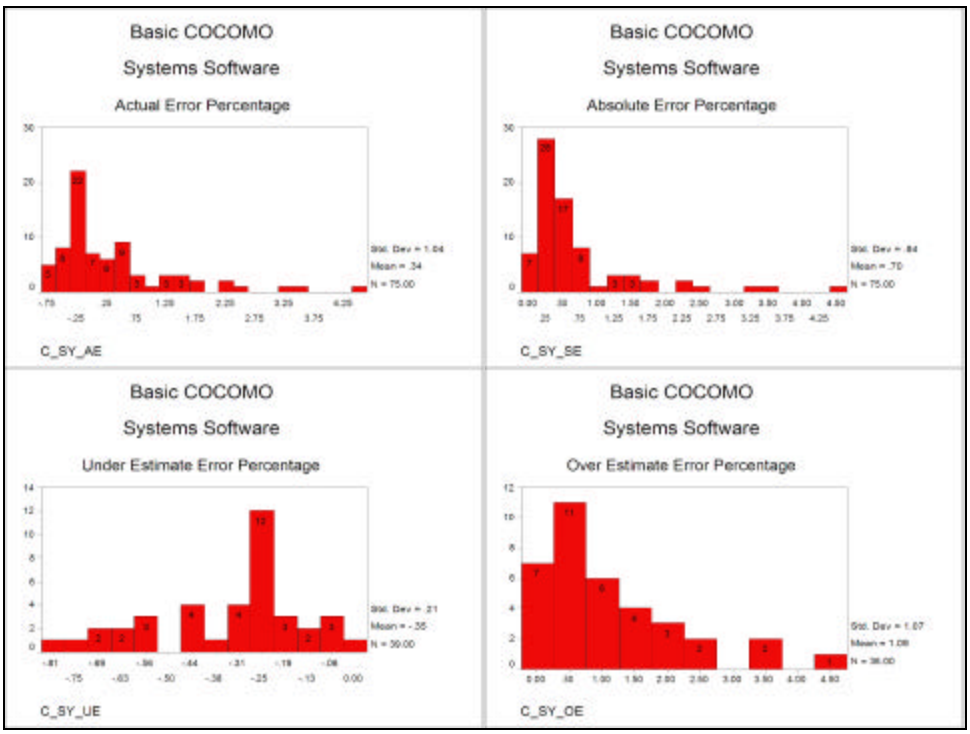



**Figure F-23. COCOMO Performance on Systems Software.**

The Basic COCOMO ranked 2nd in overall model performance for *systems* software:

- Actual Error – The mean error is 0.34 or an average of 34% from the actual value. The average error has a standard deviation of 104%. Thirty-five projects, or 47%, are projected within 25% of the actual value. The Basic COCOMO ranked 2 of 3 for average actual error.

- Absolute Error – The Basic COCOMO projected the actual project performance with an absolute error of 70% ± 84%. The Basic COCOMO ranked 3 of 3 for average absolute error.
- Under Estimate – The Basic COCOMO projected 39 out of 75 projects under the actual value, 52%. When the Basic COCOMO projects short, it does so with an average error of 35%. The majority (77%) of the under estimates occur between zero and 50 percent. The Basic COCOMO ranked 1 of 3 for balance and 1 of 3 for average under-estimation error.
- Over Estimation – The Basic COCOMO over estimated 36 projects. Fifty percent of the over-estimates were between zero and 50 percent. The Basic COCOMO ranked 3 of 3 for average over-estimation error.

### *c. Simplified Software Equation*

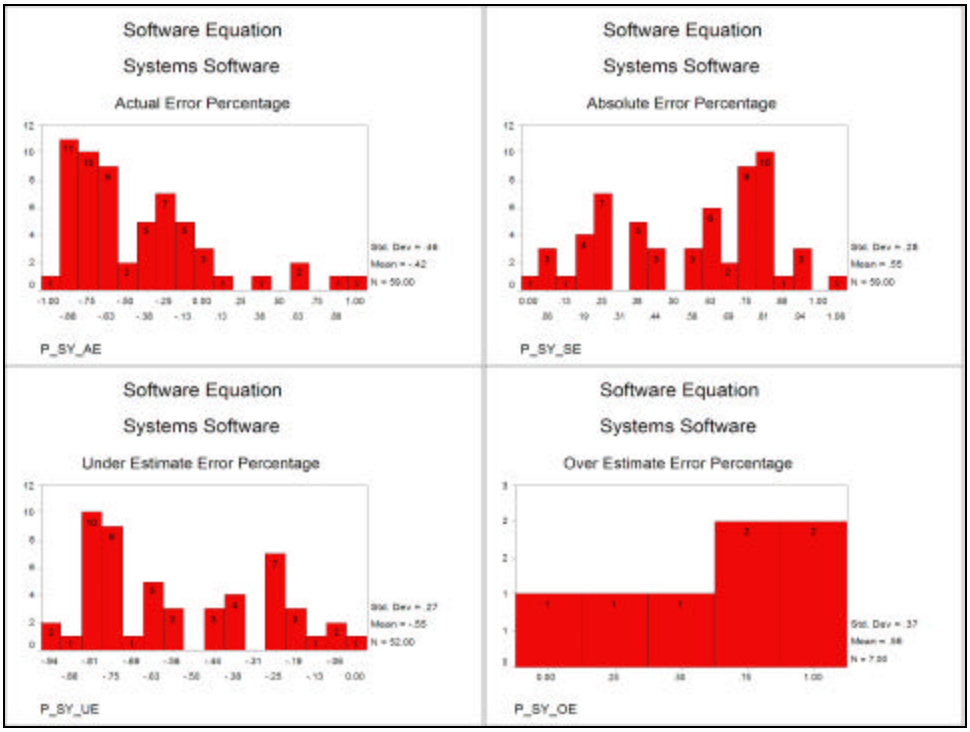


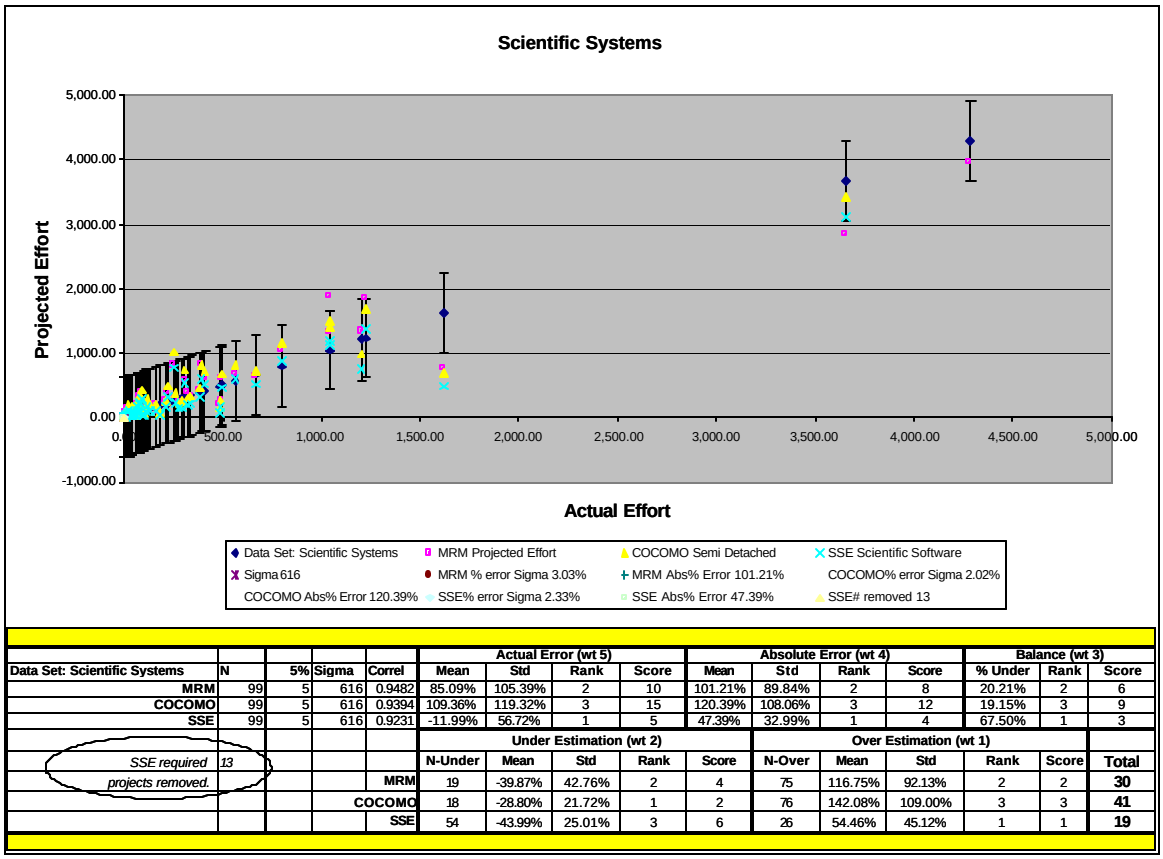

**Figure F-24. SSE Performance on Systems Software.**

The Simplified Software Equation ranked 3rd in overall model performance for *systems* software:

- Actual Error – The mean error is -0.42 or an average of 42% below the actual value. The average error has a standard deviation of 46%. Sixteen projects are projected within 25% of the actual value. The SSE ranked 3 of 3 for average actual error.
- Absolute Error – The SSE projected the actual project performance with an absolute error of 55% ± 28%. The SSE ranked 1 of 3 for average absolute error.
- Under Estimate – The SSE projected 52 of 59 projects under the actual value, 88%. When the SSE projects short, it does so with an average error of 40%. The SSE ranked 3 of 3 for balance and 3 of 3 for average under-estimation error.
- Over Estimation – The SSE over estimated 7 projects. Three projects were over-estimated within 50 percent. The SSE ranked 1 of 3 for average over-estimation error.

### 7. Scientific Systems

**Scientific.** Software that involves significant computations and analysis. Examples are statistical analysis systems, graphics products, data reduction systems. Figure F-25 is a scatter plot of the scientific systems in the project subset.




| | | | | | Actual Error (wt 5) | | | | Absolute Error (wt 4) | | | | Balance (wt 3) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data Set: Scientific Systems | N | 5% | Sigma | Correl | Mean | Std | Rank | Score | Mean | Std | Rank | Score | % Under | Rank | Score |
| MRM | 99 | 5 | 616 | 0.9482 | 85.09% | 105.39% | 2 | 10 | 101.21% | 89.84% | 2 | 8 | 20.21% | 2 | 6 |
| COCOMO | 99 | 5 | 616 | 0.9394 | 109.36% | 119.32% | 3 | 15 | 120.39% | 108.06% | 3 | 12 | 19.15% | 3 | 9 |
| SSE | 99 | 5 | 616 | 0.9231 | -11.99% | 56.72% | 1 | 5 | 47.39% | 32.99% | 1 | 4 | 67.50% | 1 | 3 |

| | | Under Estimation (wt 2) | | | | | Over Estimation (wt 1) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *SSE required 13 projects removed.* | | N-Under | Mean | Std | Rank | Score | N-Over | Mean | Std | Rank | Score | Total |
| | MRM | 19 | -39.87% | 42.76% | 2 | 4 | 75 | 116.75% | 92.13% | 2 | 2 | **30** |
| | COCOMO | 18 | -28.80% | 21.72% | 1 | 2 | 76 | 142.08% | 109.00% | 3 | 3 | **41** |
| | SSE | 54 | -43.99% | 25.01% | 3 | 6 | 26 | 54.46% | 45.12% | 1 | 1 | **19** |

**Figure F-25. Scientific Systems.**

Scientific Systems contained 99 projects. Each model had a total of five projects removed from the validation. The average standard deviation is 616 man-months. Each model achieved a correlation coefficient around 93%. A total of thirteen projects were removed from consideration for the Simplified Software Equation. Compared with the other two models, overall the Simplified Software Equation placed first followed by the MRM and the Basic COCOMO respectively.

### *a. Modified Risk Model*

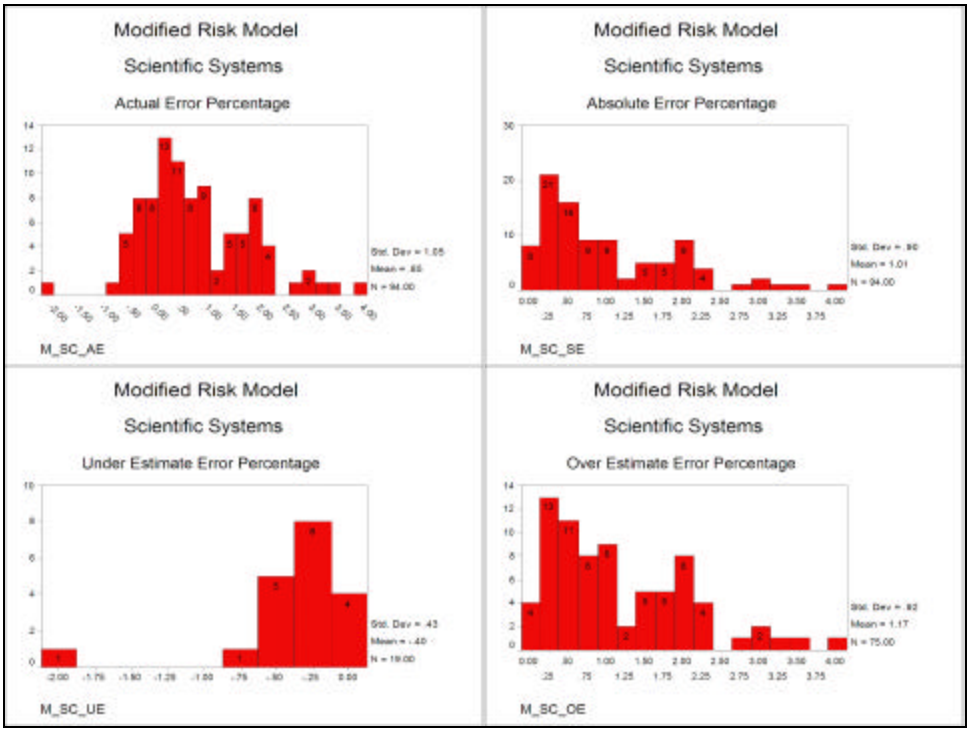



**Figure F-26. MRM Performance on Scientific Systems.**

The Modified Risk Model projected 19.15% of all of the *scientific* applications within 25% of the actual value. The MRM ranked 2nd in overall model performance for *scientific* systems:

- Actual Error – The mean error is 0.85 or an average of 85% from the actual value. The average error has a standard deviation of 105%. Twenty-eight projects, or 30%, are projected within 25% of the actual value. The MRM ranked 2 of 3 for average actual error.
- Absolute Error – The MRM projected the actual project performance with an absolute error of 101% ± 90%. The MRM ranked 2 of 3 for average absolute error.
- Under Estimate – The MRM projected 19 out of 94 projects under the actual value, 20%. When the MRM projects short, it does so with an average error of 40%. The majority (89%) of the under estimates occur between zero and 50 percent. The MRM ranked 2 of 3 for balance and 2 of 3 for average under-estimation error.

- Over Estimation – The MRM over estimated 75 projects. Thirty-seven percent of the over-estimates were between zero and 50 percent. The MRM ranked 2 of 3 for average over-estimation error.

### *b. Basic COCOMO Model*

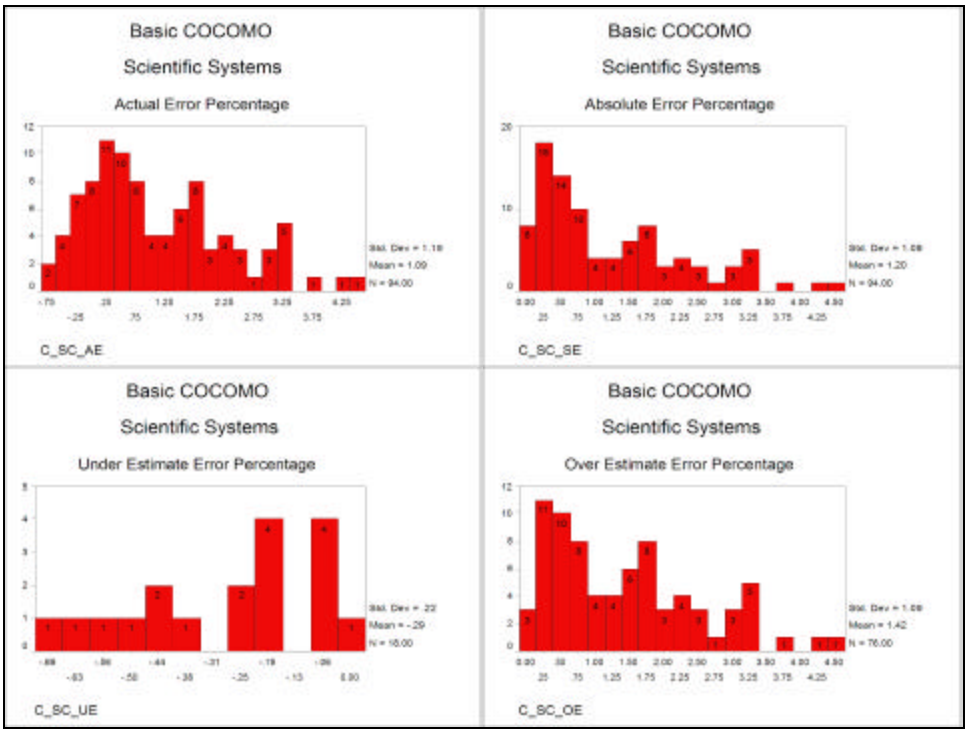



**Figure F-27. COCOMO Performance on Scientific Systems.**

The Basic COCOMO ranked 3rd in overall model performance for *scientific* systems:

- Actual Error – The mean error is 1.09 or an average of 109% from the actual value. The average error has a standard deviation of 119%. Twenty-four projects, or 26%, are projected within 25% of the actual value. The Basic COCOMO ranked 3 of 3 for average actual error.

- Absolute Error – The Basic COCOMO projected the actual project performance with an absolute error of 120% ± 108%. The Basic COCOMO ranked 3 of 3 for average absolute error.
- Under Estimate – The Basic COCOMO projected 18 out of 94 projects under the actual value, 19%. When the Basic COCOMO projects short, it does so with an average error of 29%. The majority (83%) of the under estimates occur between zero and 50 percent. The Basic COCOMO ranked 3 of 3 for balance and 1 of 3 for average under-estimation error.
- Over Estimation – The Basic COCOMO over estimated 76 projects. Thirty-two percent of the over-estimates were between zero and 50 percent. The Basic COCOMO ranked 3 of 3 for average over-estimation error.

### *c. Simplified Software Equation*

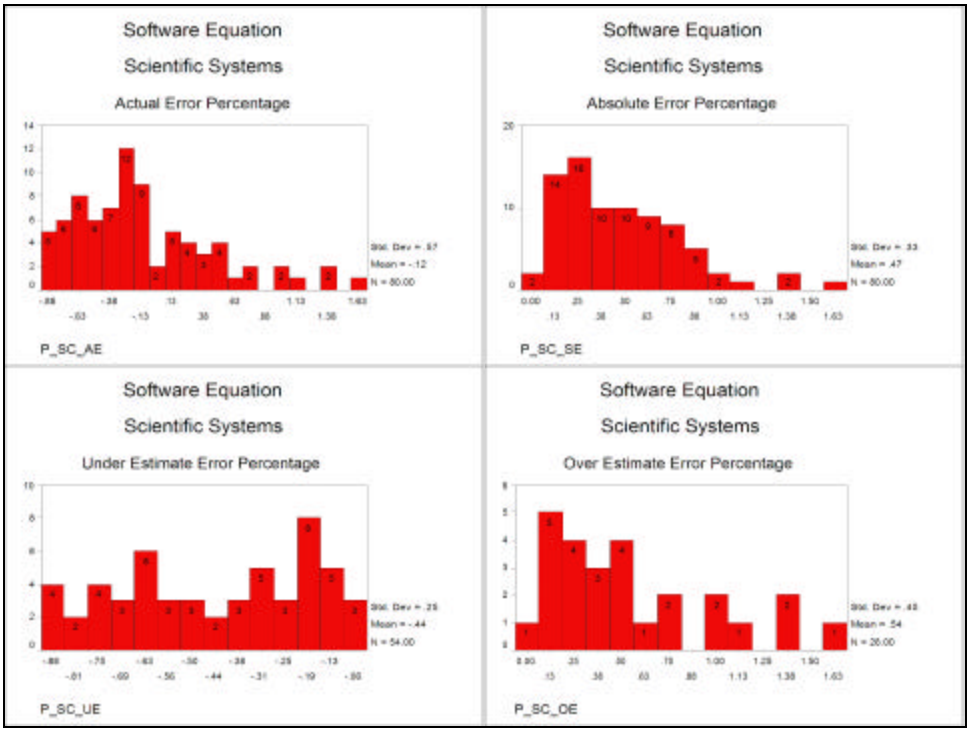


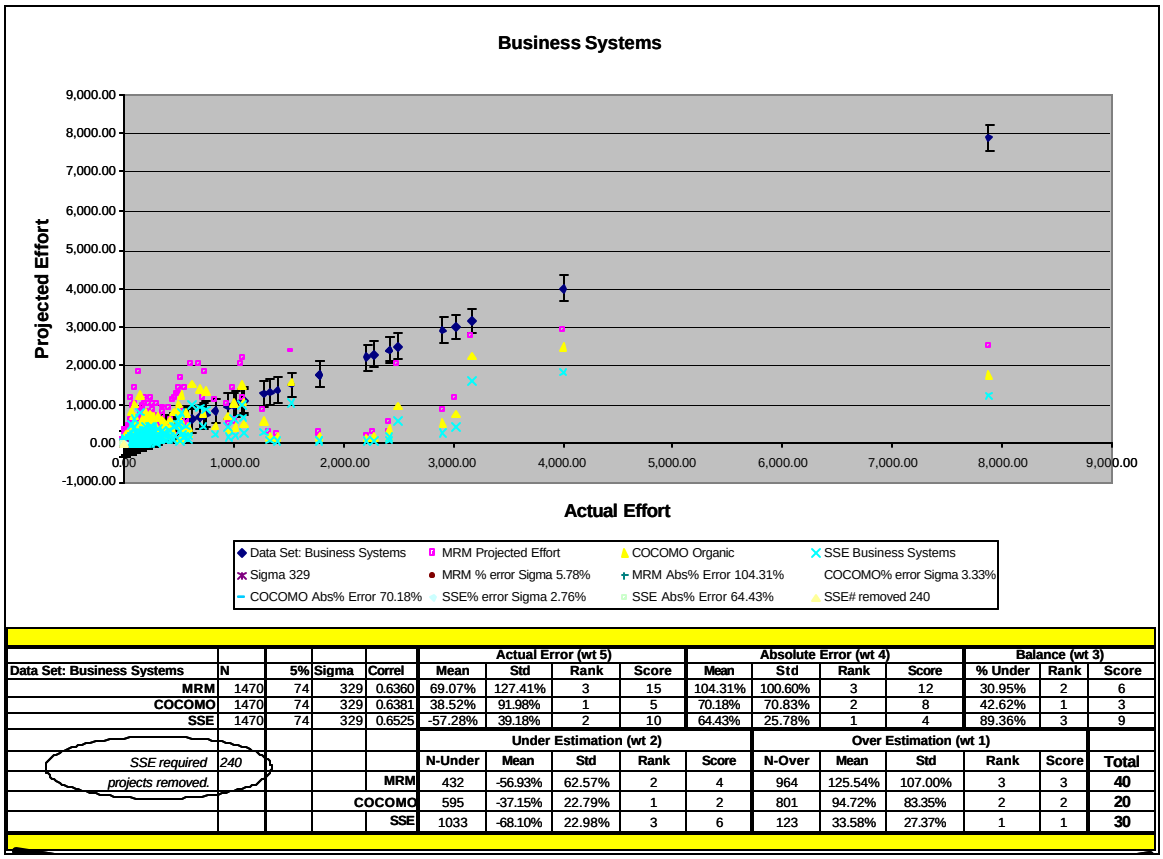

**Figure F-28. SSE Performance on Scientific Systems.**

The Simplified Software Equation ranked 1st in overall model performance for *scientific* systems:

- Actual Error – The mean error is -0.12 or an average of 12% below the actual value. The average error has a standard deviation of 57%. Thirty-two projects, 40%, are projected within 25% of the actual value. The SSE ranked 1 of 3 for average actual error.
- Absolute Error – The SSE projected the actual project performance with an absolute error of 47% ± 33%. The SSE ranked 1 of 3 for average absolute error.
- Under Estimate – The SSE projected 54 of 80 projects under the actual value, 68%. When the SSE projects short, it does so with an average error of 44%. The SSE ranked 1 of 3 for balance and 3 of 3 for average under-estimation error.
- Over Estimation – The SSE over estimated 26 projects. Seventeen projects were over-estimated within 50 percent. The SSE ranked 1 of 3 for average over-estimation error.

### 8. Business Systems

**Business.** Software that automates a common business function. Examples are payroll, personnel, order entry, inventory, materials handling, and warranty products. Figure F-29 is a scatter plot of the business systems in the project subset.




| | | | | | Actual Error (wt 5) | | | | Absolute Error (wt 4) | | | | Balance (wt 3) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data Set: Business Systems | N | 5% | Sigma | Correl | Mean | Std | Rank | Score | Mean | Std | Rank | Score | % Under | Rank | Score |
| MRM | 1470 | 74 | 329 | 0.6360 | 69.07% | 127.41% | 3 | 15 | 104.31% | 100.60% | 3 | 12 | 30.95% | 2 | 6 |
| COCOMO | 1470 | 74 | 329 | 0.6381 | 38.52% | 91.98% | 1 | 5 | 70.18% | 70.83% | 2 | 8 | 42.62% | 1 | 3 |
| SSE | 1470 | 74 | 329 | 0.6525 | -57.28% | 39.18% | 2 | 10 | 64.43% | 25.78% | 1 | 4 | 89.36% | 3 | 9 |

| | | Under Estimation (wt 2) | | | | | Over Estimation (wt 1) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SSE required 240 projects removed. | | N-Under | Mean | Std | Rank | Score | N-Over | Mean | Std | Rank | Score | Total |
| | MRM | 432 | -56.93% | 62.57% | 2 | 4 | 964 | 125.54% | 107.00% | 3 | 3 | 40 |
| | COCOMO | 595 | -37.15% | 22.79% | 1 | 2 | 801 | 94.72% | 83.35% | 2 | 2 | 20 |
| | SSE | 1033 | -68.10% | 22.98% | 3 | 6 | 123 | 33.58% | 27.37% | 1 | 1 | 30 |

**Figure F-29. Business Systems.**

Business Systems contained 1470 projects. Each model had a total of 74 projects removed from the validation. The average standard deviation is 329 man-months. Each model achieved a correlation coefficient around 64%. A total of 240 projects were removed from consideration for the Simplified Software Equation. Compared with the other two models, overall the Basic COCOMO placed first followed by the Simplified Software Equation and the MRM respectively.

### *a. Modified Risk Model*

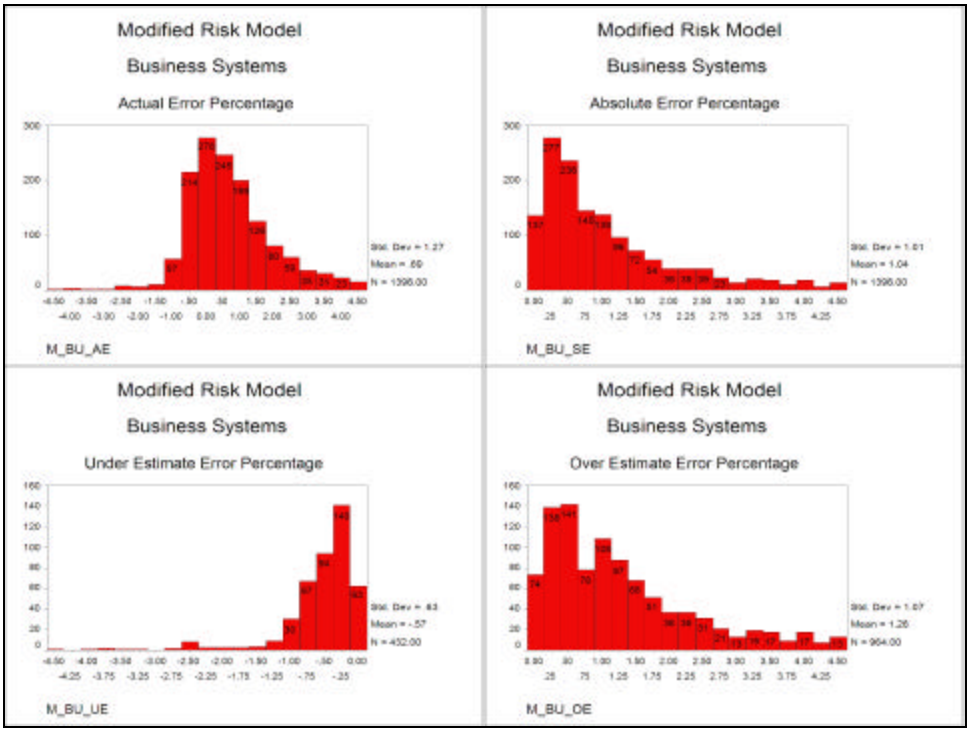



**Figure F-30. MRM Performance on Business Systems.**

The Modified Risk Model projected 19.13% of all of the *business* applications within 25% of the actual value. The MRM ranked 3rd in overall model performance for *business* systems:

- Actual Error – The mean error is 0.69 or an average of 69% from the actual value. The average error has a standard deviation of 127%. 408 projects, or 29%, are projected within 25% of the actual value. The MRM ranked 3 of 3 for average actual error.
- Absolute Error – The MRM projected the actual project performance with an absolute error of 104% ± 101%. The MRM ranked 3 of 3 for average absolute error.
- Under Estimate – The MRM projected 432 out of 1396 projects under the actual value, 31%. When the MRM projects short, it does so with an average error of 57%. The majority (69%) of the under estimates occur

between zero and 50 percent. The MRM ranked 2 of 3 for balance and 2 of 3 for average under-estimation error.

- Over Estimation – The MRM over estimated 964 projects. Thirty-six percent of the over-estimates were between zero and 50 percent. The MRM ranked 3 of 3 for average over-estimation error.

### *b. Basic COCOMO Model*

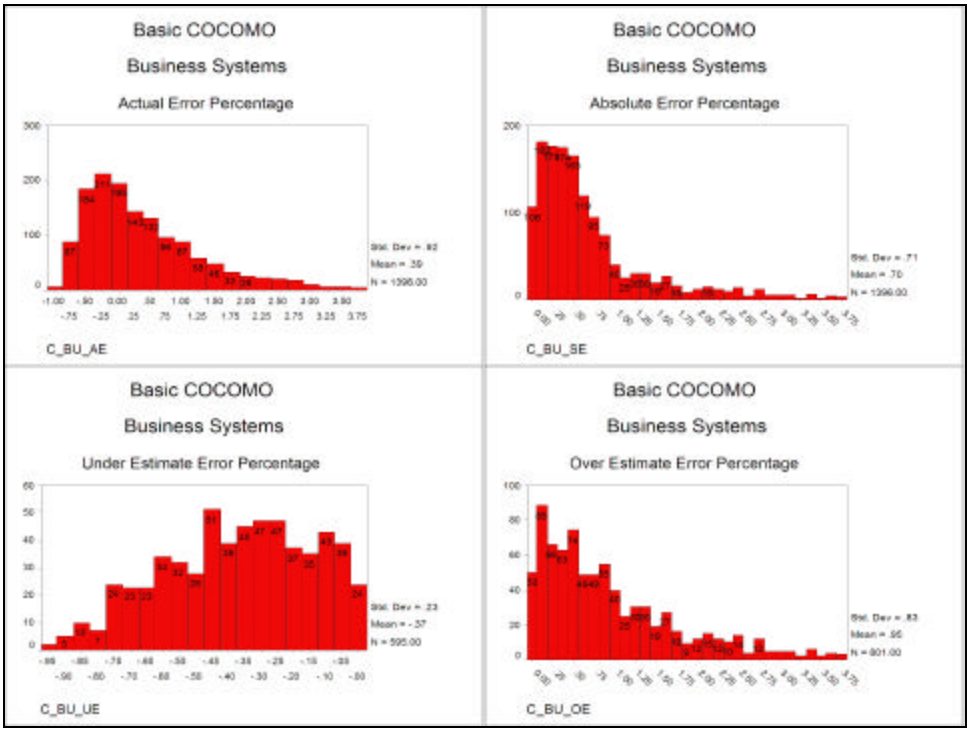



**Figure F-31. COCOMO Performance on Business Systems.**

The Basic COCOMO ranked 1st in overall model performance for *business* systems:

- Actual Error – The mean error is 0.39 or an average of 39% from the actual value. The average error has a standard deviation of 92%. 549 projects, or 39%, are projected within 25% of the actual value. The Basic COCOMO ranked 1 of 3 for average actual error.

- Absolute Error – The Basic COCOMO projected the actual project performance with an absolute error of 70% ± 71%. The Basic COCOMO ranked 2 of 3 for average absolute error.
- Under Estimate – The Basic COCOMO projected 595 out of 1396 projects under the actual value, 43%. When the Basic COCOMO projects short, it does so with an average error of 37%. The majority (73%) of the under estimates occur between zero and 50 percent. The Basic COCOMO ranked 1 of 3 for balance and 1 of 3 for average under-estimation error.
- Over Estimation – The Basic COCOMO over estimated 801 projects. Forty-three percent of the over-estimates were between zero and 50 percent. The Basic COCOMO ranked 2 of 3 for average over-estimation error.

### *c. Simplified Software Equation*

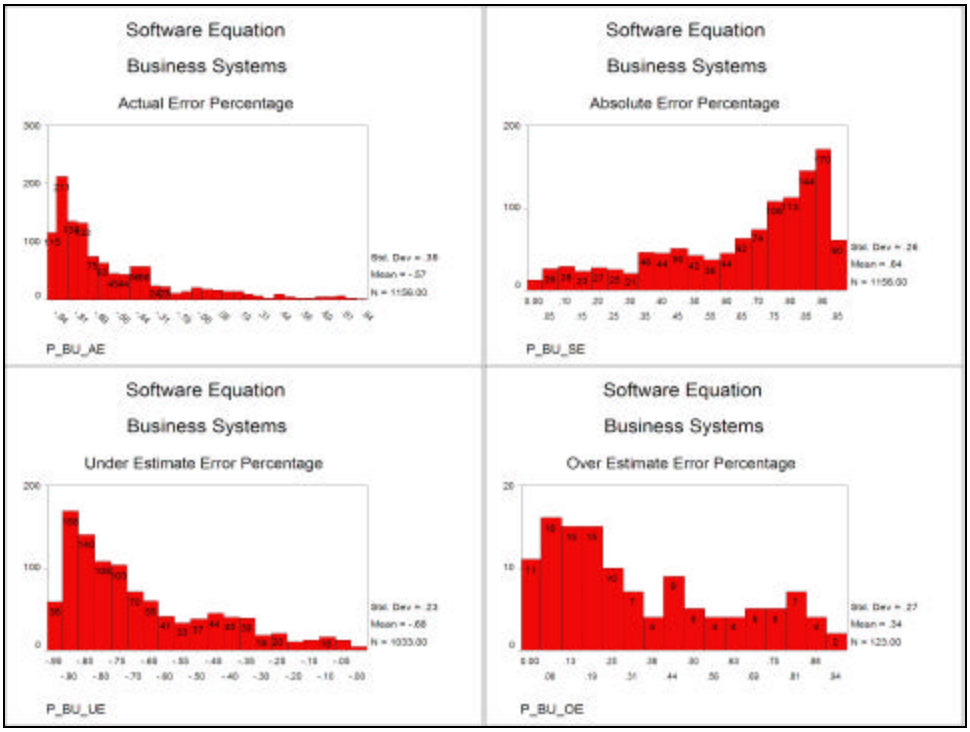



**Figure F-32. SSE Performance on Business Systems.**

The Simplified Software Equation ranked 2nd in overall model performance for *business* systems:

- Actual Error – The mean error is 0.57 or an average of 57% from the actual value. The average error has a standard deviation of 39%. 149 projects are projected within 25% of the actual value, 13%. The SSE ranked 2 of 3 for average actual error.
- Absolute Error – The SSE projected the actual project performance with an absolute error of 64% ± 26%. The SSE ranked 1 of 3 for average absolute error.
- Under Estimate – The SSE projected 1033 of 1156 projects under the actual value, 89%. When the SSE projects short, it does so with an average error of 68%. The SSE ranked 3 of 3 for balance and 3 of 3 for average under-estimation error.
- Over Estimation – The SSE over estimated 123 projects. Ninety-two projects were over-estimated within 50 percent. The SSE ranked 1 of 3 for average over-estimation error.

# LIST OF REFERENCES

(Albr79) Albrecht, A., Measuring Application Development Productivity. Proceedings IBM. October 1979.

(Albr83) Albrecht, A. and Gaffney, J., Software Function Source Lines of Code and Development Effort Prediction. IEEE Transactions on Software Engineering, SE-9, 1983.

(Balc94) Annals of Operations Research, Volume 53, 1994. pp. 121-173. Validation, Verification, and Testing Techniques throughout the Life Cycle of a Simulation Study.

(Balc98) Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice, edited by Jerry Banks. New York, John Wiley & Sons/ Engineering & Management Press, 1998. Chapter 10, pp. 335-393.

(Bark93) Barki, H., Rivard S., and Talbot J., *Toward an Assessment of Software Development Risk*. J. Management Information Systems, Vol. 10, No. 2, 1993, pp. 203-225.

(Boeh81) Boehm, B., *Software Engineering Economics.* Prentice Hall, 1981.

(Carr93) M. J. Carr et al., *Taxonomy-Based Risk Identification*, SEI-93-TR-006, Software Eng. Inst., Pittsburgh, 1993.

(Cont86) Conte. S, Dunsmore, H. and Shen, V., Software Engineering Metrics and Models. Benjamin Cummings. 1986.

(Crys93) Crystal Ball® version 3.0: User's Manual/Decisioneering, Inc. 1993.

(DoD5000.1) DoD Directive (DoDD) 5000.1, The Defense Acquisition System.

(DoD5000.2) DoD Instruction (DoDI) 5000.2, Operation of the Defense Acquisition System.

(D0D5000.2-R) DoD Regulation 5000.2-R (Interim), Mandatory Procedures for Major Defense Acquisition (MDAPs) and Major Automated Information System (MAIS) Acquisition Programs.

(DoD5000.4) DoD Directive 5000.4, OSD Cost Analysis Improvement Group.

(DoD5000.4-M) DoD Manual 5000.4-M, Cost Analysis Guidance and Procedures.

(DSMC01) DSMC Risk Management Guide for DoD Acquisition (Fourth Edition), February 2001.

(Dupo02) Dupont, Joseph, *Complexity Measure for the Prototype System Description Language (PSDL).* Master's Thesis. Naval Postgraduate School. Monterey, California. June 2002.

(Fent00) Fenton, N. E. and Neil, M., *Software Metrics: Roadmap.* Proceedings of the Conference on the Future of Software Engineering, 2000, pp. 357- 370.

(Garv97) Garvey, P. R., Phair, D. J., Wilson, J. A., *An Information Architecture For Risk Assessment And Management.* IEEE Software, Volume 14 Issue 3, May-June 1997, pp. 25–34.

(Hall97) Hall, E., Managing Risk. Methods for Software Systems Development. Addison Wesley, 1997.

(IEEE01) IEEE Std 1540-2001. Software Engineering Standards Committee of the IEEE Computer Society. Approved March 17, 2001.

(John01) Johnson, C. S., Piirainen R. A., *Application of the Nogueira Risk Assessment Model to Real-Time Embedded Software Projects.* Master's Thesis. Naval Postgraduate School. Monterey, California. March 2001.

(Jone94) Jones, Capers, Assessment and Control of Software Risks. Yourdon Press Prentice Hall, 1994.

(Karo96) Karolak, D., Software Engineering Management. IEEE Computer Society Press, 1996.

(Kesh00) Keshlaf, A. and Hashim, K., *A Model and Prototype Tool to Manage Software Risks.* Quality Software, 2000. Proceedings. First Asia-Pacific Conference on, 2000. pp. 297–305.

(Levi99) Levitt, R. *The ViteProject Handbook: A User's Guide to Modelling and Analyzing Project Work Processes and Organizations.* Vité ©. 1999.

(Lond87) Londeix, B., Cost Estimation for Software Development. Addison-Wesley, 1987.

(Luqi90) Berzins, V. and Luqi, *Software Engineering with Abstractions.* Addison-Wesley, 1990.

(Mose02) Moseman, Lloyd K., Keynote Speaker Address from Software Technology Conference, April 29, 2002.

(Moyn97) Moynihan, T., *How Experienced Project Managers Assess Risk.* IEEE Software, Volume: 14 Issue: 3, May-June 1997. pp. 35–41.

(MSEX02) Microsoft® Excel 2002 (10.4302.3219) SP-2, Copyright© Microsoft Corporation 1985-2001.

(Nogu00) Nogueira J. C., *A Formal Model for Risk Assessment in Software Projects.* PhD Dissertation. Naval Postgraduate School. Monterey, California, 2000.

(PEH99) Parametric Estimating Handbook, Spring 1999, Department of Defense.

(Pres01) Pressman, Roger S., Software Engineering: a Practitioner's Approach, 5th ed., McGraw-Hill series in Computer Science, 2001.

(Putn80) Putnam, L., *Software Cost Estimating and Life-Cycle Control: Getting the Software Numbers.* IEEE Computer Society Press. 1980.

(Putn92) Putnam, L. and Myers, W., *Measures for Excellence. Reliable Software On Time Within Budget.* Yourdon Press, 1992.

(Putn96) Putnam, L. and Myers, W., *Executive Briefing. Controlling Software Development*. IEEE Computer Society Press. 1996.

(QSMa) Quantitative Software Management®, *User's Guide to SLIM-Metrics 5.0.* QSM, Inc., McLean, Virginia, USA.

(QSMb) Quantitative Software Management®, *User's Guide to SLIM-Estimate 5.0.* QSM, Inc. McLean, Virginia USA.

(QSMc) Quantitative Software Management®, *Software Lifecycle Management (SLIM) Training Version 2.02.* QSM, Inc. McLean, Virginia, USA.

(Reel99) Reel, J. Critical Success Factors in Software Projects, IEEE Software. May - June, 1999.

(Sabo97) Mike Saboe, "Associate Director, Next Generation Software Engineering", U.S. Army Tank Automotive Research and Development Command, Memo, 1997.

(SEI96) Software Engineering Institute. Software Risk Management, Technical Report CMU/SEI-96-TR-012. June, 1996.

(SEI97) Software Engineering Institure. Software Technology Review, Function Point Analysis. Edmond VanDoren, Kaman Sciences, Colorado Springs, 1997.

(Shan75) Shannon, R.E., System Simulation: The Art and Science (Prentice-Hall, Englewood Cliffs, NJ, 1975.

(Stut96) Stutzke, Richard D., Software Estimating Technology: A Survey. CrossTalk, May 1996, pp.17-22.

(SPSS99) SPSS® Base 9.0, Applications Guide. Copyright© 1999 by SPSS Inc.

(Thay81) Thayer, Richard H., Pyster, Arthur B., Wood, Roger C., Major Issues in Software Engineering Project Management. TSE 7(4): 333-342 (1981)

(vanG91) van Genuchten, M., Why is Software Late? An Empirical Study of the Reasons for Delay in Software Development. IEEE Transactions on Software Engineering. June, 1991.

(Zues97) Zuse, Horst, *A Framework of Software Measurement*, Walter de Gruyter & Co., New York, New York, 1997.

# BIBLIOGRAPHY


Abdel-Hamid, T., Lessons learned from modeling the dynamics of Software. Communications of the ACM. December, 1989.

Abdel-Hamid, T., Software Project Dynamics: An Integrated Approach. Prentice Hall, 1991.

Agresti and Evanco, Projecting Software Defects from Analyzing Ada Designs. IEEE Transactions on Software Engineering, Vol. 18, No. 11, November 1992, pp. 988-997.

American Institute of Aeronautic and Astronautics. Recommended Practice for Software Reliability, ANSI/AIAA R-013-1992, February 1993.

ANSI/IEEE Standard Glossary of Software Engineering Terminology, STD-729-1991.

Badr, S., A Model and Algorithms for a Software Evolution Control System. PhD Dissertation, Computer Science Department. Naval Postgraduate School. Monterey, CA. 1993.

Baligh, H., Burton, R., and Obel, B., Validating an Expert System that Designs Organizations. In Computational Organization Theory edited by Carley, K. and Prietula, M. Lawrence Erlbaum Associates, Publishers. 1994.

Baligh, H., Burton, R., and Obel, B., Organizational Consultant: Creating a Useable Theory for Organizational Design. Management Science, 42(12). 1996.

Baybutt, P., Uncertainty in Risk Analysis. Mathematics in Major Accidents Risk Analysis. Edited by R.A. Cox. Clarendon Press - Oxford, 1989.

Beck, K., Extreme Programming Explained. Embrace Change. Addison-Wesley, 1999.

Berzins, V. and Luqi, Software Engineering with Abstractions. Addison-Wesley, 1990.

Boehm, B., Verifying and Validating Software Requirements and Design Specifications. IEEE Software, January 1984.

Boehm, B., A Spiral Model of Software Development and Enhancement. Computer. May, 1988.

Boehm, B. and Belz, F., Applying Process Programming to the Spiral Model. Proceedings of the 4th International Software Process Workshop. May 1988.

Boehm, B., Software Risk Management. IEEE Computer Society Press. 1989.

Boehm, B., Software Risk Management: Principles and Practices. IEEE Software, January, 1991.

Boehm, B. and De Marco T., Software Risk Management. IEEE Software. May-June, 1997.

Boehm, B., Madachy R., Selby, R. Cost Models for Future Software Life Cycle Processes: COCOMO 2.0. http://sunset.usc.edu/COCOMOII/cocomo.html

Boehm, B. et al., Software Cost Estimation with COCOMO II. Prentice Hall, 2000.

Brooks, F., The Mythical Man-Month. Datamation. December. 1974.

Brown, S. and Eisenhardt, K., Competing on the Edge. Strategy as Structured Chaos. Harvard Business School Press. 1998.

Burton, R., and Obel, B., Strategic Organizational Diagnosis and Design. Developing Theory for Application. Kluwer Academic Publishers. 1998.

Campbell, D. and Stanley, J., Experimental and Quasi-experimental Designs for Research. Rand McNally, 1966

Charette, R., Adams, K., and White, M., Managing Risk in Software Maintenance. IEEE Software, May-June, 1997.

Chen, E., Program Complexity and Programmer Productivity. IEEE Trans. Soft. Eng. May 1978.

Christiansen, T. R., Modeling the Efficiency and Effectiveness of Coordination in Engineering Design Teams. Ph.D. Dissertation, Department of Civil Engineering, Stanford University. Published as Det Norske Veritas Research Report No. 93-2063, Oslo, Norway.

Chulani, Sunita and Boehm, Steece, Bayesian Analysis of Empirical Software Engineering Cost Models. IEEE Transactions on Software Engineering. July-August, 1999.

Cohen G.P., The Virtual Design Team: An Object-Oriented Model of Information Sharing in Project Teams [Ph.D.]. Stanford: Stanford University, 1992.

Conklin, J. and Begeman, M., GIBIS: A Hypertext Tool for Exploratory Policy Discussion. ACM Transactions on Office Information Systems. Vol. 6. October, 1988.

Cook, T. and Campbell, D., The Design and Conduct of Quasi-Experiments and True Experiments in Field Settings. In Handbook of Industrial and Organizational Psychology. Rand-McNally. Dunnette (editor). 1976.

Cullen, A. and Frey, H., Probabilistic Techniques in Exposure Assessment. A Handbook for Dealing with Variability and Uncertainty in Models and Inputs. Plenum Press. 1999.

Cusumano, M. and Yoffie, D., Software Development on Internet Time. Computer. October, 1999.

Daft, R., Organization Theory and Design. West Publishing Co., 1989.

Dalkey, N. and Helmer, O., An Experimental Application of the Delphi Method to the Use of Experts. Management Science, 1963, 9, 458-467.

Devore, J., Probability and Statistics for Engineering and the Sciences. Duxbury. 1995.

Dooley, K. and Flor, R., Success and Failure in Total Quality Management Initiatives. Proceeding of the Chaos Network, Denver, 1994.

Elsayed, E., Reliability Engineering. Addison Wesley. 1996.

Field, T., When BAD Things Happen to GOOD Projects. CIO, 15 October, 1997.

Gaffney and Davis., An Approach to Estimating Software Errors and Availability. SPC-TR-88-007 version 1.0, March 1988. Proceedings of the Workshop on Software Reliability, July 1988.

Galbraith, J. R., Organization Design. Reading, MA: Addison-Wesley, 1977.

Gemmer., Risk Management: Moving Beyond Process. Computer Vol. 30, Issue 5, May, 1997.

Gilb, T., Software Metrics. Winthrop Publishers, Inc. 1977.

Gilb, T., Principles of Software Engineering Management. Addison-Wesley 1988.

Goel, A. and Okumoto, K., Time-Dependent Error-Detection Rate Model for Software and Other Performance Measures. IEEE Transactions on Software Reliability. August, 1979.

Harn, M., Berzins, V. and Luqi, Software Evolution via Reusable Architecture. Proceedings of 1999 IEEE Conference and Workshop on Engineering of Computer-Based Systems. Nashville, TN. March, 1999.

Harn, M., Computer-Aided Software Evolution Based on Inferred Dependencies. Proceedings of Conference on Advanced Information Systems Engineering: 6th Doctoral Consortium. Heidelberg, Germany. June, 1999.

Harn, M., Berzins, V. and Luqi, A Dependency Computing Model for Software Evolution. Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering. Kaiserslautern, Germany. June, 1999.

Harn, M., Berzins, V. and Luqi, Computer-Aided Software Evolution Based on a Formal Model. Proceedings of the 13th International Conference on Systems Engineering. Las Vegas, NV. August, 1999.

Harn, M., Relational Hypergraph Model. PhD Disssertation. Naval Postgraduate School. Monterey, CA. 1999.

Huberman, B. and Glance, N., Fluctuating Efforts and Sustainable Cooperation. Chapter 5 on Prietula, M., Carley, K., Gasser L. Simulating Organizations. Computational Models for Institutions and Groups. MIT Press, 1998.

Humphrey, W. et al., A Method for Assessing the Software Capability of Contractors. CMU/SEI-87-TR-23. 1987.

Humphrey, W., Managing the Software Process. Addison-Wesley, 1989.

Ibrahim, O., A Model and Decision Support Mechanism for Software Requirements Engineering. Ph.D. Dissertation. Naval Postgraduate School. Monterey, CA. 1996.

James, G. E., Chaos Theory. The Essentials for Military Applications. Naval War College. The Newport Papers, 1996.

Johnson, N. and Kotz, S., and Balakrishnan N. Continuous Univariate Distributions. Vol. 1. Wiley & Sons, 1994.

Jones, Capers, By Popular Demand: Software Estimating Rules of Thumb. Computer, March 1996.

Jones, Capers, Table of Languages. 1997.[http://www.spr.com/library/0langtabl.htm]

Jin, Y. and Levitt, R., (Department of Civil Engineering, Stanford University). The Virtual Design Team: A Computational Model of Project Organizations. Paper to appear in Computational and Mathemetical Organization Theory. 1996.

Kang, M., Waisel, L. and Wallace, W., Team Soar. A Model for Team Decision Making. Chapter 2 on Prietula, M., Carley, K., Gasser L. Simulating Organizations. Computational Models for Institutions and Groups. MIT Press, 1998.

Kauffman, Stuart, At Home in the Universe. Oxford University Press, 1995.

Kemerer, C., Reliability of Function Points Measurements: A Field Experiment. Communications of ACM, Vol. 36, No. 2. 1993.

Kemerer, C., Software Project Management. Readings and Cases. McGraw-Hill. 1997.

Kitchenham, B. and Kansala, K., Inter-Item Correlations among Function Points. First International Software Metrics Symposium. IEEE Computer Society Press. 1993.

Kitchenham, B. and Linkman, S., Estimates, Uncertainty, and Risk. IEEE Software. May-June, 1997.

Kunz, J. C., Christiansen, Tore R., Cohen, Geoff P., Jin, Yan, and Levitt, Raymond E., The Virtual Design Team: A Computational Simulation Model of Project Organizations. Communications of the Association for Computing Machinery (CACM) 41 (11), November, 1998, pp. 84-91.

Levitt, R. E., Cohen, G. P., Kunz, J. C., Nass, C. I., Christiansen, T., and Jin, Y., (1994), The Virtual Design Team: Simulating How Organization Structures and Information Processing Tools Affect Team Performance, in K. Carley and M. Prietula (Eds.) Computational Organizational Theory, Hillsdale, NJ: Lawerence Erlbaum Associates.

Levitt, R., VDT Computational Emulation Models of Organizations: State of the Art and the Practice. Center for Integrated Facility Engineering. Stanford University, 2000.

Lin, Z., The Choice Between Accuracy and Errors. A Contingency Analysis of External Conditions and Organizational Decision Making Performance. Chapter 4 on Prietula, M., Carley, K., Gasser L. Simulating Organizations. Computational Models for Institutions and Groups. MIT Press, 1998.

Levy, H., Stochastic Dominance. Investment Decision Making under Uncertainty. Kluwer Academic Publishers. 1998.

Lorenz, Kidd, OO Software Metrics. Prentice Hall, 1995.

Luqi and Ketabchi, M. A., Computer-Aided Prototyping System. IEEE Software. March, 1988.

Luqi and Berzins, V., Rapidly Prototyping Real-Time Systems. IEEE Software. September, 1988.

Luqi, Software Evolution Through Rapid Prototyping. IEEE Computer. May, 1989.

Luqi, A Graph Model for Software Evolution. IEEE Transactions on Software Engineering. Vol. 16, No. 8. August, 1990.

Luqi, Formal Models and Prototyping. Research supported by National Science Foundation (CCR-9058453) and by the Army research Office (30989-MA).

Luqi and Royce, W., Status Report: Computer-Aided Prototyping. IEEE Software. November, 1991.

Luqi and Goguen, J., Formal Methods: Promises and Problems. IEEE Software. January, 1997.

Lyu, M., Software Reliability Engineering. IEEE Computer Society Press. 1995.

McFarlan, F., Portfolio Approach to Information Systems. Harvard Business Review. January-February, 1974.

Marshall, K. and Oliver, R., Decision Making and Forecasting. McGraww-Hill, 1995.

Mostov, Luqi and Hefner, A Graph Model of Software Maintenance. Technical Report NPS52-90-014. Department of Computer Science. Naval Postgraduate School. Monterey, CA. August 1989.

Mostov, A Model of Software Maintenance for Large Scale Military Systems. Master's Thesis. Naval Postgraduate School. Monterey, CA. June, 1990.

Munson, J. and Khoshgofar, T., Chapter 12 (Lyu, 1995).

Musa, J., Software Reliability Engineering: More Reliable Software, Faster Development and Testing. McGraw-Hill, 1998.

Myers, G., Software Reliability. John Wiley & Sons. 1976.

Nissen, M., Redesigning Reengineering through Measurement-Driven Inference. MIS Quarterly. December, 1998.

Nogueira, J. C., Luqi, and Berzins, V. A., Formal Risk Assessment Model for Software Evolution. SEKE 2000. Chicago, July 2000.

Nogueira, J. C., Luqi, and Bhattacharya, S., A Risk Assessment Model for Software Prototyping Projects. IEEE Workshop on Rapid System Prototyping RSP 2000. Paris, June 2000.

Nogueira, J. C., Luqi, , Berzins, V., and Nada, N., A Formal Risk Assessment Model for Software Evolution. ICSE 2000. Limerick, June 2000.

Nogueira, J. C., Luqi, and Berzins, V., Risk Assessment in Software Requirement Engineering. IDTP 2000. Dallas, June 2000.

Nogueira, J.C., Jones, C. R., and Luqi, Surfing on the Edge of Chaos: Applications to Software Engineering. CCRP 2000. Monterey, June 2000.

Norden, Peter, Resource Usage and Network Planning Techniques. In Operations Research in Research and Development. Edited by Dean, B. John Wiley & Sons 1963 pp. 149-169.

O'Leary, D., Methods of Validating Experts Systems. Interfaces #18. 1988.

Pearl, J., Causality. Models, Reasoning and Inferrence. Cambridge University Press, 2000.

Porter, Michael, Competitive Strategy. Free Press, 1980.

Pfleeger, S .L,. Albert Einstein and Empirical Software Engineering. IEEE Computer. October 1999.

Prietula, M., Carley, K., and Gasser L., Simulating Organizations. Computational Models for Institutions and Groups. MIT Press, 1998.

Putnam, L. and Myers, W., Industrial Strength Software. Effective Management Using Measurement. IEEE Computer Society Press, 1997.

Putnam, L., Linking the QSM® Productivity Index with the SEI Maturity Level, Quantitative Software Management, Inc, 2000.

Ramesh, B. and Dhar, V., Supporting Systems Development Using Knowledge Captured During Requirements Engineering. IEEE Transactions on Software Engineering. June, 1992.

Ramesh and Luqi, An Intelligent Assistant for Requirements Validation. Journal of Systems Integration, 5, 157-177. 1995.

Render, B. and Stair, R., Quantitative Analysis for Management. Prentice Hall, 1997.

Rifkin, S., When the Project Absolutely Must Get Done: Marrying the Organization Chart with the Precedence Diagram. International Conference on Software Engineering (ICSE 2000), Limerick, Ireland, June 2000.

Roberts, N., Coping with Wicked Problems. Third Bi-Annual Research Conference of the International Public Management Network. Sydney, March 2000.

Rome Laboratory, Methodology for Software Reliability Prediction and Assessment. Technical Report RL-TR-92-52. 1992.

Roos, Johan, The Poised Organization: Navigating Effectively on Knowledge Landscapes., 1996. [http://www.imd.ch/fac/roos/paper_po.html]

Santosus, Megan, Simple, Yet Complex. Business Management CIO Enterprise Magazine. April 15, 1998.

Schneidewind, N., Analysis of Error Processes in Computer Software. Proceedings of the International Conference on Reliable Software. IEEE Computer Society, 21-23 April 1975. pp. 337-346.

Sengupta, K. and Jones, Carl R., Creating Structures for Network-Centric Warefare: Perspectives from Organizational Theory. Command & Control Research & Technology Symposium. CCRP 1999. Naval War College, 1999.

Sommerville, I., Software Engineering. 1992.

Thomsen, Jan, Levitt, Raymond E., Kunz, John C., Nass, Clifford I., and Fridsma, Douglas B., A Trajectory for Validating Computational Emulation Models of Organizations. Journal of Computational & Mathematical Organization Theory, 5, (4), December 1999, pp. 385-401.

Turban, E. and Aronson, J., Decision Support Systems and Intelligent Systems. Prentice Hall. 1998.

USAF, Software Risk Abatement. ASFC/AFLC pamphlet 800-45, US Air Force Systems Command. Andrews AFB. 1988.

University of South California, The Win Win Spiral Model and Groupware Support System
[http://sunset.esc.edu/research/WINWIN/winwin_main.html 2000]

von Bertalanfy, L., General System Theory: Foundations, Development, Applications. Braziller, 1976.

Walston, C. and Felix, C., A Method of Programming Measurement and Estimation. IBM Systems Journal. Vol. 16 No. 1, 1977.

Weibull++ Ver 5.0. Reliasoft. [http://www.reliasoft.com]

Weibull, W., A Statistical Theory of the Strength of Material. Report No. 151, Ingeniors Vetenskaps Akademiens Handligar. Stockholm, 1939.

Whitmore, G. and Findlay, M., Stochastic Dominance. Lexington Books. 1978.

Wideman, R., Risk Management. A Guide to Managing Project Risk Opportunities. Project Management Institute. 1992.

Woodward, J., Industrial Organization Theory and Practice. Oxford University Press. 1965.

Woodward, S., Evolutionary Project Management. IEEE Computer, October 1999.

THIS PAGE INTENTIONALLY LEFT BLANK

# INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor Luqi
   Naval Postgraduate School
   Code CS/Lq
   Monterey, California

4. Professor Berzins
   Naval Postgraduate School
   Code CS/Bz
   Monterey, California

5. Professor Dan Dolk
   Naval Postgraduate School
   Monterey, California

6. Professor Nabendu Chaki
   Naval Postgraduate School
   Code CS/Ch
   Monterey, California

7. Mr. Lawrence Putnam
   Quantitative Software Management
   McLean, Virginia

8. Major Michael R. Murrah
   Army Research Lab
   AMSRL-CS-PE-MP
   Adelphi, Maryland