CALIBRATION OF THE
SOFTWARE LIFE CYCLE MODEL (SLIM)
TO THE SPACE AND MISSILE SYSTEMS CENTER (SMC)
SOFTWARE DATABASE (SWDB)

THESIS

Robert K. Kressin, B.A.
Captain, USAF

AFIT/GCA/LAS/95S-6

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCA/LAS/95S-6

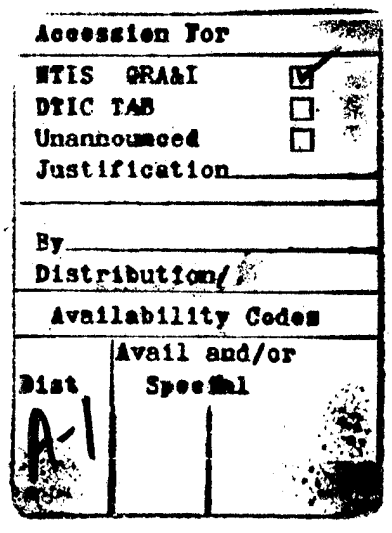
CALIBRATION OF THE SOFTWARE LIFE CYCLE MODEL (SLIM) TO THE SPACE AND MISSILE SYSTEMS CENTER (SMC) SOFTWARE DATABASE (SWDB)

THESIS

Robert K. Kressin, B.A.
Captain, USAF

AFIT/GCA/LAS/95S-6

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.



AFIT/GCA/LAS/95S-6

CALIBRATION OF THE SOFTWARE LIFE CYCLE MODEL (SLIM) TO THE SPACE AND MISSILE SYSTEMS CENTER (SMC) SOFTWARE DATABASE (SWDB)

THESIS

Presented to the Faculty of the Graduate School of Logistics

and Acquisition Management of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Cost Analysis

Robert K. Kressin, B.A.

Captain, USAF

September 1995

Acknowledgements

As I reflect upon these past 15 months at the Air Force Institute of Technology, the challenges presented were vast, however the knowledge gained is invaluable. The accomplishments that I have been able to achieve would have been infinitely more difficult if it were not for the support, guidance and sincere commitment of the following individuals.

I am deeply indebted to my thesis advisor, Professor Daniel Ferens, for his guidance and patience in this research effort. He provided me the necessary focus to keep this research on the right track. I am also thankful to Dr. David S. Christensen for being my thesis reader, and giving me valuable feedback. Additionally, I am thankful to Sherry Stukes of Management Consulting and Research and Gina Novak-Ley of the Space and Missile Systems Center for contributing their knowledge, experience, and time to this effort.

The constant love and support of my wife Annette and five wonderful children make everything worthwhile. A very special thank you to my oldest daughter, Tiffany for her understanding and being such a big help when Dad was too busy! And last but not least, to my cat Shizza, who slept on my lap many nights while I burned the midnight oil, and my dogs, who checked on me frequently while I worked. Annette, Tiff, Darren, Michelle, Bobby and Allie, thank you and I love you.

Robert K. Kressin

Table of Contents

Page

List of Figures

List of Tables

List of Equations

Abstract

Several sophisticated Department of Defense (DOD) weapon and space systems depend on a variety of mission-critical computer software applications. Because of the increasing costs to develop these applications, and today's austere defense budget, accurately estimating the costs to develop software is gaining a great deal of attention within the DOD. Unfortunately, software cost estimation models have been encountering difficulties in accurately estimating the costs of software development. However, research has shown that calibrating one's software cost estimation model can improve the predictive capability of that model. As such, this study focused on calibrating the SLIM cost model to sets of historic software projects that were described along a specific development environment, and assessing the predictive ability of the model resulting from this calibration.

The SMC database was first stratified along specific operating environment and application parameters, as well as some key SLIM input parameters (size, effort, development schedule, and minimum peak staffing). Stratification of the SMC database resulted in three data subsets, two of which were defined along a specific operating environment and application type, and one which was only defined along a specific operating environment. Once SLIM was calibrated to each data set, the calibrated SLIM models were validated in an effort to verify their predictive ability. Validation was accomplished using MRE, MMRE, percentage method, and standard deviation measures.

The Wilcoxon signed-rank test was also accomplished to test for randomness of the estimates from the calibrated models.

The author concluded that successfully calibrating SLIM depends largely on the user's ability to calibrate a consistent PI and MBI across a set of historic software projects. PI and MBI consistency, as well as model predictive ability, was best for data sets that were stratified along a specified application type.

CALIBRATION OF THE SOFTWARE LIFE CYCLE MODEL (SLIM) TO THE SPACE AND MISSILE SYSTEMS CENTER (SMC) SOFTWARE DATABASE (SWDB)

I. Introduction

Overview

Perhaps one of the most difficult tasks cost analysts face today is that of estimating the cost of software development. This chapter introduces problems associated with software cost estimation. In particular, this chapter examines the general need for accurate software cost estimates, presents the specific problem faced by the Space and Missile Systems Center (SMC) and their need to calibrate the Software Life Cycle Model (SLIM) to their Software Database (SWDB), addresses the importance of calibrating software cost estimating models to specific environments, and presents the objectives and scope of this research. Lastly, an overview of the structure of this thesis is presented. Also, for the reader's convenience, a glossary is provided at the end of the thesis.

General Issue

A company can further its competitive edge in today's global economy by capitalizing on the latest computer technology and software applications. However, "estimating the cost of a software project is one of the most difficult and error-prone tasks

in software engineering" (9:64). This, and a company's push to be on the leading edge of software use, necessitates the need for decision-makers to have access to accurately estimated costs of software development projects.

The Department of Defense (DOD) depends on a variety of mission-critical computer software applications to support sophisticated weapon and space systems. However, as pointed out by Coggins and Russell, in their study of software cost estimating models, "a recent study of DOD mission-critical software costs predicts a 12 percent annual growth rate from $11.4 billion in 1985 to $36 billion in 1995" (4:2). Interestingly, approximately fifteen percent of all software programs fail to deliver anything; they don't achieve their goal (7:3). Additionally, "overruns of one hundred to two hundred percent are common in software projects" (7:3). Furthermore, "a General Accounting Office study pointed out that more than 50 percent of the software systems studied had significant cost overruns; more than 60 percent had serious schedule slippages" (22:9). Because of proliferating costs in the area of software development, and the fact that software projects are experiencing significant cost overruns, and the difficulty in estimating software costs, software cost estimating models that can accurately capture the development and maintenance costs of a software project are becoming increasingly important to DOD.

There are several software cost estimating techniques (discussed in Chapter 2); however, DOD predominantly uses algorithmic cost models in estimating software cost (10). Algorithmic cost models have the advantage of objectivity, and of deriving a fast, useful estimate early in the software project (1:330; 10). However, these advantages are

often subdued by the perceived inaccuracy of algorithmic models, as well as the lack of training and experience of the person using the model. Recent research has supported the deficient performance of algorithmic cost estimating models (4:12). Therefore, there is a need to increase algorithmic cost model accuracy.

Because of the myriad of software applications within DOD, and because software cost models are built on and reflect different databases, different cost estimating models are used depending on the specific type of software project being estimated. However, there is no one model that best captures the future costs of software in all situations. Consequently, DOD faces the problem of determining the type of cost model to use which gives the most accurate cost estimate regarding a particular type of software project.

Specific Problem

Given the perceived inaccuracy of algorithmic cost models, DOD must employ calibration principles to improve the estimating accuracy of the SLIM model. Moreover, "calibration of the model is an important consideration at the selection stage as it directly affects the usefulness of the model in the chosen environment. . ." (31:4). SMC would like to have the SLIM model calibrated to their SWDB. Also, given the SLIM model is calibrated to the SMC SWDB, how accurate is the model in estimating SMC type software applications?

Importance

Computer software cost estimation models serve to estimate future development and support costs associated with a particular software effort, and are very useful tools

that facilitate the decision-making process. By successfully calibrating the SLIM model to the SMC SWDB, cost estimates of SMC software-intensive projects will be based on relevant historical data, and as Boehm points out in his summary of his book, Software Engineering Economics, "a well-defined software cost estimation model can help avoid the frequent misinterpretations, underestimates, overexpectations, and outright buy-ins which still plague the software field" (2:19). The accuracy of cost estimating models is very much environment, application, and language dependent. Calibration can greatly improve the estimating accuracy of a cost model. Thibodeau indicates in his evaluation of software cost estimating models, "the use of models that are not calibrated to a given development environment can lead to large estimating errors" (30:5-29). Thibodeau's research suggests that accuracy of calibrated models may increase by a factor of 5 over the uncalibrated version of the same model (30:5-29). That is, if the uncalibrated version provides an estimate within 50 percent of actual cost, the calibrated version should provide an estimate within 10 percent. Other research efforts have also asserted that calibration can improve accuracy of cost estimates (6; 17). Increased cost estimating accuracy supports the decision-making process and provides leadership the necessary confidence in properly allocating resources. This is perhaps more important in today's DOD environment of austere fiscal policies and dwindling budgets.

Research Objectives

The objective of this research is to successfully calibrate the SLIM cost estimating model to the SMC SWDB, and to determine whether calibration improves the accuracy of

the SLIM model in estimating the development efforts of SMC software projects. The objectives are to:

(1) Learn and become knowledgeable about the SLIM cost estimating model, particularly with respect to calibration.

(2) Obtain and become familiar with the 1994 edition, version 1.0, of the SMC SWDB.

(3) Stratify the SMC SWDB into data sets to be calibrated. That is, review and organize the SWDB into homogeneous groups of data.

(4) Calibrate the SLIM model to each of the stratified data sets.

(5) Determine the accuracy of and validate the calibrated model using data reserved for this purpose.

(6) Provide conclusions and observations.

(7) Provide recommendations regarding future calibration efforts.

Scope of Research

This research supports a thesis proposal from the SMC's Cost Division (SMC/FMC) and Management Consulting & Research, Inc. (MCR). The scope of this research effort will focus on calibrating the SLIM model to the SMC SWDB, and improving the estimating accuracy of the SLIM model for SMC software development projects.

Thesis Overview

This research effort uses the SMC SWDB to calibrate the SLIM model. The results of the calibrated model are then validated and the accuracy assessed.

Chapter 2, Literature Review, reviews research efforts and literature in the area of software cost estimation. Specifically:

(1) Software cost estimation techniques are reviewed. The advantages and disadvantages of each technique are then outlined.

(2) A general background in the area of software cost estimation is given. The concept of uncertainty and its relationship to software cost estimation is presented. The use of algorithmic models in software cost estimation is then discussed.

(3) A general background and description of the SLIM model is provided.

Chapter 3, Methodology, gives a general description of the SMC SWDB and presents how the SMC SWDB was stratified and how the SLIM model was calibrated. Additionally, this section outlines the specific methods used in validating the results and assessing the accuracy of the calibrated version of SLIM.

Chapter 4, Findings, presents the results of the calibration effort. This section presents the validation, and accuracy testing results.

Conclusions based on the findings are then given in Chapter 5, Conclusions and Recommendations. Recommendations in the area of calibration and possible follow-on research are then given.

II. Literature Review

Overview

Because computers have touched everyone's life in one way or another, and because of the increasing costs to develop software, the field of software cost estimating is gaining a great deal of attention and respect among top management. This chapter reviews research efforts and literature in the area of software cost estimating. More specifically, this chapter addresses various cost estimation techniques, the basic concepts and background of software cost estimation, calibration efforts in the area of software cost estimation, and provides a general description of the SLIM estimating model.

Cost Estimation Techniques

There are several cost estimating techniques to choose from when estimating software costs. Generally, organizations will base their software cost estimates on past performance represented by historical data from which relevant cost factors are identified (9:72). Historical data, however, is not always available or easily obtained. For the most part, the cost estimation technique used is dependent on the estimating organization's objectives, resources, and the basic capabilities and limitations associated with each technique.

Cost estimates are either top-down or bottom-up (9:72). Top-down estimates basically emphasize total system costs first, whereas bottom-up first focuses on the costs of lower sub-levels of the system and then aggregates these costs to derive a cost for the

total system. Whether an estimate is made top-down or bottom-up, there are several methods available to cost analysts to derive the estimate.

Description of Techniques. The major cost estimation methods used today are: algorithmic models, expert judgement, analogy, Parkinson, price-to-win, top-down, and bottom-up (2:7). The following is a brief description of each method:

> (1) **Algorithmic models:** "These methods provide one or more algorithms which produce a software cost estimate as a function of a number of variables which are considered to be the major cost drivers" (2:7).
>
> (2) **Expert judgement:** This method is based on the experience of one or more experts, and relies on their judgement (2:7).
>
> (3) **Analogy:** This method is based on real projects. It relates the costs of similar past projects to estimate a new project (2:7).
>
> (4) **Parkinson:** "A Parkinson principle ('work expands to fill the available volume') is invoked to equate the cost estimate to the available resources" (2:7).
>
> (5) **Price-to-win:** In this method, the cost estimate is based on the objective of winning the contract or job (2:7).
>
> (6) **Top-down:** A cost estimate for the total system is derived first. "The total cost is then split up among various components" (2:7).
>
> (7) **Bottom-up:** "Each component of the software job is separately estimated, and the results aggregated to produce an estimate for the overall job" (2:7).

One should note that the Parkinson and price-to-win methods are unacceptable and do not produce reasonable estimates (1:337; 2:7; 10). Additionally, note that the first five techniques listed above are usually applied in concert with either a top-down or bottom-up approach toward estimating.

Technique Advantages and Disadvantages. Every cost estimation technique has its advantages and disadvantages, and the particular technique used depends largely on

various limiting factors such as: the availability of historical data, the objective of the estimate, available resources to derive the estimate, and how soon the estimate is required. Therefore, when using a particular cost estimation technique, the capabilities and limitations of that technique should be considered. Table 1 outlines the basic strengths and weaknesses of the common cost estimation techniques.

Table 1
Strengths and Weaknesses of Software Estimation Methods (1:342)

Method	Strengths	Weaknesses
Algorithmic model	• Objective, repeatable, analyzable, formula • Efficient, good for sensitivity analysis • Objectively calibrated to experience	• Subjective inputs • Assessment of exceptional circumstances • Calibrated to past, not future
Expert judgement	• Assessment of representativeness, interactions, exceptional circumstances	• No better than participants • Biases, incomplete recall
Analogy	• Based on representative experience	• Representativeness of experience
Parkinson	• Correlates with some experience	• Reinforces poor practice
Price to win	• Often gets the contract	• Generally produces large overruns
Top-down	• System level focus • Efficient	• Less detailed basis • Less stable
Bottom-up	• More detailed basis • More stable • Fosters individual commitment	• May overlook system level costs • Requires more effort

Boehm suggests that no one method is better than the next in all aspects (2:7). Given this and the variety of software cost estimation approaches, and the individual strengths and weaknesses of each, Fairly, in his book Software Engineering Concepts, suggests that more than one estimation technique should be used if possible, and the results examined to derive a more comprehensive estimate (9:84).

Software Cost Estimation

Background. As mentioned earlier, software development is plagued with cost overruns and schedule slippages, and software costs are increasing at an alarming rate. Software cost estimations serve as measurements in the planning and controlling of software development (7). They assist management in making informed decisions before, during, and after the software development process. However, the software development process is not a clear tangible process like the process of constructing a house. The software development process is a very complex process involving several complex interrelated tasks (15:1). As a result, numerous factors affect the cost of software development. Simply translated, software cost estimating by itself is a very involved undertaking that requires a great deal of understanding of the complexities associated with software development.

Because of the complexity involved in the development of software and the uniqueness of each software project, several software cost models have been developed and are in use today. Before escalating software costs became a significant issue in software development, the time where software costs were less than forty percent of the

total cost of software development and hardware was the dominant cost, software costs estimates were basically derived as a percentage of the hardware they supported (31:30). Today, this type of model produces inaccurate results and is for the most part obsolete, because software costs currently account for about ninety percent of the total cost of the software over its life cycle (31:30).

Most models today are parametric-based algorithmic models that relate output to several inputs. A parametric-based cost model is simply a model that estimates cost based on observed relationships between one or more key characteristics of the system being estimated and cost. Some of the first algorithmic cost models were developed as simple single variable models that estimated the effort, so many lines of code per man-month, to develop software (22:11; 10). The model took a linear form represented by the equation:

$$E = a_0 + a_1 x_1 \tag{1}$$

where

E = the effort expended to develop software expressed in man-months

a_0 = a coefficient determined by a set of data points to give the best fit

a_1 = a coefficient determined by a set of data points to give the best fit

x_1 = a key factor that influences effort (usually size measured in lines of code)

From this equation, costs are estimated by multiplying the effort by an average monthly labor rate. Likewise, dividing the effort by the budgeted manpower gives the estimated time to completion (22:11). This implied that there was a linear tradeoff

between cost, size, and time. If management felt the estimates should be adjusted for reasons that were not reflected by the size of the program, they merely traded manpower for months. Moreover, as Fred Brooks points out in his book, The Mythical Man-Month, effort measured in man-months suggests that manpower and months are "interchangeable commodities" which in-turn suggests that the time to complete a software project varies as a product of manpower and the number of months (3:16). For example, if the manpower were doubled on a project, the time to complete that project would be reduced by half the original time. Brooks shows that this is only true when the separate tasks associated with software development are uncorrelated (3:18). That is, the time and effort necessary to complete a task is independent of all other tasks. This assumption of independent software tasks is violated in large software development projects that involve numerous interrelated tasks. The effects on schedule of adding manpower to software development is illustrated in Figure 1.

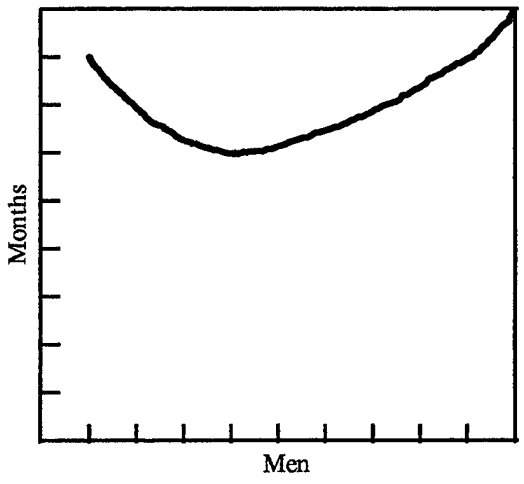


Figure 1. Effects on Schedule when adding Manpower to a Project (3:19)

Brooks submits that when tasks are interrelated, there exists the burden of increased communication. As a result, the time required to develop software increases after a certain point at an increasing rate with the addition of manpower.

Because of the complex interrelated tasks involved in software development, current cost models incorporate several key factors (such as software size and complexity, various project attributes, programmer ability and tools, to name a few) that determine costs which are not expressed in a simple linear relationship--key factors are expressed in multiplicative or "analytic" relationships with costs--with effort, cost, schedule, and productivity (1:330-332). Additionally, a great deal of insight into software cost estimation is provided by examining the Rayleigh-Norden distribution of manpower requirements versus time. Figure 2 depicts the Rayleigh-Norden distribution.

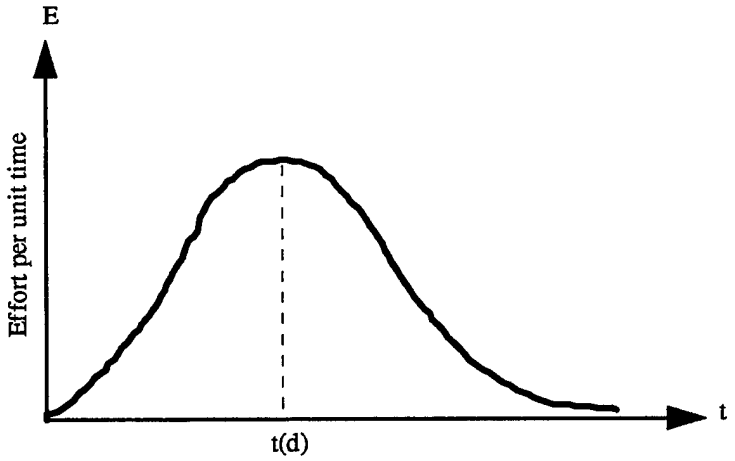


Figure 2. The Rayleigh-Norden Curve of Effort Versus Time (9:80)

Norden showed that the Rayleigh distribution can be used to approximate the manning profile of a software development project at any point in time during the software development life cycle. Note that in Figure 2, t = development time and t(d) = the time at which the curve reaches its maximum value. Basically, the area of any interval of the Rayleigh-Norden curve "represents the total amount of effort expended in that interval" (9:80). Lawrence Putnam of Quantitative Software Management, Inc. further observed that at t(d) a software project usually encounters final testing and preparation for release. Putnam estimated that 40 percent of the area under the Rayleigh-Norden curve is to the left of t(d), and 60 percent to the right primarily because more effort is expended in maintaining a software product after it has been released (9:80; 22).

Software development is, no doubt, very complex, and the cost models of today attempt to capture and quantify that complexity through using empirically derived mathematical functions that relate the estimate to several key cost related inputs. To further understand and appreciate the complexity associated with software development and cost estimating, the idea of "uncertainty" needs to be examined.

Uncertainty. Decision-makers must constantly decide on the proper allocation of resources and evaluate alternative ways of doing business so as to generate profit for their firm. However, the majority of these decisions are made under conditions of uncertainty. Uncertainty basically refers to the degree decision-makers are unsure of the impact of alternative choices presented to them and the resources required to undertake a particular alternative. It represents the difference between what is estimated and what actually happens. Cost estimating assists the decision-making process in that it attempts to

account for and describe (quantify) as much uncertainty as possible regarding alternative courses of action. The goal of cost estimating is to provide decision-makers with necessary and explicit information so that informed decisions can be made. Two major sources of uncertainty exist in the field of cost estimating: requirements uncertainty and cost estimating uncertainty (11:4).

Requirements Uncertainty. "Requirements uncertainty refers to variations in cost estimates stemming from changes in the configuration of the system or force being costed," and is the major source of uncertainty with respect to military systems (11:4). Basically, there is more uncertainty in a software cost estimate if the requirements are not well defined. The idea of requirements uncertainty is best explained by Boehm. He explains that the accuracy of a cost estimate is increased as the software development effort progresses and requirements become more refined (2:8). Obviously, one would expect to be more confident in an estimate of something required a month from now versus a year from now. The limitation that requirements uncertainty presents during the course of a software project is presented in Figure 3. Notice that uncertainty decreases as a project progresses through the software life cycle phases. This would support the notion that algorithmic cost models are more useful early in the software development life cycle. That is, the more uncertain the requirements, the more likely an estimate will be based on algorithms derived from historical data. No matter how much requirements uncertainty exists, there will always be a certain degree of cost estimating uncertainty present throughout the software development process.

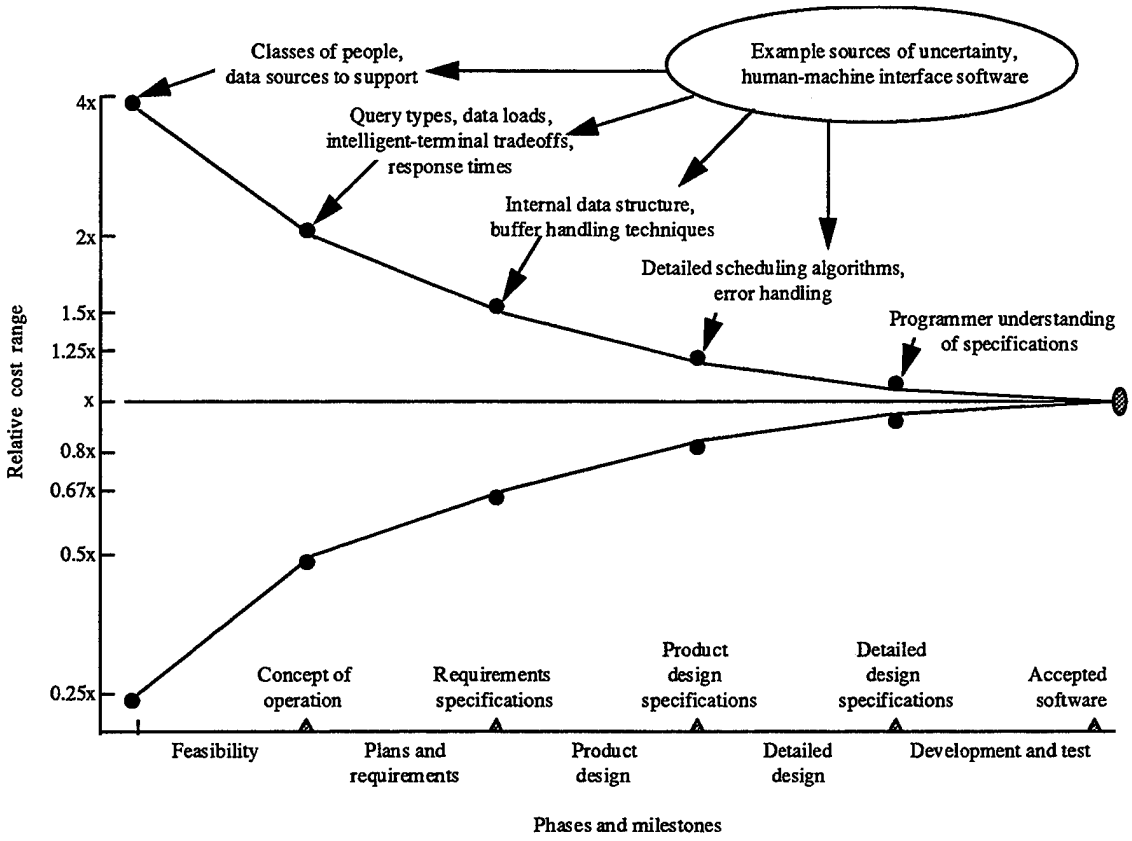


Figure 3. Software Cost Estimation Accuracy Versus Phase (2:8)

Cost Estimating Uncertainty. "Cost estimating uncertainty refers to variations in cost estimates of a system or force where the configuration of the system or force is essentially constant" (11:4). Cost estimating uncertainty generally arises due to cost analyst differences, errors in the data, and errors in developing the cost estimating relationships expressed in the model (11:4). Cost estimating uncertainty tends to be relatively small with respect to requirements uncertainty, however significant variability can occur in estimates as a result of the input that goes into the estimate (11:6).

DeMarco, in his book, Controlling Software Projects, explains that an estimate is only as good as the data that goes into the estimating process (7:17). He refers to this as the estimating dilemma, better known as garbage-in, garbage-out estimating. Figure 4 illustrates this dilemma.

Figure 4. The Estimating Dilemma (7:10)

Recap. Software development is inundated with complex tasks and uncertainty, more so in the earlier stages of the development process. As such, software cost estimating is a process that takes place throughout the software development life cycle, and provides a measurement and foundation on which management can base their planning and controlling decisions. Today's software cost models attempt to quantify and express as much complexity and uncertainty as possible by developing and using algorithmic models based on historical data. But how much past projects reflect future ones becomes an issue by itself when deriving an estimate from an empirically based algorithm. Considering the proliferation of technology in the area of software development and that the software development process is still maturing, deriving an estimate based on past projects could be misleading. As a result, the data used to construct the model should be updated periodically and the model calibrated to specific environments. Basically, in order

to derive more accurate and useful estimates, better input into the estimating process needs to be used. What exactly this input should be is the problem faced by cost analysts and software engineers (11:17). The input and how the input is treated by the model is what separates one cost model from the next. Nonetheless, cost models should, as a minimum, be calibrated to the environment they are estimating within.

Calibration Efforts

There is no one software cost estimation model that is significantly better than the next, and that can better estimate software costs under all situations and environments. This is best pointed out in Wellman's book, Software Costing. He suggests, "that due to the great variability of factors influencing software projects in different organizations, no cost models are truly transportable between environments; ultimately there can be no useful generic model" (31:33). One of the major problems in the software estimation field is that there is a dissimilarity between the models (16). This dissimilarity among the various cost models suggests that a model's ability to accurately estimate software costs is dependent on the specific software development environment the model is estimating within (30). Also, because of this dissimilarity, not much confidence is held in any particular model. Furthermore, according to Genuchten and Koolen, software estimation models are rarely used; as they point out, "a recent survey showed that only 14 percent of the respondents used a model to estimate software development" (12:37). The lack of using software estimation models stems primarily from the lack of predictive accuracy of the models, and dissimilarity among them. Nonetheless, as Wellman again points out,

"that does not mean that organizations should abandon software prediction but rather that each should examine its own environment carefully and produce or calibrate a model suited to its own requirements" (31:4).

The lack of accurate software cost estimation models fosters an increasing need to calibrate software estimation models to specific software environments and applications. Gulezian writes, "it is desirable to develop models which not only apply to specific development environments, but which are also configured to accommodate different project types within an environment as well as developmental differences among environments" (14:235). As mentioned in Chapter 1, calibration of software cost estimation models can significantly enhance a model's predictive accuracy. Thibodeau's research also reveals the importance of properly identifying the characteristics of the model being calibrated relative to the development environment the model is predicting within (30:5-27). He suggests that the criteria for properly stratifying an organization's historical data to be used in calibrating a model should be mapped closely to the actual model structure; the stratified data sets should be representative of the model's key influencing factors. The issue of proper stratification of historical data will be discussed further in Chapter 3.

In using the SMC SWDB to calibrate the SLIM model, the predictive accuracy of the SLIM model will most likely increase for software projects developed within the SMC environment. Subsequently, leadership and management will be better able to properly select, staff, and monitor SMC software development projects similar to the development projects contained in the SMC database.

SLIM, Version 3.1

SLIM is a software, cost estimating tool offered by Quantitative Software Management, Incorporated (QSM) located in McLean, Virginia (25). SLIM is proprietary in that there are restricted rights attached to the software and documentation that support the software (25). SLIM is a powerful management tool designed to estimate, analyze, and present the characteristics of a software project over its life cycle(25:1). "These characteristics include: schedule, effort, cost, staffing and quality" (24).

Foundation. The model is an algorithmic model founded on empirical data of past projects, and based on Lawrence Putnam's "analysis of the software life cycle in terms of the Rayleigh distribution of project personnel levels versus time" (2:10). Basically, Putnam fine tuned the work of Norden (refer to Figure 2) by collecting, and analyzing large amounts of data (7:174). QSM began gathering data to support their model in 1978, and has since then analyzed over 3,500 completed projects. Of the 3,500 projects, 2,800 have been validated as useful in supporting the model (24). The data have come from many sources, and applications. "QSM's data have been gathered and verified from projects in the United States, Canada, Western Europe, Japan, and Australia. . ." (22:17). Table 2 shows the various types of applications (defined in Appendix B) that are represented in QSM's database. The nice thing about QSM's database is that it is updated annually in order to detect and reflect the latest trends in those variables they know are changing. This allows QSM to replace old, out-dated projects with newer, more relevant ones (22:17).

Table 2
QSM Database--Projects by Application Type (22:17)

Business	60%
Real-time embedded	10%
Systems software	8%
Scientific	8%
Command and control	6%
Telecom and message switching	4%
Avionic	2%
Process control	1%
Microcode or firmware	1%

As mentioned, the SLIM model is designed to provide management with the necessary information to track and control a software project throughout the project's life cycle, which is modeled by the Rayleigh-Norden curve. This allows management to view the key characteristics of a project at any point in time or during any time interval; management can see the dollar, manpower, and necessary code to produce for each time period (22:208). There are four life cycle phases that are presented in the SLIM tool that emulate a certain rise, peak, and fall in resource requirements over time: (1) Feasibility Study, (2) Functional Design, (3) Main Build, and (4) Maintenance (22:250-251). These four phases are defined as follows:

> (1) **Feasibility Study.** The Feasibility Study phase is the first phase of a software project's life cycle, and begins when someone feels there is a deficiency that needs correcting. "The objectives of this phase are to develop a complete and consistent set of system needs and to formulate top-level, feasible plans for meeting them" (22:250).
>
> (2) **Functional Design.** This is the next phase, and the objective is to develop a design that fits the needs established by the Feasibility Study (22:250).
>
> (3) **Main Build.** "This phase produces a working system that implements the design specifications and meets the requirements (performance, interface, and reliability) for installation" (22:251).

(4) **Maintenance.** This is the last phase, and represents the support required to keep the product in use by meeting the user's current and changing needs (22:251).

SLIM also uses ten milestones over the phases of the life cycle to track and control projects: (0) Feasibility Study Review (FSR), (1) Preliminary Design Review (PDR), (2) Critical Design Review (CDR), (3) First Code Complete (FCC), (4) Systems Integration Test (SIT), (5) User-Oriented System Test (UOST), (6) Initial Operating Capability (IOC), (7) Full Operational Capability (FOC), (8) 99-percent reliability level (99%), and (9) 99.9-percent reliability level (99.9%). A milestone is merely "an event whose occurrence may be tracked to assure that a project is proceeding on schedule" (22:52). The definition of the various milestones are provided in Appendix C, and Figure 5 depicts the resource profile of a software project by individual life cycle phases, and approximately where each milestone falls along the life cycle of a project.

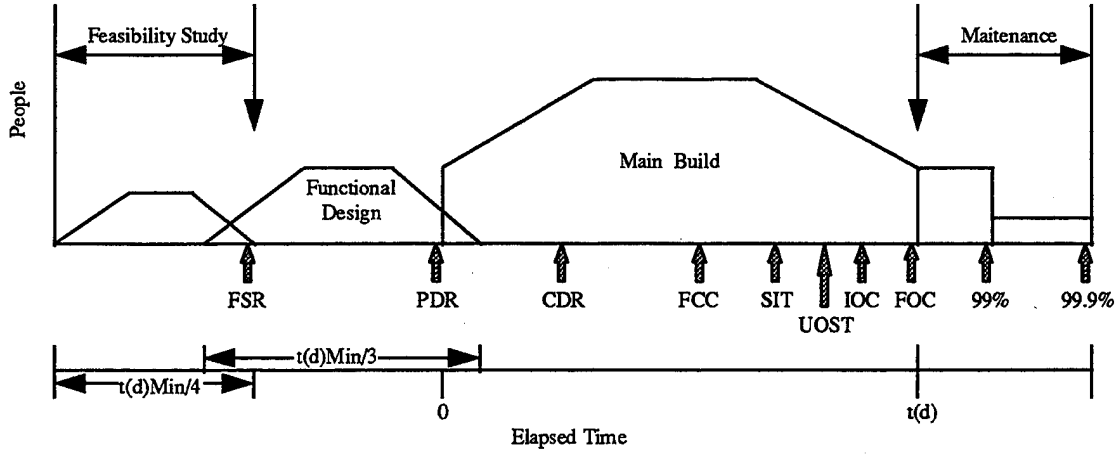


Figure 5. Software Resource Profile by Phase and Milestone (10; 22:53; 24; 25)

Note that t(d) represents the minimum time to develop ("build") the software. Also, the minimum length of the feasibility study is approximated as one fourth of the minimum development time, and the functional design phase as one third (22:224-226). Putnam suggests that the milestones as defined in SLIM are widely used; however, he also suggests that if an organization uses different milestones, their location can be approximated by analogy (22:226). Moreover, "SLIM can be customized to match the phase and milestone scenario of nearly any software life cycle environment..." (16:VII-11).

The notion that SLIM is a very time sensitive model is supported by several authors, and Putnam himself (2:10; 9:69; 7:174; 22). Basically, the model assumes that project effort is "inversely proportional to the fourth power of development time. . ." (9:69). "This curve indicates an extreme penalty for schedule compression and an extreme reward for expanding the project schedule" (9:69). Boehm gives an excellent example of this tradeoff, "this relationship says that one can cut the cost of a software project in half, simply by increasing its development time by 19 percent. . ." (2:10). Nevertheless, Putnam's analysis of effort versus time trend data supports this "fourth-power" tradeoff between time and effort (22). Additionally, Putnam contends that his model (SLIM) is only useful for estimating medium or large software projects, and not small projects (22:216).

> He found that for large projects the model converged acceptably to the sample data. In other words, the Rayleigh curve was useful in predicting how an effort of *n* man-months would be staffed at any given point in its life cycle. For smaller projects, political manpower-loading considerations ("Hide these guys on your project now or you won't have them when you need them") and individual differences tended to cloud the picture. (7:174)

Putnam suggests that this problem with small programs can be overcome if two or more of the following conditions pertain: (1) source lines of code equal or exceed 5,000, (2) development time equals or exceeds six months, (3) effort equals or exceeds 18-20 manmonths, or about $150,000, (4) peak manpower equals or exceeds three people (22:216).

SLIM Modes of Operation. SLIM operates under three different, but interrelated modes: (1) Estimation, (2) History, and (3) Reports and Presentation. All of the modes are useful in planning, tracking, and controlling a software project over its life cycle. Additionally, it is very useful in performing tradeoffs among key software development characteristics. These modes are discussed in the following paragraphs.

Estimation. The Estimation mode allows the user to input, edit, and save key information and assumptions pertaining to a new project estimate. The data to be input includes a measure of the size of the project, applicable life cycle phases and the corresponding shape that manpower loading takes, application type, and a Productivity Index (PI). Basically, the PI (to be discussed in the following pages) is a measure of the productivity of the processes involved in software development during the main build phase; the PI is a measure of the total development environment. The PI can be calibrated to past projects and input directly by the user in deriving a new project estimate or it can be calculated by SLIM. SLIM calculates the PI based on several answers the user gives with regard to questions on manpower, schedule, quality, tools and methods, technical constraints, and personnel profile. The Estimation mode also allows the user to set several constraints (user-defined) on key project characteristics and perform "what-ifs" and

tradeoffs on the results. The user can then store these separate solutions and view them in different ways.

History. The History mode allows the user to input key information on past projects, and evaluate the effects on key estimating parameters. This mode allows the user to calibrate a PI from past projects to be input in the Estimation mode for a new project. The History mode will be discussed in greater length in Chapter 3.

Reports and Presentation. The primary objective of the Reports and Presentation mode is to provide the user an easy, quick means to develop, generate, and present reports on an estimate. The mode provides a variety of graphs and reports on various key project, planning and controlling factors. This mode also provides the user on-line briefing capability (22).

Key SLIM Parameters. The macro-estimation equation in SLIM is primarily a function of three key input parameters: (1) size, (2) PI, and (3) the Manpower Buildup Index (MBI). These factors are discussed in greater detail in the subsections to follow. First, the basic macro-estimation equation in SLIM is

$$S_s = C_k K^{1/3} t_d^{4/3} \qquad (2)$$

where

S_s = the size of the product, measured in lines of code

C_k = a technology constant

K = life-cycle effort, measured in man-years

t_d = minimum development time, measured in years (2:10)

SLIM allows the user to calibrate C_k to past projects or it can be estimated in the estimation mode of SLIM. Simply stated, C_k is a measure of the productivity of the software development process and is equivalent to the PI. Also, note the fourth-power tradeoff, as discussed earlier, represented in Equation (2) between effort and time.

Size. Size refers to the product size as measured by equivalent source lines of code (ESLOC). With respect to the SLIM model, ESLOC refers to only software instructions, as opposed to actual lines of code (22; 24; 25). SLIM uses "logical" ESLOC. That is, if one instruction takes up two lines of code, that instruction will only be counted as one line. ESLOC is calculated by PERT sizing techniques (22). SLIM also allows function points to be input, however the input is then converted in SLIM to ESLOC by use of a "gearing factor" (22; 24; 25). A gearing factor is nothing more than a number used to convert alternative sizing measures such as function points to ESLOC, and is dependent on the programming language used. Lastly, note that users must input "at least 5,000 ESLOC, because the equations used by the program start to lose validity below that size" (22:256).

PI. The PI is a scaled value that represents a corresponding Productivity Parameter (PP). Because of the wide variability associated with PP values, and because PP values are not easily understood, PI values were created to give management a more intuitive scale of numbers (24). The PI and PP are actually measures of the same thing; they constitute a macro measure of the efficiency (productivity) of the total development environment (22:34). Table 3 details the SLIM default values of PI and PP, and where they fall in relationship to application type. Application type is important in determining

the complexity of the software and deriving an initial PI (24). Refer to Appendix B for definitions of application types.

Table 3
SLIM Default PI and PP Values by Application (22:33)

Productivity Index	Productivity Parameter	Application Type
1	754	Microcode
2	987	
3	1220	
4	1597	Firmware (ROM)
5	1974	Real-time embedded Avionics
6	2584	
7	3194	Radar systems
8	4181	Command and control
9	5186	Process control
10	6765	
11	8362	Telecommunications
12	10,946	
13	13,530	Systems software Scientific systems
14	17,711	
15	21,892	
16	28,657	Business systems
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	

PI values range from .1 to 40, 25 being the highest value encountered thus far by QSM (22:34; 24). Lower values of PI and PP "generally are associated with poor

working environments, poor tools and high product complexity" (24). Basically, application types that have low PI values are more difficult to program and design than applications with higher PI values (22:34). Also, as illustrated in Table 3, the PP behaves exponentially with respect to PI. This is obvious when one considers that the PP is actually calculated by Equation (2), the macro-estimation equation, solved for C_k. The equation for PP is merely a rearranged version of Equation (2): $C_k = S_s/(K^{1/3}t_d^{4/3})$. As a result, the PP can be viewed as a function of the time versus effort tradeoff discussed earlier--"an increase or decrease of one value on the PI scale has a significant impact on time and effort" (24). Higher values result in lower effort and shorter schedules. Also, as already mentioned, the PI can be calibrated from historical data and mapped more precisely to an organization's software development environment (see calibration section in Chapter 3).

MBI. The MBI is a measure of the rate personnel are applied to a software project. Like PI and PP, the MBI corresponds to a Manpower Buildup Parameter (MBP). The MBP is calculated from

$$MBP = K/(t_d)^3 \tag{3}$$

where

MBP = Manpower Productivity Parameter

K = life cycle effort, measured in man-years

t_d = minimum development time, measured in years

MBI values typically range from 1 to 6 (going as low as -3), and like the PI, the MBI affects effort and development time. "A steep buildup gets more manmonths under the curve in the early part of the project, reaches peak manpower sooner, and consequently shortens development time" (22:49). Basically, the lower the MBI, the slower the personnel buildup rate. Table 4 shows the SLIM MBI and MBP values, and their corresponding staffing buildup rate.

Table 4
SLIM MBI and MBP Values and Staffing Rate (22:50)

Manpower Buildup Index	Manpower Buildup Parameter	Staffing Buildup Rate
1	7.3	Slow
2	14.7	Moderately slow
3	26.9	Moderate
4	55	Rapid
5	89	Very rapid
6	233	Extremely rapid

Like the PI, the MBI can also be calibrated to past projects and obtained from the History mode of SLIM. A calibrated MBI is a measure of how an organization has applied personnel to past projects (22:256). Unlike a calibrated PI, a calibrated MBI can not be input directly to SLIM when estimating a new project. However, the user can set the MBI value indirectly when deriving an estimate. This requires the user to adjust the size of the project in either the "Ballpark" or "Sensitivity" views of the Estimation mode. The user can then return to the "Assumptions" screen and enter the appropriate size without affecting the MBI value. Consequently, both the PI and the MBI are of interest to

the user with respect to calibration. As such, the intent of this research effort is to calibrate the PI and MBI to the SMC SWDB.

Factors that Influence the PI and MBI. If the user has no historical data and can not derive a calibrated PI, SLIM will calculate a default PI based on current projects in the QSM database. Size and application type are two key factors that influence the PI value, and must be provided in order to enable SLIM to generate a default PI. The default PI is then adjusted by SLIM based on the user's responses to a comprehensive range of questions in the following broad categories: (1) Tooling and Methods, (2) Technical Difficulty, and (3) Personnel Profile. Basically, the user has the option of selecting a value from 1 to 10 that best describes his/her organization's capabilities with respect to the broad category. A value of 1 indicates that there is no capability, and 10 indicates superior capability. The user can also select "UNKNOWN/NA" if the response is unknown or not applicable to the project. Rather than providing a macro response to the broad categories, the user can select "<DETAIL>" among the response scale to enable the user to respond to several more detailed categories within each broad category. A similar scale of possible responses is provided for the detailed categories. Appendix D contains the detailed questions within each broad category. The responses the user gives, either at the broad category level or the more detailed level, are then combined and an adjusted PI is determined (22; 23).

Size, project schedule and resource constraints are the key factors that influence the value of the MBI. As revealed by Equation 3, the MBI is a function of effort and time. It is also dependent on any constraints that might be placed on staffing levels.

As already mentioned, the user can lock both a calibrated PI and MBI value in the Estimation mode. However, once the MBI value is locked, the user can not use the "Constraints" option in SLIM to conduct tradeoffs between time, effort, and required resources (staffing) of an estimate without the MBI being unlocked and recalculated, and then used in the new estimate. This is primarily because the MBI is a function of schedule and resource constraints. Nonetheless, size can be varied with a locked MBI.

Summary

This chapter reviewed literature and research in the area of software cost estimating. Descriptions of the most common cost estimation techniques were given, to include their advantages and disadvantages. A general background of software cost estimation, and its relationship to the complexities and uncertainties of the software development process were examined. The review of literature and research revealed that the software development process is still maturing, and likewise, so is software cost-estimating. As a result, the importance and need for models to be calibrated to an organization's software development environment were discussed. Lastly, this chapter provided a description of the SLIM estimating tool. A brief background, and key characteristics of the SLIM estimating tool were reviewed. Additionally, the basic estimating methodology of the SLIM tool was examined.

Because of the ambiguous nature of the software development process and software cost estimating, a cost analyst should have a good understanding of how the estimating model their working with captures and explains costs. A cost analyst should

also have an understanding of the software development process. Nevertheless, cost estimates will always be wrong; one can only improve on the accuracy of the estimate. Calibrating one's model to past similar projects is one answer toward improving that model's predictive accuracy. Calibrating and improving the accuracy of the SLIM estimating model with respect to the SMC SWDB is the focus of this research.

III. Methodology

Overview

The intent of this research is to calibrate the SLIM cost estimating model to the SMC Software Database (SWDB) and to assess the estimating accuracy of the calibrated model with respect to the baseline (uncalibrated) model. Properly documenting the methods used in this effort is important so that readers can easily follow, repeat and verify the procedures used in this research. This chapter discusses the methods used to calibrate the SLIM cost model. First, the SMC SWDB is described, and stratification and calibration procedures are discussed. Next, the methods and measures used in validating the accuracy of the calibrated model are explained. Lastly, several assumptions and limitations of this research are addressed.

Data Description

The SMC SWDB was used in calibrating the SLIM cost model. The SMC SWDB consists of 2,614 records of software development data. Each record contains approximately 220 data fields containing information that attempts to resemble the user's model and cost structure (27).

The SWDB is an automated (PC based) tool that allows users to easily access and use stored data. The SMC SWDB user's manual gives a nice introduction of the database.

> The Space and Missile Center Software Database was developed to access and display data stored in the Space and Missile Systems Center (SMC) Software Database (SWDB). The SWDB was developed under the direction of the USAF Space and Missile Systems Center, with assistance from the Space Systems Cost Analysis Group (SSCAG). (18:2)

The SWDB is a user-friendly program that provides a variety of applications. Users can quickly query information along user-defined criteria. The SWDB also allows users to generate a variety of reports on the queried data. A convenient feature of the SWDB is that it also operates under a "Windows" interface.

Several sources of data comprise the SMC SWDB: (1) government, (2) industry, and (3) other database sources. Government sources include SMC, European Space Agency (ESA), NASA, and Air Force Materiel Command (AFMC) programs. Industry sources include major aerospace companies, suppliers, non-aerospace companies, and model developers. Other data sources include aerospace, SSCAG, General Dynamics, ESA, and Jet Propulsion Laboratory (27).

The actual data was collected and entered into the SWDB under five broad categories of information: (1) proprietary data, (2) general information, (3) cost, size and schedule information, (4) software characteristics, and (5) maintenance information. The proprietary data is the only category of information that is not accessible to the user, thereby protecting the anonymity of the program and contractor(s). This category includes information regarding the program's name and the development contractor(s) (18). Refer to Appendix E for a general breakdown of each category.

Stratification of SWDB

The first step in this research was to stratify the SWDB into data sets to be used in calibrating SLIM. Stratifying the SWDB simply refers to reviewing and organizing the SWDB into homogeneous subsets of data. Fortunately, the SWDB program simplifies the

stratification process. As mentioned previously, the SWDB allows the user to query data along several parameters and user-defined effort and size ranges.

The six primary parameters on which data can be queried are: (1) software level, (2) operating environment, (3) applications, (4) software functions, (5) programming language, and (6) confidence level. However, the sixth parameter (confidence level) has since been deleted from the SWDB because it was merely a measure of the confidence level of one model developer (28). Software level refers to the level at which the software development data is reported (i.e. project, Computer Software Configuration Item, Computer Software Component, Computer Software Unit). Operating environment reflects the primary mission of the software (i.e. unmanned space, military ground, military avionics, etc.). Application reflects the type of application the software is used for (i.e. command and control, signal processing, message switching, etc.). Software function describes the function(s) of the software (i.e. interface, display, communication, etc.). Programming language reflects the language(s) used in the software development. Additionally, the user can specify effort and size ranges in the query mode of the SWDB.

For the purpose of this research, the SWDB was stratified along the following operating environments: unmanned space, military avionics, military mobile, missile, and military ground. The military ground data set was stratified further by three application types: command and control, signal processing, and management information system (MIS). Additionally, all seven subsets of data (unmanned space, military avionics, military mobile, missile, military ground/command and control, military ground/signal processing, and military ground/MIS) were restricted to the Computer Software Configuration Item

(CSCI) software level. To be more environment specific, the stratified subsets were limited to software developed in the USA. The stratified data subsets were restricted further by size, effort, schedule, and minimum peak staffing requirements. These constraints are discussed further in the following calibration section.

Calibration

The primary focus of this research is to calibrate SLIM to the stratified subsets of data stated in the previous section. Calibration was limited to the development efforts of a program. In this case, calibration of the SLIM cost model reflected only the main build phase of a program's life cycle. In particular, it reflected only the effort used from PDR to FOC (refer to Figure 5).

The SMC SWDB used the life cycle phases specified in DOD-STD-2167A as the basis for breaking down the software development life cycle (29). Although the phases and milestones stated in DOD-STD-2167A do not match the SLIM phases and milestones, they still can be mapped to the SLIM phases and milestones. Figure 6 illustrates this mapping.

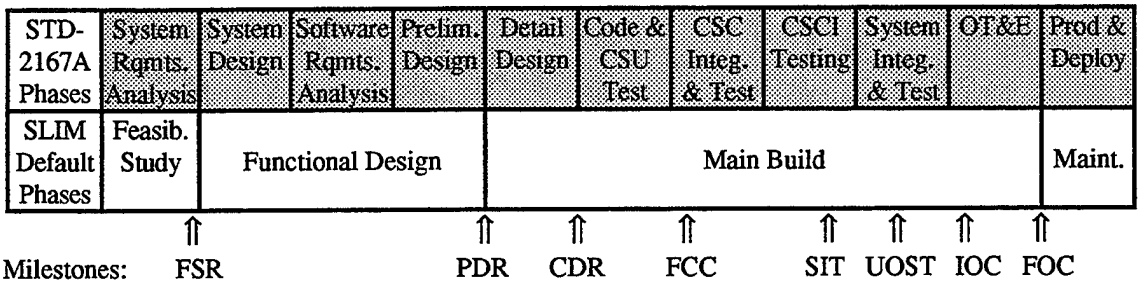


Figure 6. Mapping of SLIM Phases and Milestones to DOD-STD-2167A (16:VII-3)

Note that the SLIM PDR milestone maps closest toward the completion of the Preliminary Design phase, and FOC maps closest toward the completion of the Operational Test and Evaluation (OT&E) phase. As a result, calibration of the SLIM tool was restricted to software development effort consumed from PDR through OT&E.

As discussed in Chapter 2, calibration is achieved through the History mode of the SLIM tool. The user needs only to input key information (effort, schedule, and size) on past programs, and SLIM outputs a PI and MBI for each program and an average PI and MBI for all the programs included in a historical data file. At a minimum, users must input effort, schedule, and size information for each program. Figure 7 presents a basic look at the SLIM calibration process.

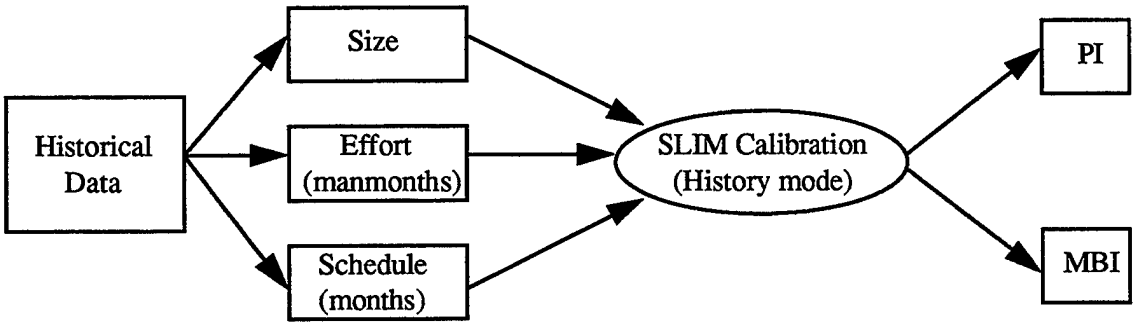


Figure 7. SLIM Calibration Process

Users can also input number of defects for each program. However, since the current SMC SWDB has very limited information regarding defects, and the calibration

efforts were centered on solely the main build phase of a program's life cycle, defect information was not considered in the calibration process.

As discussed in the description of the SMC SWDB, each record contains approximately 220 data fields. Unfortunately, not all data fields are populated with information. As such, the suitability of each record for use in calibrating the SLIM cost model was first determined. Putnam indicated that four conditions should be met when estimating a software project: (1) equivalent source lines of code (ESLOC) equals or exceeds 5,000, (2) schedule equals or exceeds six months, (3) effort equals or exceeds 20 manmonths, and (4) minimum peak manpower equals or exceeds three people (22). He suggested that the estimating algorithms of SLIM begin to lose their validity if these conditions are not met. Consequently, the same limiting factors used in estimating development effort were also subject to the records used in calibrating the SLIM model. These limiting factors are discussed in greater detail in the paragraphs to follow.

ESLOC Equals or Exceeds 5,000. Records to be used in calibrating SLIM required a minimum of 5,000 ESLOC. ESLOC as reported by the SMC SWDB is defined very much like ESLOC reported by SLIM. "A Source Line of Code is defined as a single instruction, not necessarily a physical line" (26:3-1). Normalized effective size as reported by the SMC SWDB was used in calibrating SLIM. As a result, when stratifying the SWDB, the size of each record queried was restricted to 5,000 ESLOC (normalized effective size) in the query mode of the SMC SWDB.

The SMC database reported the total size of a program as the sum of new code and pre-existing code. So as not to overstate the size estimate that is associated with the

reported development effort, normalized effective size was used in calibrating SLIM. Normalized effective size was considered a more appropriate size measure to use in calibrating the SLIM tool because it attempts to reflect the percentage of pre-existing code that was modified. Normalized effective size was calculated and reported in the SMC database using the following ground rules:

> (1) "If there is a reported new size and no pre-existing size, then it is assumed all code is new code and normalized size and effective size are equal to reported new size" (19:F-2).
>
> (2) If there is reported new and pre-existing size, then normalized effective size is equal to actual new size plus 40% of the re-design percentage (as specified in the SMC database for applicable record) plus 25% of the re-implementation percentage plus 35% of the re-tested percentage. If no percentages were reported, then averages were calculated over the entire SMC database and used at the time of normalization (19:F-3).
>
> (3) "If there is a reported estimation of the new size and no reported pre-existing or new size, then both normalized total size and normalized effective size are equal to new size estimation" (19:F-3).
>
> (4) If there are reported estimated new and pre-existing sizes, but no actual size, then normalized effective and normalized new size are calculated as in case 2, except estimated new size is used instead of actual new size (19:F-3).

It should be noted, however, that by using normalized effective size to calibrate SLIM, the calibration results depend largely on the creditability of the normalization procedures (identified above) used within the SMC database.

Schedule Equals or Exceeds Six Months. Suitable records for calibration also required a minimum of six months development time. The SMC SWDB also reports the schedule in months for a software project. However, this schedule information (item 3.5.2 of Appendix E) is based on the total effort (from Systems Requirements through System Test and Integration) of a software program. Since this research is only concerned with

development effort from PDR through OT&E, and because no procedures were available to normalize the reported schedule information to reflect only the development effort from PDR through OT&E, the reported schedule information was not used (28). Rather than making some broad assumptions and attempting to normalize the reported schedule information, and in an effort to maintain as much objectivity in calibrating the SLIM tool, the actual schedule was derived by examining the actual milestone completion dates of PDR and OT&E (item 3.5.3 of Appendix E). The actual schedule that reflected development effort from PDR through OT&E was calculated simply by determining the elapsed time from the actual completion dates of the PDR and OT&E milestones. For example, if PDR was completed on 01/89 (MM/YY) and OT&E on 03/92, then the actual schedule that reflects the development effort from PDR through OT&E was calculated as 38 months (the difference between 01/89 and 03/92). Deriving the actual schedule by examining the milestone completion dates assumes, however, that there were no breaks in the development effort from PDR to OT&E.

Effort Equals or Exceeds 20 Manmonths. Records also required a minimum of 20 manmonths of effort. Normalized effort as reported by the SMC SWDB reflects the total development effort from PDR through CSCI testing (19:F-2). Table 5 lists the percentages by phase, used by the SMC SWDB to normalize effort. The percentages listed in Table 5 refer only to software development effort (29).

Table 5
SMC SWDB Percentages used to Normalize Effort (19:F-2)

Phase	Percentage of Normalized Effort
Software Requirements	5.5
Preliminary Design	11.4
Detail Design	19.1
Code and Unit Test	29.8
CSC Testing and Integration	35.6
CSCI Testing	4.1
Systems Test and Integration	7.2
OT&E	4.8

Normalized effort was calculated and reported in the SMC database using the following ground rules:

(1) "If total effort was reported and based on preliminary design through CSCI testing, then the normalized effort is equal to total effort" (19:F-2).

(2) If total effort was reported but based on other than preliminary design through CSCI testing, then normalized effort equals total effort scaled by the percentages assigned to each phase in Table 5 (19:F-2).

(3) If total effort was reported but no phases were, then it was assumed that total effort was based on preliminary design through CSCI testing, and equals normalized effort (19:F-2).

(4) If there was no total effort, then there was no normalized effort (19:F-2).

Unfortunately, since this research effort focused on software development effort from PDR through OT&E, normalized effort had to be adjusted manually to reflect development effort absorbed during the Systems Test and Integration, and OT&E phases. The adjustment was made by using the percentages identified in Table 5. Therefore, normalized effort was scaled up by 12 percent (7.2 percent for Systems Test and

Integration, and 4.8 percent for OT&E). As an example, if normalized effort was reported as 20 manmonths, then adjusted normalized effort was calculated as 20 manmonths pus 12 percent (20 plus 12 percent of 20 equals 22.4 manmonths).

In addition, normalized effort was based on 152 hours per manmonth for the vast majority of records contained in the SMC database. Nonetheless, normalized effort for a few records were based on other than 152 hours per manmonth. Consequently, every record contained in the stratified data subsets had to be reviewed for the number of hours per manmonth (item 3.1.2 of Appendix E). If the number of hours per manmonth was other than 152, then the reported normalized effort was manually adjusted to reflect 152 hours. This was done by multiplying the reported normalized effort by the number of hours per manmonth in item 3.1.2 (Appendix E), then dividing by 152. If hours per manmonth were not reported, then normalized effort was assumed to be based on 152 hours and no adjustment was made (28).

In order to capitalize on the SMC SWDB automatic querying function, records were restricted in the query mode to a minimum of 15 manmonths of normalized effort. However, normalized effort then had to be adjusted manually and reviewed to ensure only records with a minimum of 20 manmonths of adjusted normalized effort were used for calibrating the SLIM cost model.

Minimum Peak Manpower Equals or Exceeds Three People. Lastly, each record included in the four subsets of the SWDB required a minimum peak manpower of at least three people. Minimum peak manpower refers to the number of people that should be working at the peak of the software project (22:220). Although SMC SWDB

records include a peak manpower field, this field was often discovered not to be populated. Consequently, in an effort to provide consistency in the review of records for minimum peak manpower, minimum peak manpower was calculated by the following equation described by Putnam (22:30):

$$PKMP = 1.5\ E/t_d \tag{4}$$

where

PKMP = minimum peak manpower

E = development effort for main build phase

t_d = development schedule in months for main build phase

In addition to the limiting factors placed on each record, records for the unmanned space, military avionics, military mobile, and missile subsets were restricted (stratified) further to the predominate application type reflected in each subset. Because application type is a key factor that influences the PI (discussed in Chapter 2) and the estimate is sensitive to the PI, the SLIM model was calibrated to subsets that specified a common application type. The predominate application type within the unmanned space, military avionics, military mobile, and missile operating environments was chosen primarily to extend the availability of useable records for calibration and validation purposes. Also, note that three application types (command and control, signal processing, and MIS) were already specified for the military ground subset.

Upon completing the screening of records in each data subset for suitability in calibrating the SLIM model, data subsets were reviewed to ensure each subset contained at least nine suitable data records. If a subset contained eight or less records, then all the data records for that subset were used solely for calibrating the SLIM model, and validation efforts of the calibrated model were abandoned. However, the SLIM tool was not even calibrated to data subsets that contained fewer than four records; both calibration and validation efforts were abandoned for data sets with fewer than four records. Although the SLIM tool can be calibrated to one record, the ability to assess that calibration is severely diminished.

Once each data subset was reviewed for the minimum required records, those data subset which contained more than eight records were then partitioned into two distinct groups of data: Group 1, and Group 2. Group 1 data sets were reserved strictly for calibrating SLIM, whereas Group 2 data sets were reserved for validation. This was done primarily to see how well both the calibrated and uncalibrated models estimated development effort from data that was excluded from calibration.

Group 2 data sets were partitioned by ranking each record in each subset from lowest to highest reported ESLOC. A record was then chosen at random to be the starting point for identifying Group 2 records. From that point in the ranking, every third record in the ranking was selected to be included in Group 2 data sets. This was determined to give the best representation of each subset for validation purposes (validation will be discussed more in the validation section). Group 1 data sets included all records not included in Group 2 data sets.

The following is a brief summary of the ground rules used in reviewing the number of suitable data records to be used in calibration and validation:

(1) If the data set has less than four data points (records), then calibration and validation was not conducted.

(2) If the data set has between four and eight data points, all the points were used for calibration and none were used for validation.

(3) If the data set contains more than eight data points, one third (rounded to the nearest multiple of 3) of the points were used for validation and the rest for calibration (e.g. 10 records, use 3 for validation and 7 for calibration).

These ground rules were determined by the advisor and sponsors of this research effort to be sufficient in providing a sound base for calibrating the SLIM model and validating the accuracy of the calibrated model.

Finally, once the subsets were partitioned, the SLIM cost model was calibrated using Group 1 data sets. A historical file for each Group 1 data set was created in the History mode of SLIM. This was accomplished by inputting SWDB reported size, schedule and effort parameters of each record in the History mode of SLIM (refer to Figure 6). SLIM then returned the calculated PI and MBI of each record included in a historical file in addition to the average PI and MBI of the total historical file. The average calibrated PI and MBI were then locked (procedure described in Chapter 2) in the Estimation mode of SLIM to establish a calibrated model for each data subset. Figure 8 provides a conceptual look at the stratification and calibration process used in this research.

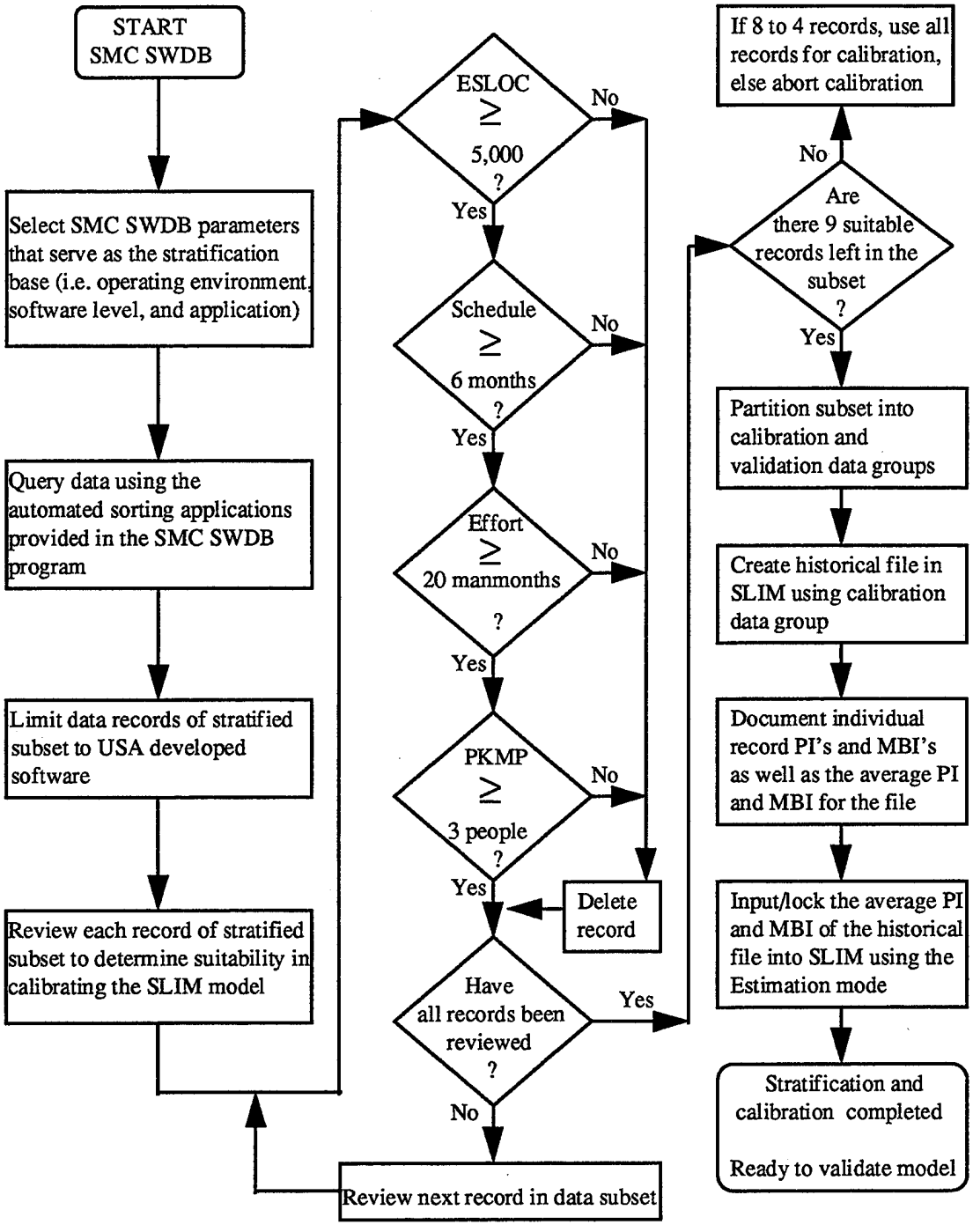


Figure 8. Flow Diagram of SWDB Stratification and SLIM Calibration Process

Validation

Because of the sensitive nature of the PI and MBI and their impact on an estimate, successfully using and capitalizing on the SLIM estimating tool may depend largely on the user's ability to calibrate consistent PI's and MBI's within a particular historical data set. Consequently, after stratifying the SMC database and attempting to calibrate SLIM to each of the stratification platforms (unmanned space, military avionics, military mobile, missile, military ground/command and control, military ground/signal processing, and military ground/MIS), the range of PI and MBI values for each calibration data set (Group 1 data) was charted to reflect the variability of these parameters across separate stratified historical data sets. Additionally, the standard deviation (σ) of each set of PI's and MBI's was calculated. The standard deviation is a measure of how wide a set of observations are distributed about the mean of those observations. The standard deviation is given by the equation

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n - 1}} \qquad (5)$$

where

n = total number of records in a particular data set

i = the ith record (software project) in a particular data set

x = the calibrated PI (or MBI) of a record

$\bar{x}$ = the mean calibrated PI (or MBI) of the data set in question

After the variability of the PI and MBI values was assessed for each historical file, the estimating accuracy of the calibrated versus the uncalibrated SLIM tool was assessed. First, using the data that was set aside for validation purposes (Group 2 data), effort estimates were derived from both the calibrated and uncalibrated models. In deriving an estimate through both the calibrated and uncalibrated versions of SLIM for a particular software development project, the actual software size was input in the Estimate mode. Additionally, estimates only reflected the main build phase of a program's life cycle and assumed the default manpower load given in SLIM for that phase. The only other required input was application type. Application type was input when estimating only from the uncalibrated version of SLIM. As discussed in Chapter 2, selecting an application type is required in SLIM when not using a calibrated PI. Inputting a software size estimate and application type will return a default PI and MBI for that software. Furthermore, any information contained in the software characteristics section of each record was used to adjust the default PI through the broad and detailed categories discussed in Chapter 2. Second, once the estimates were derived from both the calibrated and uncalibrated models, estimated effort was compared to actual normalized effort.

Three measures were used to validate the estimating accuracy of the calibrated and uncalibrated models: (1) the Magnitude of Relative Error (MRE), (2) the Mean Magnitude of Relative Error (MMRE), and (3) the Prediction at Level *l* (PRED(*l*)) (5:172-173). These measures were then compared in an attempt to assess the estimating accuracy of the calibrated versus uncalibrated models. Additionally, the Wilcoxon Signed-Rank Test was used to test for bias associated with the calibrated model.

Magnitude of Relative Error (MRE). The MRE reflects the degree of estimating error in a particular estimate. This measure was calculated for each software project contained in the validation data sets. MRE is defined by the equation

$$MRE = |E_{actual} - E_{estimate}| / |E_{actual}| \quad (6)$$

where

E_{actual} = actual normalized total development effort reported in the SMC SWDB

$E_{estimate}$ = estimated total development effort as reported by SLIM

Note that as the MRE becomes smaller, the estimate is said to become more accurate (reflect less error) (5:172).

Mean Magnitude of Relative Error (MMRE). The MMRE reflects the average degree of estimating error produced by a set of estimates. An MMRE was calculated for each validation data set of both the calibrated and uncalibrated models. The MMRE is defined by the equation

$$MMRE = \frac{1}{n} \sum_{i=1}^{n} MRE_i \quad (7)$$

where

n = total number of records (software projects) in a particular data set

i = the ith record in a particular data set

MRE = Magnitude of Relative Error

Note that the smaller the MMRE the better the model produces on average a set of estimates (5:172).

Prediction at Level *l* (PRED(*l*)). This measure is sometimes known as the percentage method and is used frequently in validating and reporting the predictive accuracy of a model. It basically reflects the percentage of project estimates in a given data set that fall within a predefined percentage of their actual values. This measure is defined as

$$\text{PRED}(l) = \frac{k}{n} \tag{8}$$

where

l = predefined percentage of actual values

k = number of software projects in a particular data set of n projects whose MRE $\leq l$

n = total number of software projects in a particular data set

As an example, "if PRED(0.25) = 0.83, then 83% of the predicted values fall within 25% of their actual values" (5:173). With respect to this research effort, this measure reflected the percentage of estimates within a given validation data set that fell within 25 percent of their actuals ($l = 0.25$).

Wilcoxon Signed-Rank Test. The Wilcoxon signed-rank test is a non-parametric test and was used to test for bias in the distributions of the estimate and actual observations. This was accomplished by ranking the differences between the actual and

estimate observations. The absolute value of the differences were ranked from the least to the greatest. If the data were truly unbiased, one would expect that just as many negative differences would occur as would positive differences, thereby the number of positive and negative differences would sum to zero. As such, the ranking was then partitioned into rankings of positive (T^+) and negative (T^-) differences. "Sizable differences in the sums of the ranks assigned to the positive and negative differences would provide evidence to indicate a shift in location between the distributions" (20:680). Simply put, if there exists a significant difference in the sums of the ranks assigned to the positive and negative differences, one would conclude that the estimate observations, when compared to the actuals, are bias toward being either high or low.

The smaller of the rank sum of the positive versus negative differences was then used as a test statistic (T) to test the null hypothesis (H_o) that the center of the paired (positive and negative) distributions is zero. Since all the validation data sets used in this research were relatively small (had less than 26 observations) this test was simplified by merely comparing the T to some value, T_o. The applicable T_o value can be derived from any statistics table giving the "critical values of T in the Wilcoxon matched-pairs, signed-ranks test" (20:780). For $T \leq T_o$, H_o is rejected, thereby signifying that the distributions of the estimates and actuals differ in location (20:681). One could then conclude that the estimates are biased.

The Wilcoxon T table provided in the book Mathematical Statistics with Applications was used in this research (20:780). However, the Wilcoxon T table used in this research requires at least five data records ($n = 5$), therefore, only data sets (Group 2

data sets) containing five or more data records were tested. Lastly, the Wilcoxon signed-rank test was conducted at both the 90 and 95 percent confidence levels ($\alpha = .05$ and .025, respectively, for a two-tailed test).

Assumptions and Limitations

This research is based on the assumption that the information contained in the SMC database is accurate and that there were no breaks in the development effort of a project from PDR to OT&E. This research is also limited to only estimating the development effort of the main build phase of a software project. Moreover, uncertainty in the size of a software project was not an issue addressed in this research. That is, size input was assumed to be certain and effort estimates did not reflect any uncertainty associated with size that would normally be indicative of a software cost estimate.

Additionally, the results of the Wilcoxon test are contingent on the assumption that the distribution of the differences between the estimates and actuals is symmetric (8:603).

Objectivity should be maintained to the fullest extent possible when calibrating any model. Because this research relied on normalized effort and normalized effective size in calibrating the SLIM tool, the results of this research are contingent on the credibility of the normalization procedures used within the SMC database for size and effort.

IV. Findings

Overview

This chapter presents the analysis of the data and gives the results. The first part of this chapter gives the results of the stratification of the SMC database to the seven platforms specified in Chapter 3. Next, the results of calibrating SLIM to the stratified data subsets are presented. Finally, the results of the validation effort are presented.

SMC Database Stratification Results

The SMC data base was initially stratified along seven separate operating environments/application types: (1) unmanned space, (2) military avionics, (3) military mobile, (4) missile, (5) military ground/command and control, (6) military ground/signal processing, (7) military ground/MIS. The seven stratified sets were refined further to include only projects defined at the CSCI software level and developed in the USA. Additionally, the records included in the stratified data sets were subject to the size, schedule, effort, and peak manpower constraints specified in Chapter 3.

Analysis of the stratified data subsets revealed that ten data records in the unmanned space operating environment met all the stratification requirements. However, Sherry Stukes of Management Consulting and Research revealed that records within the unmanned space operating environment were improperly coded in the SMC SWDB (29). She indicated that the records within the unmanned space environment should have been defined along ground versus flight parameters. Consequently, the original ten records of

the stratified data set were reviewed to see which records pertained to ground applications in support of unmanned space as opposed to flight applications.

Review of the original stratified unmanned space data set revealed that nine of the ten records pertained to ground, while only one record pertained to flight. As a result, the one record pertaining to flight was deleted from the stratified data set. Subsequently, the operating environment stratification platform was redefined as ground in support of unmanned space (unmanned space-ground).

The unmanned space-ground data set was stratified further to reflect the predominate application type specified within the data set. It turned out that all nine records within the data set specified a command and control application type.

Two data records in the military avionics environment met the stratification requirements, while no records in the military mobile environment and only one in the missile environment met the criteria. The military ground/command and control data set had no data records that met all the criteria and military ground/signal processing had only one record that met the criteria. The stratification results of the military avionics, military mobile, and missile operating environments reflect the number of records prior to further stratification along application type. Nonetheless, further stratification and calibration efforts were abandoned for these data sets, as well as for the military ground/signal processing data set, due to insufficient data records.

There were, however, six records in the military ground/MIS data set that met all the stratification criteria. Table 6 gives a summary of the SMC database stratification results. In addition, Appendix F presents the adjusted normalized effort calculations.

Table 6
Summary of SMC Database Stratification Results

Record #	PDR Date (MM/YY)	OT&E Date (MM/YY)	Schedule (t_d) ≥ 6	Adj. Norm. Effort (E) ≥ 20	Normalized Eff. Size ≥ 5000	PKMP ($1.5E/t_d$) ≥ 3
Unmanned Space-Ground/Command & Control						
0074	10/82	09/85	35	90	11700	3.84
0075	05/81	09/85	52	1021	116800	29.46
0076	03/81	09/85	54	129	14000	3.58
0077	05/81	09/85	52	586	56200	16.90
0078	06/81	02/86	56	535	48300	14.34
0079	05/81	09/85	52	484	50300	13.96
0080	09/81	09/85	48	332	69450	10.36
0081	03/81	09/85	54	184	22900	5.10
0082	03/81	02/86	59	157	16300	3.99
Military Avionics						
0014	07/84	04/86	21	520	22148	37.12
0346	10/85	03/90	53	732	40000	20.73
Military Mobile **No Records Met Stratification Criteria**						
Missile						
0036	02/81	07/83	29	538	13658	27.81
Military Ground/Command & Control **No Records Met Stratification Criteria**						
Military Ground/Signal Processing						
0054	11/83	08/84	9	142	45035	23.71
Military Ground/MIS						
2526	03/91	12/92	21	236	6681	16.88
2527	03/91	12/92	21	263	7457	18.80
2528	03/91	12/92	21	763	21588	54.48
2610	03/91	12/92	21	513	14536	36.64
2611	03/91	12/92	21	419	11840	29.92
2612	03/91	12/92	21	349	9899	24.96

In an attempt to derive a broader (more data points) stratified data set than any of the original seven sets listed in Table 6, the SMC database was stratified to the military ground operating environment across all application types (command & control, signal

processing, MIS, etc. . .). This was also done in an effort to examine the results of calibrating the SLIM tool to a historical set of data that was not stratified along any particular application type. Analysis of this data set revealed that 25 records met all the stratification criteria. The results of this stratification are given in Table 7.

Table 7
Stratification Results -- Military Ground/All Application Types

Military Ground/All Application Types						
Record #	PDR Date (MM/YY)	OT&E Date (MM/YY)	Schedule (t_d) ≥ 6	Adj. Norm. Effort (E) ≥ 20	Normalized Eff. Size ≥ 5000	PKMP ($1.5E/t_d$) ≥ 3
0002	01/87	07/88	18	260	28805	21.65
0004	02/84	05/87	39	94	46000	3.62
0005	02/84	08/85	18	55	16000	4.57
0006	03/84	08/85	17	74	41000	6.52
0024	04/86	07/88	27	253	18000	14.06
0028	11/83	07/85	20	942	112917	70.64
0030	05/83	02/85	21	170	41000	12.16
0048	04/78	11/79	19	365	136227	28.83
0053	11/83	08/84	9	54	21122	8.96
0054	11/83	08/84	9	142	45035	23.71
0055	11/83	01/85	14	7127	260882	763.56
0056	11/83	08/84	9	25	22150	4.11
0058	01/84	01/85	12	679	28191	84.84
0059	01/84	01/85	12	156	15554	19.46
0061	10/83	06/84	8	120	72716	22.47
0062	10/83	06/84	8	40	16428	7.56
0063	10/83	06/84	8	25	11753	4.62
0301	09/86	11/88	26	457	76566	26.36
2497	01/91	09/93	32	*89	10000	4.17
2526	03/91	12/92	21	236	6681	16.88
2527	03/91	12/92	21	263	7457	18.80
2528	03/91	12/92	21	763	21588	54.48
2610	03/91	12/92	21	513	14536	36.64
2611	03/91	12/92	21	419	11840	29.92
2612	03/91	12/92	21	349	9899	24.96
* Adjusted from 160 to 152 hours per manmonth: [85(160)]/152 = 89						

SLIM Calibration Results

Calibrating the SLIM cost model to a historical data set basically involves calibrating a PI and MBI value that best represents that data set.

Unmanned Space-Ground/Command & Control. Since the unmanned space-ground/command and control data set had nine data records, the data set was partitioned into two sets of data--six records were reserved for calibration while three were reserved for validation purposes (refer to Chapter 3). Using the six records reserved for calibration purposes, the PI value was calibrated to 2.5 and the MBI value to -1.8. Table 8 gives a summary of the calibration results using the unmanned space-ground data set.

Table 8
SLIM Calibration Results -- Unmanned Space-Ground/Command & Control

Unmanned Space-Ground/Command & Control						
i	Record #	Schedule (Months)	Adj. Norm. Effort	Normalized Eff. Size	Calibrated PI	Calibrated MBI
1	0074	35	90	11700	1.1	-1.1
2	0076	54	129	14000	0.1	-2.4
n/a	0082	59	157	16300	**	**
3	0081	54	184	22900	0.9	-2.4
4	0078	56	535	48300	3.0	-1.7
n/a	0079*	52	484	50300	**	**
5	0077	52	586	56200	4.0	-1.3
6	0080	48	332	69450	6.1	-1.8
n/a	0075	52	1021	116800	**	**

* Randomly chosen starting point--every third point from there was reserved for validation purposes
** Records reserved for validation data set
n = 6
Mean PI = 2.5
Mean MBI = -1.8

Military Ground/MIS. Since the military ground/MIS data set had only six data records, all the records were used for calibrating the SLIM cost model. The PI value for this data set was calibrated to 1.7 and the MBI value to 3.2. Table 9 gives a summary of the calibration results using the military ground/MIS data set.

Table 9
SLIM Calibration Results -- Military Ground/MIS

Military Ground/MIS						
i	Record #	Schedule (Months)	Adj. Norm. Effort	Normalized Eff. Size	Calibrated PI	Calibrated MBI
1	2526	21	236	6681	0.2	2.5
2	2527	21	263	7457	0.5	2.7
3	2528	21	763	21588	3.8	3.9
4	2610	21	513	14536	2.4	3.7
5	2611	21	419	11840	1.8	3.4
6	2612	21	349	9899	1.3	3.1
n = 6 Mean PI = 1.7 Mean MBI = 3.2						

Military Ground/All Application Types. The military ground data set was not partitioned into two sets of data. All 25 data points were used for calibration purposes. This was done primarily because data records that might have been reserved for validation purposes could not readily be used to derive an estimate through the uncalibrated version of the SLIM cost model. This was due to vague application types specified within the military ground environment--not all the application types within the military ground environment could be mapped to a particular application type specified in the SLIM tool. For example, the "test" application type specified in the SMC SWDB can cover a broad gamut of application complexity, depending on what type of application was being tested.

Nevertheless, the PI value was calibrated to 8.1 and the MBI value to 2.8 using the military ground data set across all application types. Table 10 gives a summary of the calibration results using the military ground data set.

Table 10
SLIM Calibration Results -- Military Ground/All Application Types

Military Ground/All Application Types							
i	Record #	Application Type	Schedule (Months)	Adj. Norm. Effort	Normalized Eff. Size	Calibrated PI	Calibrated MBI
1	0002	Test	18	260	28805	7.8	2.6
2	0004	OS/Executive	39	94	46000	7.2	-2.6
3	0005	Simulation	18	55	16000	6.7	1.1
4	0006	Test	17	74	41000	11.6	0.7
5	0024	Test	27	253	18000	2.9	1.5
6	0028	Mission Plan.	20	942	112917	11.6	3.4
7	0030	Test	21	170	41000	9.3	1.0
8	0048	Mission Plan.	19	365	136227	14.0	2.3
9	0053	Test	9	54	21122	12.1	3.8
10	0054	Sig. Processing	9	142	45035	14.7	4.3
11	0055	Test	14	7127	260882	14.2	7.9
12	0056	Database	9	25	22150	13.4	2.6
13	0058	Test	12	679	28191	8.6	5.7
14	0059	Test	12	156	15554	7.4	4.4
15	0061	MMI/Graphics	8	120	72716	17.7	4.4
16	0062	Utilities	8	40	16428	11.8	4.2
17	0063	Test	8	25	11753	11.0	3.5
18	0301	Training	26	457	76566	9.5	1.3
19	2497	Diagnostics	32	*89	10000	0.9	-0.7
20	2526	MIS	21	236	6681	0.2	2.5
21	2527	MIS	21	263	7457	0.5	2.7
22	2528	MIS	21	763	21588	3.8	3.9
23	2610	MIS	21	513	14536	2.4	3.7
24	2611	MIS	21	419	11840	1.8	3.4
25	2612	MIS	21	349	9899	1.3	3.1

* Adjusted from 160 to 152 hours per manmonth: [85(160)]/152 = 89
n = 25
Mean PI = 8.1
Mean MBI = 2.8

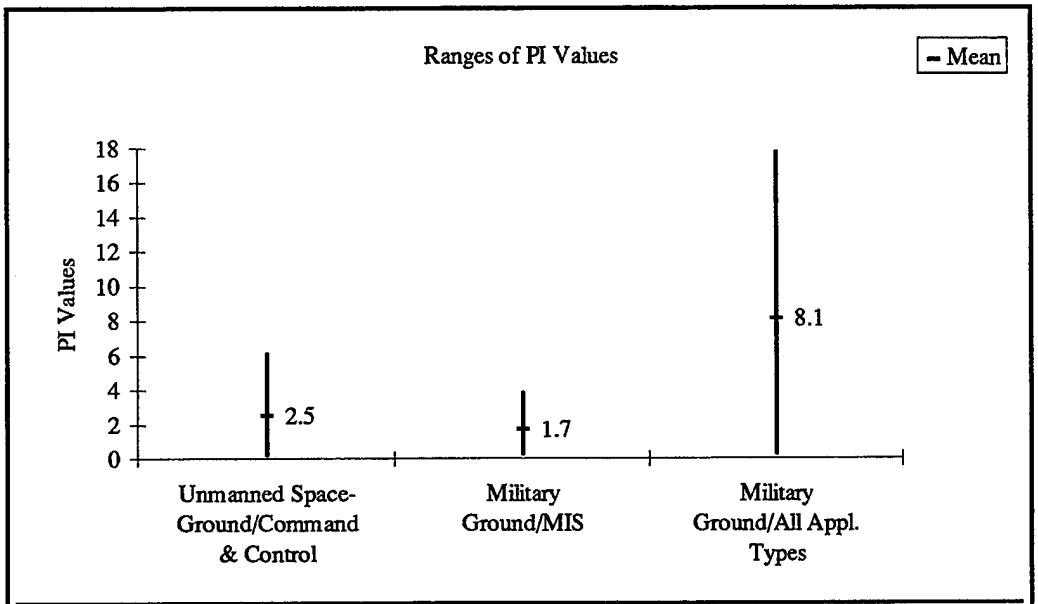
Validation Results

Successfully calibrating the SLIM cost model may depend largely on calibrating consistent PI and MBI values across historical programs of the same data set. As such, the variability of PI and MBI values within a given data set was examined. Additionally, the magnitude of relative error (MRE), mean magnitude of relative error (MMRE), and the percentage method (PRED(l)) were employed to give the reader an idea of the estimating capability of the calibrated SLIM cost model. Lastly, the Wilcoxon signed-rank test was conducted to see if the estimates from the calibrated SLIM model were bias toward being high or low.

PI Variability. The mean PI value for the unmanned space-ground/command and control data set was calibrated to 2.5. However, individual record PI values ranged from a high of 6.1 to a low of 0.1. The standard deviation (σ) of PI values for unmanned space-ground indicated a relatively loose distribution of PI values. The standard deviation was calculated to 2.27.

Analysis of the PI values for the military ground/MIS data set revealed an improvement with respect to the consistency of PI values. The mean PI value was calibrated to 1.7 using the military ground/MIS data set. Additionally, the distribution of PI values for the military ground/MIS data set was tighter than the distribution of PI values calibrated to the unmanned space-ground environment. Individual record PI values ranged from a high of 3.8 to a low of 0.2, and the standard deviation was 1.32.

The military ground environment across all application types disclosed the least consistent results with respect to the variability of the PI value. The mean PI was 8.1, while the high value was 17.7 and the low was 0.2. Furthermore, the standard deviation was 5.17. Figure 9 presents a summary of the variability of PI values for each data set.



PI Summary

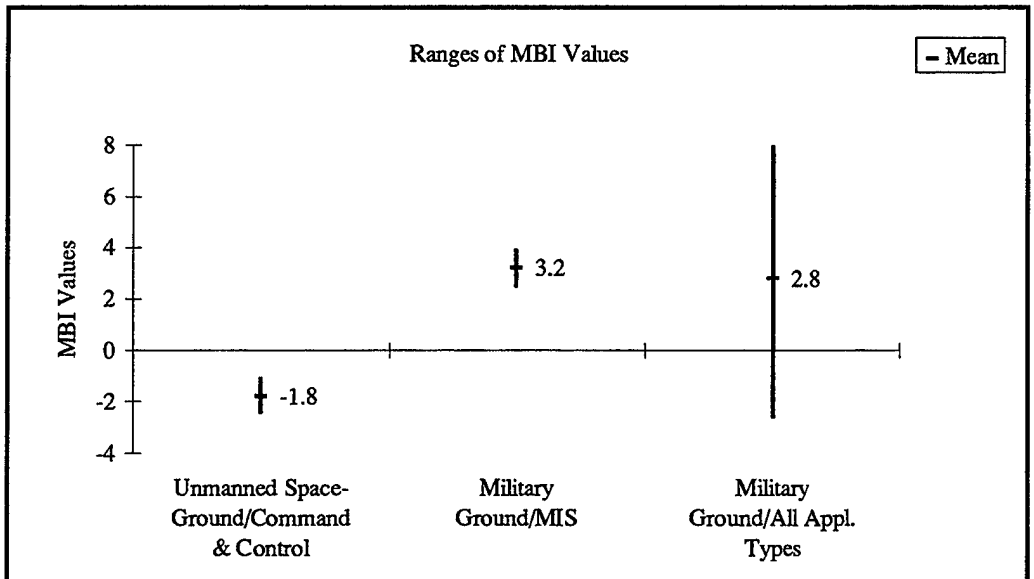
	High	Low	Mean	σ
Unmanned Space-Ground/Com. & Ctrl.	6.1	0.1	2.5	2.27
Military Ground/MIS	3.8	0.2	1.7	1.32
Military Ground/All Application Types	17.7	0.2	8.1	5.17

Figure 9. PI Variability by Stratified Data Set

MBI Variability. The mean MBI for the unmanned space-ground/command and control data set was calibrated to -1.8, while the high value was -1.1 and the low was -2.4. The standard deviation of PI values for the unmanned space-ground operating environment was 0.54.

The mean MBI for the military ground/MIS data set was 3.2. The spread of the distribution of MBI values for the military ground/MIS environment was relatively consistent with the spread of MBI values for the unmanned space-ground environment. The high value was 3.9 and the low was 2.5, while the standard deviation was 0.55.

Again, the military ground environment across all application types disclosed the least consistent results with respect to the variability of the MBI value. The mean MBI was calibrated to 2.8, while the high was 7.9 and the low was -2.6. Additionally, the standard deviation of MBI observations was 2.09. Figure 10 presents a summary of the variability of MBI values for each data set.



MBI Summary

	High	Low	Mean	σ
Unmanned Space-Ground/Com. & Ctrl.	-1.1	-2.4	-1.8	0.54
Military Ground/MIS	3.9	2.5	3.2	0.55
Military Ground/All Application Types	7.9	-2.6	2.8	2.09

Figure 10. MBI Variability by Stratified Data Set

MRE and MMRE. The effects of calibration were assessed for all three data sets. This was accomplished by using MRE and MMRE measures. However, there was a difference in the way the calibration of the SLIM cost model was assessed between the three different operating environments.

(1) Set of 9 records (unmanned space-ground/command and control). Six records were used exclusively for calibration, while three records were reserved to validate the change in estimating accuracy resulting from calibration.

(2) Set of 6 records (military ground/MIS). All six records were used for calibration, as well as for validating the change in estimating accuracy resulting from calibration. This was done primarily because the stratification process had significantly restricted the number of suitable data sets and the number of records contained in each data set. Any attempt to reserve any of the six records exclusively for validation would have resulted in too few records to effectively carry out any validation measures. Note that the estimates that were derived from the uncalibrated model assumed that the MIS application type specified in the SMC SWDB best mapped to the systems software application type defined in SLIM.

(3) Set of 25 records (military ground/all application types). All 25 records were used for calibration. Validating the change in estimating accuracy resulting from calibration could not be accomplished. This was the case primarily because of the inability to accurately match the application types specified in the SMC database to those specified in the SLIM tool. Because application type is a key input in the baseline SLIM model, estimates could not reasonably be obtained from that model. As a result, validating a change in the estimating accuracy resulting from calibration could not be accomplished. Nevertheless, all 25 records were used to assess the estimating accuracy of the calibrated model. The same measures used for validation purposes were also used for this assessment, though.

It should also be noted that software characteristic information (item 4.0 of Appendix E) that was specified for each record in the SMC database was not used in adjusting the default PI value when estimating from the uncalibrated version of the SLIM cost model. This was the case primarily because drawing any type of relationship between the software characteristic information listed in the SMC database and the detailed

adjustment categories contained in the SLIM tool would have been subjective. Additionally, the software characteristic information section of each data record contained very little information. And when there was some information, most of the information was coded as nominal--nominal just so happens to be the default setting for the SLIM detailed PI adjustment categories.

The MMRE of the estimates from the model that was calibrated to the unmanned space-ground environment was 66.6%, whereas estimates from the uncalibrated model showed a slight improvement in the MMRE--MMRE equal to 62.1%. This indicates that the average estimated effort from the calibrated and uncalibrated models is within 66.6% and 62.1%, respectively, of the average actual effort. The lowest MRE of the estimates from the calibrated model was 36.7 %, while the highest was 105.9%. The estimates from the uncalibrated model had a low MRE of 57.2%, while the highest MRE was 71.7%.

The estimates from the model that was calibrated to the military ground/MIS environment produced the best results with respect to the MRE and MMRE measures. The MMRE of the estimates from the calibrated model was only 15.7%, while the MMRE of the estimates from the uncalibrated model was 96.2%. The lowest MRE of the estimates from the calibrated model was 2.1%, and the highest was 46%. The lowest MRE of the estimates from the uncalibrated model was 95.3%, and the highest was 96.5%.

The least effective results were produced by the estimates from the model that was calibrated to the military ground environment across all application types. The average error in the set of estimates was in excess of 200% of the actuals--MMRE equal to

216.6%. Though the lowest MRE for this set of estimates was 1.4%, the highest MRE was 924.2%. Detailed MRE and MMRE results for all three data sets are in Appendix G.

PRED(*l*). The percentage method was administered to all three data sets with respect to the results of both the calibrated and uncalibrated model. However, note that there were no estimates from the uncalibrated model for the military ground/all application types data set. The test was conducted at *l* equal to 25%.

0% of the estimates from both the calibrated and uncalibrated model for the unmanned space-ground/command and control data set came within 25% of the actuals. 83.33% of the estimates from the calibrated model for the military ground/MIS data set came within 25% of the actuals, while 0% of the estimates from the uncalibrated model fell within 25%. Only 8% of the estimates from the calibrated model for the military ground/all application types data set came with 25% of the actuals. Detailed results are in Appendix G. In addition, Table 11 presents a summary of the test results.

Table 11
Summary of MMRE and PRED(*l*) Results

	Uncalibrated SLIM		Calibrated SLIM	
	MMRE	PRED(.25)	MMRE	PRED(.25)
Unmanned Space-Ground/ Command and Control	0.621	0.00	0.666	.0000
Military Ground/MIS	0.962	0.00	*0.157	*.8333
Military Ground/All Application Types	n/a	n/a	2.166	.0800
* Best results				

Wilcoxon Signed-Rank Test. The Wilcoxon signed-rank test was conducted on the military ground/MIS and military ground/all application types data sets. The test was

given with respect to the estimates of the calibrated model. The test indicated that there was no evidence of bias in the estimates from the calibrated models at both the 90% and 95% levels of confidence. Detailed results of the test are in Appendix H.

V. Conclusions and Recommendations

Overview

This chapter gives a summary of the objectives of this research. Additionally, conclusions are drawn from the stratification of the SMC database and from the SLIM cost model calibration and validation efforts. Lastly, this chapter provides recommendations regarding future SLIM calibration efforts, and regarding the SMC database and stratification of that database.

Summary

The objectives of this study was to calibrate the SLIM cost model to stratified subsets of the SMC database and to assess the accuracy of the calibrated model. This was done with the intent of determining whether calibrating the SLIM cost model would improve the accuracy of the model, and whether the model would be useful in estimating the development effort of SMC type software projects.

First, in order to gain a general understanding of the area of software cost estimating, a literature review was accomplished. Additionally, the basic framework and estimating methodologies of the SLIM tool were examined. A review and knowledge of the SMC SWDB was also necessary to effectively conduct this research.

Once a working knowledge of the SLIM tool and the SMC SWDB was obtained, the SMC database was stratified along operating environment and software application type parameters. The subsets were then stratified further along software size,

development effort, development schedule, and minimum peak staffing criteria. The SLIM cost model was then calibrated separately to each suitable stratified data set.

After the model was calibrated, the variability of the calibrated PI and MBI values pertaining to each data subset was examined. This was accomplished primarily through charting the range of PI and MBI values that were derived from each data set. Additionally, the standard deviation of each set of PI and MBI values was calculated, which gave further indication of the variability of these values within each data set.

Estimates of the development effort, with respect to the records contained in each subset of data, from both the calibrated and uncalibrated versions of the SLIM cost model were then derived. The accuracy of these estimates was assessed primarily through employing such measures as the MRE, MMRE, and percentage method.

Lastly, the Wilcoxon signed-rank test was administered in an effort to detect any bias in a set of estimates.

Conclusions

The results of this study were extremely limited with respect to assessing the accuracy of the calibrated SLIM cost model. This was for the most part due to insufficient suitable data in which to effectively calibrate and assess the predictive accuracy of the SLIM model.

SMC Database Stratification. The SMC database was not structured with the SLIM cost model in mind. This is supported by the insufficient number of data sets and records within each data set that resulted from the stratification process. Although the

SMC database consists of 2,614 records of software development data, the vast majority of records that were excluded by the stratification process was due to the lack of schedule information provided for each record. Development schedule information was essential in calibrating the SLIM cost model. The unmanned space-ground/command and control data set consisted of only nine records, while the military ground/MIS data set consisted of only six records. Furthermore, even after eliminating the requirement to specify an application type in the stratification process, the military ground data set across all application types consisted of only 25 records.

SLIM Calibration. Successfully calibrating the SLIM cost model depends largely on the user's ability to calibrate a consistent PI and MBI across a set of historical software projects. This was more the case with the PI value as opposed to the MBI value, though. The PI values for the military ground/MIS data set were significantly more consistent than the values obtained for the unmanned space-ground and military ground/all applications types data sets. The standard deviation of the PI observations for the military ground/MIS was 1.32 as opposed to 2.27 and 5.17 respectively for the unmanned space-ground and military ground/all application types data set.

The MRE and MMRE results for the military ground/MIS data set give evidence that calibrating a consistent PI for a given data set plays heavily on the predictive accuracy of the calibrated model. The MMRE of the estimates from the model that was calibrated to the military ground/MIS environment was only 15.7%, whereas the MMRE of the estimates from the model that was calibrated to the unmanned space-ground and military ground/all application types data sets were 66.6% and 216.6% respectively. Furthermore,

the calibrated model for the military ground/MIS environment can be said to predict software development effort for projects of that environment within 15.7% of the actual effort, 83.33% of the time (within the model's data base).

In calibrating the SLIM cost model to a historical set of data, that data set should as a minimum be stratified along a particular application type. The results of the SLIM model that was calibrated to a data set that had not been stratified along any particular application type were very poor. In particular, the predictive accuracy of the SLIM model that was calibrated to the military ground across all application types environment was extremely poor. This is evidenced by the MMRE of 216.6% for this data set. Moreover, this model was estimated to predict development effort within 25% of the actual effort, only 8% of the time. Actually, in reviewing the MRE's for this data set, the calibrated model is estimated to predict effort within 50% of the actual effort, only 12% of the time.

On the whole, the SLIM model that was calibrated to the military ground/MIS data set generated the best results with respect to calibration and predictive accuracy. Table 12 presents a summary of test results.

Table 12
Summary of Results

	Uncalibrated SLIM		Calibrated SLIM		Calibrated PI & MBI Variability	
	MMRE	PRED(.25)	MMRE	PRED(.25)	σ (PI)	σ (MBI)
Unmanned Space-Ground/ Command and Control	0.621	0.00	0.666	.0000	2.27	*0.54
Military Ground/MIS	0.962	0.00	*0.157	*.8333	*1.32	0.55
Military Ground/All Application Types	n/a	n/a	2.166	.0800	5.17	2.09
* Best results						

SLIM Validation. The ability to objectively assess the predictive capability of the calibrated model was impaired due to the number of suitable records contained in each data set. Because of the insufficient number of records contained in each data set, the means by which the predictive accuracy of the calibrated model was assessed are subject to dispute. One might suggest that in using three of only nine records of the unmanned space-ground data set for validating the accuracy of the calibrated model might be inappropriate--that is, the results might be subject to and dependent on the randomness of the three records chosen for validation purposes. Others might also suggest that in using the same six records that were used for calibrating the SLIM model to the military ground/MIS data set for validation purposes, the validation results could be skewed in favor of the calibrated model. Nevertheless, the predictive accuracy of the calibrated model was assessed subject to the confines of suitable data.

Recommendations

A recommendation regarding the usefulness of the SLIM cost model in estimating SMC type applications can not reasonably be made. The SLIM cost model should be calibrated to historical data that is stratified along application type and development organization. Calibrating the SLIM model to past software projects that were developed by various organizations defeats the meaning of the PI, which is a macro measure of an organization's complete development environment.

Unfortunately, the SMC database could not be stratified along a particular development organization because of the proprietary nature of this information. However,

perhaps the development organization for each record could somehow be coded in the SMC database in a way to allow for this stratification while still protecting the anonymity of the organization. Given the ability to calibrate the SLIM cost model to projects developed by a particular organization, the predictive accuracy of the calibrated model may be exceptional.

Additionally, the structure of the SMC database should be modified slightly in order to accommodate the calibration of the SLIM cost model. Future calibration efforts of the SLIM cost model to the SMC database will only be useful if sufficient development schedule information is available for each record, thereby generating more suitable records that can be used to effectively calibrate and validate the SLIM model. Nevertheless, the following are a few suggestions for further thesis efforts.

(1) Calibrate SLIM to historical data that is stratified by development organization and application type. Then assess the predictive ability of the calibrated SLIM model with respect to its ability to accurately estimate development effort.

(2) Calibrate SLIM in the same manner as stated above; however, assess the predictive ability of the calibrated model with respect to estimating development schedule.

(3) Investigate SLIM's "fourth-power" tradeoff between effort and time, and examine its effects with respect to successfully calibrating the SLIM model.

Remember that the results of these efforts are contingent on having sufficient historical data to effectively accomplish calibration, while at the same time having enough data to properly assess the change in predictive ability of the calibrated model.

Appendix A: Glossary

Accuracy. The degree of error in an estimate. An estimate is said to be more accurate if the amount of error in that estimate is reduced.

Algorithm. "A mathematical set of ordered steps leading to the optimal solution of a problem in a finite number of operations" (4:5).

Calibration. "Establishes the value of a parameter by measuring all the other factors in a situation and solving for the parameter. Specifically, measures the variables of completed projects--size, effort, time--and solves the [SLIM] software equation for the productivity parameter and manpower buildup parameter" (22:362).

Cost Estimating. "The art of collecting and scientifically studying costs and related information on current and past activities as a basis for projecting costs as an input to the decision process for a future activity" (4:5).

Cost Model. "A tool consisting of one or more cost estimating relationships, estimating methodologies, or estimating techniques and used to predict the cost of a system or its components" (4:5).

Magnitude of Relative Error (MRE). A measure of accuracy that reflects the degree of error in a particular estimate. Specifically, it is the absolute difference between an estimate and actual observation, divided by the actual observation (5:172).

Main Build Phase. "Produces a working system that implements the specifications and meets the requirements in terms of performance, reliability, and interfaces. Begins with detailed logic design, continues through coding, unit testing, integration, and system testing, and ends at full operational capability" (22:366).

Manpower Buildup Index (MBI). A management scale, corresponding to the manpower buildup parameter, that " represents the rate of staff increase during the main build" (22:366).

Mean Magnitude of Relative Error (MMRE). A measure of accuracy that reflects the average degree of error produced by a set of estimates. Specifically, it is the sum of the individual MRE measures for a set of estimates divided by the number of estimates (5:172).

Prediction at Level *l* (PRED(*l*)). Sometimes referred to as the percentage method, the PRED(*l*) is a measure that represents the percentage of estimates that fall within a predefined amount of their actuals. Specifically, it is the percentage of estimates in a set of estimates whose MRE is less than or equal to a preset value (*l*) (5:173).

Productivity Index (PI). "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" (22:368).

Standard Deviation (σ). "A measure of the amount of variability about the mean of a set of numbers" (22:370).

Stratification. The process of reviewing and arranging a set of data into homogeneous subsets of data.

Validation. The process of determining whether a cost model is estimating what it is suppose to be estimating. Specifically, it is the process of assessing the accuracy of the calibrated model's estimates and determining whether calibration of that model improved the accuracy (predictive capability) of that model.

Wilcoxon Signed-Ranked Test. A non-parametric test that indicates whether there is bias in a set of observations (20:680).

Appendix B: SLIM Application Type Definitions (16:E-4)

Microcode/Firmware (Most Complex). 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 generally written in C, Ada or Assembly language. Typical examples are military systems like radar, signal processing, and missile guidance systems.

Avionic. Software that is on-board and controls the flights and operations of the aircraft.

System Software. Layers of software that sit between the hardware and applications programs. Examples are operating systems (DOS, UNIX, VMS, etc.), executives or database management systems, network projects, and image processing products.

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, and electric utility power control systems.

Telecommunications and Message Switching. Software that facilitates the transmission of information from one physical location to another. Examples are telephone switches, transmission systems, modem communication products, fax communication projects, and satellite communications products.

Scientific. Software that involves significant computation and analysis. Examples are statistical analysis systems, graphics projects, and 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 (Least Complex). Software that automates a common business function. Examples are payroll, personnel, order entry, inventory, material handling, warranty and maintenance products.

Appendix C: SLIM Milestone Definitions (22:57)

0. FSR-Feasibility Study Review. A software requirements review held at the completion of the feasibility study or system definition phase to assess the risks associated with commencing the Functional Design. The high-level software requirements specifications are approved at this time.

1. PDR-Preliminary Design Review. (Nominal from -0.1 t_d to +0.1 t_d) Earliest time that a formal review of the functional design specifications can be expected to be satisfactory enough to continue into the next phase of development. Functional design and (high-level) system engineering are essentially complete.

2. CDR-Critical Design Review. A review of the detailed logic design for each element of the system. Design consists of flow charts, HIPO diagrams, pseudo code logic, or equivalent. Held when design and coding are separated by management decisions, for example, when required by a Military Standard. Coding cannot start until after a successful CDR under this philosophy. Sufficient design to code from.

3. FCC-First Code Complete. In a top-down, structured design and coding environment, FCC is the time at which all units of code have been written, the units have been peer and management checked, successfully compiled and run as units, and are thought to be satisfactory end-product code. Entered into library of completed code. (Note: Coding will continue thereafter as rework of these modules as integration, testing, and quality-assurance actions force changes to be made.)

4. SIT-(Start of) Systems Integration Test. The earliest time that all elements and subsystems have been put together and the system can work together as a complete integrated package and be demonstrated as such in a formal system test.

5. UOST-(Start of) User-Oriented System Test. Following correction of deficiencies resulting from SIT, the first time that a test of the system in a full user environment--target machine and operating system, real data, real operating conditions--can be conducted.

6. IOC-Initial Operating Capability, or start of installation, depending on the environment. A careful, tentative first use under rigid control. Often a first site installation in a live environment with anticipated later multisite deployment. Star of operation in parallel with the predecessor system in a single site replacement environment.

7. FOC-Full Operation Capability. System meets specified quality standards sufficiently well that organizations will use it in everyday routine mission operations. The quality standard we employ at this point is 95-percent reliability. Ninety-five percent of errors have been found and fixed. This level is suitable for use where reliability is not critical. (Calibration and productivity factors are normalized to this level.)

8. 99%-99-percent reliability level. Ninety-nine percent of errors have been found and fixed. Further work--typically stress testing with final hardware--has been carried out to improve mean time to defect.

9. 99.9%-99.9-percent reliability level. Ninety-nine point nine percent of the original body of errors have been found and fixed. This is the point at which the system is considered to be 'fully' debugged. However, there can never be complete assurance that all the defects have been found and fixed.

Appendix D: SLIM Detailed Input Categories (25:170-183)

Tooling and Methods

Development System

1a. How familiar are you with the development hardware (0-10)?

1b. How available is the development system (0-10)?

Database Management System

2a. How extensive is the role of the database management system (0-10)?

2b. What is your DBMS tools capability (0-10, 0 = no tools)?

Screen Writer

3a. How screen intensive is the system (0-10)?

3b. What is your screen writer capability (0-10, 0 = no tools)?

Report Writer

4a. How report intensive is the system (0-10)?

4b. What is your report writer capability (0-10, 0 = no tools)?

File Handling

5a. How intensive is file handling in this system (0-10)?

5b. What is the capability of your file handling tools (0-10, 0 = no tools)?

Tools

6. How well do you use diagramming tools (0-10, 0 = no tools)?

7. How well do you use testing tools (0-10, 0 = no tools)?

8. How well do you use programming tools (0-10, 0 = no tools)?

9. How well do you use configuration management tools (0-10, 0 = no tools)?

10. How well do you use project management tools (0-10, 0 = no tools)?

11. How well do you use documentation tools (0-10, 0 = no tools)?

12. How well do you use quality assurance tools (0-10, 0 = no tools)?

13. How good are your database conversion utilities (0-10, 0 = no tools)?

14. How well are your tools integrated (0-10, 0 = no tools)?

Development Standard

15. How robust are your development standards (0-10, 0 = no standard)?

16. How well do you adhere to your development standards (0-10)?

17. What is your level of experience with them (0-10)?

18. How well do your standards tailor to different sized systems (0-10)?

Technical Difficulty

Constraints

1. How memory intensive is this system (0-10)?

2. How data intensive is this system (1-10)?

3. How complex is data manipulation in this system (1-10)?

New Algorithms

4a. What is the volume of new algorithms in this system (0-10)?

4b. What is the complexity of developing the new algorithms (1-10)?

New Logic

5a. What is the volume of new logic in this system (1-10)?

5b. What is the complexity of developing the new logic (1-10)?

Complexity

6. What is the volume of expected requirements changes (1-10)?
7. How complex is the customer interface (1-10)?
8. How complex is the external system interface (1-10)?
9. How difficult is existing code integration and testing (1-10)?
10. How intensive are documentation requirements (0-10)?

Platform Stability

11a. How stable is the hardware platform (1-10)?

11b. How stable is the system software platform (1-10)?

Personnel Profile

Management/Leadership

1. How good is management and leadership on this project (0-10)?

Environment

2. What is the availability of training (0-10)?
3. What is the anticipated level of staff turnover (0-10)?

Development Team

4. What is the availability of skilled manpower (0-10)?
5. What is the level of functional knowledge (0-10)?
6. How experienced is the development team with this application type (0-10)?
7. How motivated is the development team (0-10)?
8. How cohesive is the development team (0-10)?
9. What is the level of human communication complexity (0-10)?

Appendix E: SMC SWDB Data Collection Breakdown (18)

1.0 Proprietary Data
- 1.1 Program Name
- 1.2 Program Description
- 1.3 Development Contractor(s)
  - 1.3.1 Prime Development Contractor(s)
  - 1.3.2 Subcontractor(s)
- 1.4 Preparer Information
  - 1.4.1 Name of Preparer
  - 1.4.2 Telephone Number
  - 1.4.3 Fax Number
  - 1.4.4 Preparer Title
- 1.5 Date Prepared

2.0 General Information
- 2.1 End User
- 2.2 Contracting Agency
- 2.3 Development Country
- 2.4 Type of Contract
- 2.5 Software Level
  - 2.5.1 Functional Description
- 2.6 Operating Environment
- 2.7 Application
- 2.8 Software Function
- 2.9 Development Standard

3.0 Cost, Size and Schedule Information
- 3.1 Cost
  - 3.1.1 Total Effort
  - 3.1.2 Hours/Person Month
  - 3.1.3 Average Staffing
  - 3.1.4 Peak Staffing
  - 3.1.5 Labor Categories Included (%)
  - 3.1.6 Phases Included
- 3.2 Source Lines of Code (SLOC)
  - 3.2.1 New Unique SLOC
  - 3.2.2 Common SLOC
  - 3.2.3 Reused SLOC
  - 3.2.4 % Re-Design Effort of Reused
  - 3.2.5 % Re-Implementation Effort of Reused
  - 3.2.6 % Re-Test Effort of Reused
  - 3.2.7 % Total Code Development for Firmware
- 3.3 COTS Packages

- 3.3.1 Number of COTS Packages
- 3.3.2 Size (SLOC) of COTS Packages
- 3.3.3 Integration Complexity
- 3.4 Total Database Size (in bytes)
- 3.5 Schedule
  - 3.5.1 Approximate Year of Development
  - 3.5.2 Schedule Months
  - 3.5.3 Schedule

4.0 Software Characteristics

- 4.1 Level of Complexity
- 4.2 Programming Languages
- 4.3 Software Attributes
  - 4.3.1 Application Complexity *
  - 4.3.2 Turnaround Time with Recompile *
  - 4.3.3 Requirements Volatility *
  - 4.3.4 Rehosting Requirements *
  - 4.3.5 Display requirements *
  - 4.3.6 Reusability Requirements *
  - 4.3.7 Security Level *
  - 4.3.8 Memory Constraints *
  - 4.3.9 Timing Constraints *
  - 4.3.10 Real Time *
- 4.4 Source Code Mix
- 4.5 Functional Metrics
- 4.6 System Architecture
- 4.7 Development Method
- 4.8 Personnel Attributes
  - 4.8.1 Personnel Experience *
  - 4.8.2 Personnel Capabilities *
  - 4.8.3 Target Virtual System *
  - 4.8.4 Host Virtual System *
  - 4.8.5 Team Programming Language Experience *
  - 4.8.6 Development Methods Experience *
  - 4.8.7 Development System Experience *
  - 4.8.8 Target System Experience *
- 4.9 Development Computer Manufacturer(s)
- 4.10 Target Computer Manufacturer(s)
- 4.11 Development Computer Model(s)
- 4.12 Target Computer Model(s)
- 4.13 Development Main Memory Size
- 4.14 Target Main Memory Size
- 4.15 Development Max Main Memory Size
- 4.16 Target Max Main Memory Size
- 4.17 Development CPU Processing Speed

- 4.18 Development Reserve Memory Required %
- 4.19 Target Reserve Memory Required %
- 4.20 Development Reserve Timing Required %
- 4.21 Target Reserve Timing Required %
- 4.22 Description of Differences Between Development and Target System
- 4.23 Software Environment
  - 4.23.1 Inherent Difficulty of Application *
  - 4.23.2 Turnaround Time to Compile *
  - 4.23.3 Terminal Responses *
  - 4.23.4 Development System Volatility *
  - 4.23.5 Specification Level *
  - 4.23.6 Quality Assurance Level *
  - 4.23.7 Test Level *
  - 4.23.8 Multiple Site Development *
  - 4.23.9 Resource Dedication *
  - 4.23.10 Resource/Support Location *
  - 4.23.11 Number of Shifts *
  - 4.23.12 Amount of Travel *
  - 4.23.13 Modern Practices Experience *
  - 4.23.14 Automated Tool Support *
- 4.24 Number of Errors at the conclusion of CSCI Testing
- 4.25 System Integration and Test Effort (% of Total)
- 4.26 Software Integration
- 4.27 New Hardware Design
- 4.28 Hardware Developed in Parallel with Software

5.0 Maintenance Information

- 5.1 Number of Years in Maintenance
- 5.2 Size
- 5.3 COTS
- 5.4 Schedule (MM/YY)
- 5.5 Number of Operational Sites
- 5.6 Documentation
- 5.7 Quality
- 5.8 Percent of Engineering Change Orders per Year
- 5.9 Effort
  - 5.9.1 Average Annual Software Maintenance
  - 5.9.2 Average Number of Hours per Person Month
  - 5.9.3 Annual Effort expended on Software Maintenance by Year(up to 30 years)

* NOTE: These elements listed in this breakdown are described by rating factors (ie. very low, low, nominal, high, and very high).

Appendix F: Adjusted Normalized Effort Calculations

Record #	Normalized Effort (manmonths)	Adjusted Normalized Effort (manmonths)
Military Ground/All Application Types		
0002	232	232(1.12) = 260
0004	84	84(1.12) = 94
0005	49	49(1.12) = 55
0006	66	66(1.12) = 74
0024	226	226(1.12) = 253
0028	841	841(1.12) = 942
0030	152	152(1.12) = 170
0048	326	326(1.12) = 365
0053	48	48(1.12) = 54
0054	127	127(1.12) = 142
0055	6363	6363(1.12) = 7127
0056	22	22(1.12) = 25
0058	606	606(1.12) = 679
0059	139	139(1.12) = 156
0061	107	107(1.12) = 120
0062	36	36(1.12) = 40
0063	22	22(1.12) = 25
0301	408	408(1.12) = 457
2497	76	76(1.12) = 85
2526	211	211(1.12) = 236
2527	235	235(1.12) = 263
2528	681	681(1.12) = 763
2610	458	458(1.12) = 513
2611	374	374(1.12) = 419
2612	312	312(1.12) = 349
Military Avionics		
0014	464	464(1.12) = 520
0346	654	654(1.12) = 732
Missile		
0036	480	480(1.12) = 538

Military Ground Signal Processing		
Record #	Normalized Effort (manmonths)	Adjusted Normalized Effort (manmonths)
0054	127	127(1.12) = 142
Unmanned Space-Ground/Command & Control		
0074	80	80(1.12) = 90
0075	912	912(1.12) = 1021
0076	115	115(1.12) = 129
0077	523	523(1.12) = 586
0078	478	478(1.12) = 535
0079	432	432(1.12) = 484
0080	296	296(1.12) = 332
0081	164	164(1.12) = 184
0082	140	140(1.12) = 157

NOTE: The percentage (7.2 for the Systems Test and Integration phase plus 4.8 for OT&E) used to adjust normalized effort was obtained from Table 5. Also, the Military Ground/MIS subset (record #'s 2526 through 2528 and 2610 through 2612) is listed with the Military Ground set.

Appendix G: MRE, MMRE and PRED(*l*) Detail

Unmanned Space-Ground/Command and Control
(Calibrated SLIM)

i	Record #	Actual	Estimate	MRE	PRED(.25)
1	82	157	67.29	.571	Fail
2	79	484	661.49	.367	Fail
3	75	1021	2102.61	1.059	Fail
				1.997	
MMRE = 1.997/3 = 0.666 PRED(.25) = #Pass/3 = 0/3 = 0					

Unmanned Space-Ground/Command and Control
(Uncalibrated SLIM)

i	Record #	Actual	Estimate	MRE	PRED(.25)
1	82	157	44.40	0.717	Fail
2	79	484	207.27	0.572	Fail
3	75	1021	433.78	0.575	Fail
				1.864	
MMRE = 1.864/3 = 0.621 PRED(.25) = #Pass/3 = 0/3 = 0					

Military Ground/MIS
(Calibrated SLIM)

i	Record #	Actual	Estimate	MRE	PRED(.25)
1	2526	236	192.70	.183	Pass
2	2527	263	221.94	.156	Pass
3	2528	763	1113.80	.460	Fail
4	2610	513	523.53	.021	Pass
5	2611	419	402.15	.040	Pass
6	2612	349	319.46	.085	Pass
				0.945	
MMRE = 0.945/6 = 0.157 PRED(.25) = #Pass/6 = 5/6 = .8333					

Military Ground/MIS
(Uncalibrated SLIM)

i	Record #	Actual	Estimate	MRE	PRED(.25)
1	2526	236	8.19	0.965	Fail
2	2527	263	9.33	0.965	Fail
3	2528	763	35.50	0.953	Fail
4	2610	513	19.18	0.963	Fail
5	2611	419	15.26	0.964	Fail
6	2612	349	12.70	0.964	Fail
				5.773	
MMRE = 5.773/6 = 0.962 PRED(.25) = #Pass/6 = 0/6 = 0					

Military Ground/All Application Types
(Calibrated SLIM)

i	Record #	Actual	Estimate	MRE	PRED(.25)
1	0002	260	263.52	0.014	Pass
2	0004	94	625.00	5.649	Fail
3	0005	55	71.54	0.301	Fail
4	0006	74	515.92	5.972	Fail
5	0024	253	83.23	0.671	Fail
6	0028	942	2191.73	1.327	Fail
7	0030	170	515.92	2.035	Fail
8	0048	365	2790.34	6.645	Fail
9	0053	54	127.19	1.355	Fail
10	0054	142	603.79	3.252	Fail
11	0055	7127	6433.76	0.097	Pass
12	0056	25	143.58	4.743	Fail
13	0058	679	251.69	0.629	Fail
14	0059	156	68.98	0.558	Fail
15	0061	120	1229.07	9.242	Fail
16	0062	40	74.01	0.850	Fail
17	0063	25	48.11	0.924	Fail
18	0301	457	1318.13	1.884	Fail
19	2497	89	39.09	0.561	Fail
20	2526	236	23.27	0.901	Fail
21	2527	263	26.81	0.898	Fail
22	2528	763	134.53	0.824	Fail
23	2610	513	63.23	0.877	Fail
24	2611	419	48.57	0.884	Fail
25	2612	349	38.59	0.889	Fail
				51.983	

MMRE = 51.983/25 = 2.166
PRED(.25) = #Pass/25 = 2/25 = .08

Appendix H: Wilcoxon Signed-Rank Test

Wilcoxon Test - Military Ground/All Application Types (Calibrated SLIM)

i	Record #	Actual	Estimate	Delta	Rank +	Rank -
1	0002	260	263.52	-3.52		1
2	0004	94	625.00	-531.00		19
3	0005	55	71.54	-16.54		2
4	0006	74	515.92	-441.92		16
5	0024	253	83.23	169.77	9	
6	0028	942	2191.73	-1249.73		24
7	0030	170	515.92	-345.92		13
8	0048	365	2790.34	-2425.34		25
9	0053	54	127.19	-73.19		6
10	0054	142	603.79	-461.79		18
11	0055	7127	6433.76	693.24	21	
12	0056	25	143.58	-118.58		8
13	0058	679	251.69	427.31	15	
14	0059	156	68.98	87.02	7	
15	0061	120	1229.07	-1109.07		23
16	0062	40	74.01	-34.01		4
17	0063	25	48.11	-23.11		3
18	0301	457	1318.13	-861.13		22
19	2497	89	39.09	49.91	5	
20	2526	236	23.27	212.73	10	
21	2527	263	26.81	236.19	11	
22	2528	763	134.53	628.47	20	
23	2610	513	63.23	449.77	17	
24	2611	419	48.57	370.43	14	
25	2612	349	38.59	310.41	12	
				SUMS:	141	184

H_o = center of the paired distributions = 0
Test Statistic: T = 141
n = 25
T_o at α = .90: T_o = 101 ---------- Since 141 ≥ 101, can not reject H_o
T_o at α = .95: T_o = 90 ---------- Since 141 ≥ 90, can not reject H_o

Wilcoxon Test - Military Ground/MIS (Calibrated SLIM)

i	Record #	Actual	Estimate	Delta	Rank +	Rank -
1	2526	236	192.70	43.30	5	
2	2527	263	221.94	41.06	4	
3	2528	763	1113.80	-350.80		6
4	2610	513	523.53	-10.53		1
5	2611	419	402.15	16.85	2	
6	2612	349	319.46	29.54	3	
				SUMS:	14	7

H_o = center of the paired distributions = 0
Test Statistic: $T = 7$
$n = 6$
T_o at $\alpha = .90$: $T_o = 2$ ---------- Since $7 \geq 2$, can not reject H_o
T_o at $\alpha = .95$: $T_o = 1$ ---------- Since $7 \geq 1$, can not reject H_o

Bibliography

1. Boehm, Barry W. Software Engineering Economics. Englewood Cliffs NJ: Prentice-Hall, 1981.

2. -----. "Software Engineering Economics," IEEE Transactions on Software Engineering, SE-10: 4-21 (January 1984).

3. Brooks, Frederick P., Jr. The Mythical Man-Month. Reading MA: Addison-Wesley Publishing Company, 1975.

4. Coggins, George A., and Roy C. Russell. Software Cost Estimating Models: A Comparative Study of What the Models Estimate. MS thesis, AFIT/GCA/LAS/93S-4. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1993 (AD-A275989).

5. Conte, Samuel D., H. E. Dunsmore, and V. Y. Shen. Software Engineering Metrics and Models. Menlo Park CA: The Benjamin/Cummings Publishing Company, Inc., 1986.

6. Daly, Bryan A. A Comparison of Software Schedule Estimators. MS thesis, AFIT/GCA/LSQ/90S-1. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1990 (AD-A229532).

7. DeMarco, Tom. Controlling Software Projects. Englewood Cliffs NJ: Prentice-Hall, 1982.

8. Devore, Jay L. Probability and Statistics for Engineering and the Sciences, Third Edition. Belmont CA: Wadsworth Publishing Company/Duxbury Press, 1991.

9. Fairley, Richard E. Software Engineering Concepts. New York: McGraw-Hill Book Company, 1985.

10. Ferens, Daniel V. Class handout, COST 677, Quantitative Management of Software. School of Systems and Logistics, Air Force Institute of Technology, Wright-Patterson AFB OH, Fall 1994.

11. Fisher, Gene H. "A Discussion of Uncertainty in Cost Analysis (A Lecture for the AFSC Cost Analysis Course)," Memorandum RM-3071-PR, Contract AF 49(638)-700. Santa Monica CA: The RAND Corporation, April 1962.

12. Genuchten, Michiel van, and Hans Koolen. "On the Use of Software Cost Models," Information and Management, 21: 37-44 (August 1991).

13. Goodson, James L. The Development of an Expert System for Software Cost Estimation. MS thesis, AFIT/GLM/LSM/90S-21. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1990 (AD-A229305).

14. Gulezian, Ronald. "Reformulating and Calibrating COCOMO," Journal of Systems and Software, 16: 235-242 (November 1991).

15. Londeix, Bernard. Cost Estimating for Software Development. New York: Addison-Wesley Publishing Company, 1987.

16. Management Consulting and Research. Air Force Cost Analysis Agency: Software Model Content Study. Contract F19628-88-D-OOO2, Task 51. Oxnard CA: Management Consulting and Research, 18 April 1994 (TR-9359/51-8).

17. Management Consulting and Research. Application Oriented Software Data Collection: Software Model Calibration Report. Contract F33657-87-D-0107. Oxnard CA: Management Consulting and Research, 15 March 1991 (TR-9007/49-1).

18. Management Consulting and Research. Space and Missile Systems Center Software Database User's Manual: Version 1.0. Oxnard CA: Management Consulting and Research, 1994.

19. Management Consulting and Research. Space and Missile Systems Center Software Development Database (Phase 5). Contract F04701-90-D-0002. Oxnard CA: Management Consulting and Research, 1 November 1994 (TR-9338/025-02).

20. Mendenhall, William, Dennis D. Wackerly, and Richard L. Scheaffer. Mathematical Statistics with Applications, Fourth Edition. Belmont CA: Wadsworth Publishing Company/Duxbury Press, 1990.

21. Miyazaki, Y., H. Nozaki, K. Ozaki, and M. Terakado. "Robust Regression for Developing Software Estimation Models," Journal of Systems and Software, 27: 3-16 (October 1994).

22. Myers, Ware, and Lawrence H. Putnam. Measures for Excellence: Reliable Software on Time, Within Budget. Englewood Cliffs NJ: Prentice-Hall, 1992.

23. Ourada, Gerald L. Software Cost Estimating Models: A Calibration, Evaluation, and Comparison. MS thesis, AFIT/GSS/LSY/91D-11. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1991 (AD-A246677).

24. Quantitative Software Management, Inc. SLIM 3.1 for Windows - Software. McLean VA: Quantitative Software Management, Inc., 1993.

25. Quantitative Software Management, Inc. SLIM 3.1 for Windows User's Manual. McLean VA: Quantitative Software Management, Inc., 1993.

26. Space and Missile Systems Center. Software Database Data Collection Dictionary. 1993.

27. Stukes, Sherry. Senior Associate for Management Consulting and Research, Oxnard CA. Personal Interview. 7 April 1995.

28. Stukes, Sherry. Senior Associate for Management Consulting and Research, Oxnard CA. Personal Interview. 26 June 1995.

29. Stukes, Sherry. Senior Associate for Management Consulting and Research, Oxnard CA. Personal Interview. 4 August 1995.

30. Thibodeau, Robert. An Evaluation of Software Cost Estimating Models. Contract F30602-79-C-0244. Huntsville AL: General Research Corporation, 10 April 1981 (TR10-2670).

31. Wellman, Frank. Software Costing. New York: Prentice-Hall, 1992.

VITA

Captain Robert K. Kressin was born on 9 January 1960 on Ramey Air Force Base, Puerto Rico. He received a Bachelor of Business Administration from the University of Texas at San Antonio on 16 May 1987. He received his commission on 17 May 1987 through the Reserve Officer Training Corps at the University of Texas at San Antonio. He first served as a Deputy Accounting and Finance Officer for the 7100 Combat Support Wing at Lindsey Air Station, Germany. He was then assigned as an Accounting and Finance Officer to the 7275th Air Base Group at San Vito dei Normanni Air Station, Italy. He then served as an Accounting and Finance Officer for the 47th Flying Training Wing at Laughlin AFB, TX. In May 1994, he entered the School of Logistics and Acquisition Management, Air Force Institute of Technology.

Permanent Address: 14706 Hermes
Selma, TX 78154

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1995	Master's Thesis

4. TITLE AND SUBTITLE
CALIBRATION OF THE SOFTWARE LIFE CYCLE MODEL (SLIM) TO THE SPACE AND MISSILE SYSTEMS CENTER (SMC) SOFTWARE DATABASE (SWDB)

5. FUNDING NUMBERS

6. AUTHOR(S)

Robert K. Kressin, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCA/LAS/95S-6

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

SMC/FMC
2430 E. El Segundo Blvd. #2010
Segundo, CA 90245-4687

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This study focused on calibrating the SLIM cost model to sets of historic software projects that were described along a specific development environment, and assessing the predictive ability of the model resulting from this calibration. The SMC database was first stratified along specific operating environment and application parameters, as well as some key SLIM input parameters (size, effort, development schedule, and minimum peak staffing). Stratification of the SMC database resulted in three data subsets, two of which were defined along a specific operating environment and application type, and one which was only defined along a specific operating environment. Once SLIM was calibrated to each data set, the calibrated SLIM models were validated in an effort to verify their predictive ability. Validation was accomplished using MRE, MMRE, percentage method, and standard deviation measures. The Wilcoxon signed-rank test was also accomplished to test for randomness of the estimates from the calibrated models. The author concluded that successfully calibrating SLIM depends largely on the user's ability to calibrate a consistent PI and MBI across a set of historic software projects. PI and MBI consistency, as well as model predictive ability, was best for data sets that were stratified along a specified application type.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Cost Estimates, Cost Models, Computers, Software Cost Models	107
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited