The impacts of the handoffs on software development: A cost estimation model

Michael Jay Douglas
University of South Florida

Follow this and additional works at: http://scholarcommons.usf.edu/etd
Part of the American Studies Commons

Scholar Commons Citation
http://scholarcommons.usf.edu/etd/2507

This Dissertation is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.
The Impacts of the Handoffs on Software Development: A Cost Estimation Model

by

Michael Jay Douglas

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy
Department of Information Systems and Decision Sciences College of Business Administration University of South Florida

Co-Major Professor: Alan R. Hevner, Ph.D.
Co-Major Professor: Rosann Webb Collins, Ph.D.
Anol Bhattacherjee, Ph.D.
Kaushal Chari, Ph.D.

Date of Approval:
May 8, 2006

Keywords: cocomo ii, pestimate, experiment, design science

© Copyright 2006, Michael Jay Douglas
TABLE OF CONTENTS

LIST OF EQUATIONS ....................................................................................................................... VI

LIST OF TABLES .............................................................................................................................. VII

LIST OF FIGURES .......................................................................................................................... IX

ABSTRACT ........................................................................................................................................ X

CHAPTER 1 INTRODUCTION ........................................................................................................ 1

1.1 INTRODUCTION ....................................................................................................................... 1

1.2 SOFTWARE COST ESTIMATION DIFFICULTIES ................................................................. 3

1.3 SOFTWARE COST ESTIMATION MODELS HELP ............................................................... 5

1.4 ATTRIBUTES OF A GOOD MODEL ...................................................................................... 6

1.5 THE SOFTWARE HANDOFF .................................................................................................. 9

1.6 THE SOFTWARE HANDOFF AND TEAM SIZE ................................................................. 11

1.7 SOFTWARE HANDOFF AND PROCESS STRUCTURE ....................................................... 12

1.8 INTER-GROUP COORDINATION ....................................................................................... 15

1.9 RESEARCH QUESTIONS .................................................................................................... 16

1.10 NEW SOFTWARE COST ESTIMATION MODEL .............................................................. 17

1.11 RESEARCH PARADIGM .................................................................................................... 18

1.12 CONTRIBUTIONS .............................................................................................................. 21

1.13 DISSERTATION FORMAT .................................................................................................. 21

CHAPTER 2 LITERATURE REVIEW .......................................................................................... 23

2.1 INTRODUCTION ................................................................................................................... 23

2.2 COST ESTIMATION NEEDS ............................................................................................ 26

2.3 COST ESTIMATION SOLUTIONS ..................................................................................... 28

2.4 EMPIRICAL MODEL BUILDING ......................................................................................... 39

2.5 COCOMO ........................................................................................................................... 40
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4 Using the COCOMO II Outputs</td>
<td>75</td>
</tr>
<tr>
<td>5.5 Modeling the Work Breakdown Structure in Process Structures</td>
<td>79</td>
</tr>
<tr>
<td>5.6 Mapping of the Three-Tier Process Structure</td>
<td>82</td>
</tr>
<tr>
<td>5.7 Mapping of the Two-Tier Process Structure</td>
<td>85</td>
</tr>
<tr>
<td>5.8 Mapping of the One-Tier Process Structure</td>
<td>86</td>
</tr>
<tr>
<td>5.9 Populating Staffing into the Process Structures</td>
<td>87</td>
</tr>
<tr>
<td>5.10 Effort Calculation</td>
<td>91</td>
</tr>
<tr>
<td>5.11 Three-Tier Structure</td>
<td>91</td>
</tr>
<tr>
<td>5.12 Two-Tier Structure</td>
<td>99</td>
</tr>
<tr>
<td>5.13 One-Tier Structure</td>
<td>104</td>
</tr>
<tr>
<td>5.14 Staff Loading</td>
<td>105</td>
</tr>
<tr>
<td>5.15 Optimization</td>
<td>105</td>
</tr>
<tr>
<td>5.16 Conclusion</td>
<td>107</td>
</tr>
<tr>
<td>Chapter 6 Decision Support Tool</td>
<td>108</td>
</tr>
<tr>
<td>6.1 Example Test Run</td>
<td>108</td>
</tr>
<tr>
<td>6.2 Tool Discussion</td>
<td>110</td>
</tr>
<tr>
<td>6.3 Tool Construction</td>
<td>113</td>
</tr>
<tr>
<td>Chapter 7 Experimental Validation</td>
<td>115</td>
</tr>
<tr>
<td>7.1 Introduction</td>
<td>115</td>
</tr>
<tr>
<td>7.2 Study Rationale</td>
<td>116</td>
</tr>
<tr>
<td>7.3 Institutional Review</td>
<td>118</td>
</tr>
<tr>
<td>7.4 Research Question</td>
<td>118</td>
</tr>
<tr>
<td>7.5 Hypotheses</td>
<td>119</td>
</tr>
<tr>
<td>7.6 Pretest</td>
<td>120</td>
</tr>
<tr>
<td>7.7 Pilot Test</td>
<td>121</td>
</tr>
<tr>
<td>7.8 Main Study</td>
<td>121</td>
</tr>
</tbody>
</table>
LIST OF EQUATIONS

EQUATION 2.1 EFFORT EQUATION .................................................................32
EQUATION 2.2 SIZE EQUATION .................................................................32
EQUATION 2.3 BASIC EFFORT EQUATION .................................................39
EQUATION 2.4 BASIC COCOMO EQUATION ..............................................40
EQUATION 2.5 COCOMO EFFORT EQUATION ..........................................41
EQUATION 2.6 INTERMEDIATE COCOMO EFFORT EQUATION ...............42
EQUATION 3.1 ECONOMY OF SCALE EQUATION .....................................54
EQUATION 3.2 NOMINAL EFFORT ..............................................................63
EQUATION 3.3 SCHEDULE ESTIMATION ...................................................63
EQUATION 3.4 STAFFING EQUATION .......................................................64
EQUATION 4.1 COMMUNICATION PATHS FOR N PEOPLE .......................66
EQUATION 4.2 PREDICTION EQUATION FOR COMMUNICATION OVERHEAD ........................................68
EQUATION 4.3 COPMO EQUATION ............................................................69
EQUATION 5.1 COMMUNICATION PATHS FOR N PEOPLE .......................92
EQUATION 5.2 PREDICTION EQUATION FOR COMMUNICATION OVERHEAD ........................................92
EQUATION 5.3 EFFORT MULTIPLIERS DUE TO INTRA-GROUP COMMUNICATION ........................................94
EQUATION 5.4 TIER-THREE EFFORT MAPPING EQUATIONS ....................95
LIST OF TABLES

TABLE 2-1 BASIC COCOMO CONSTANTS .................................................................................................41
TABLE 3-1 UNADJUSTED FP TO SLOC CONVERSION RATIOS .......................................................................52
TABLE 3-2 SCALE FACTORS .........................................................................................................................54
TABLE 3-3 POST-ARCHITECTURE EFFORT MULTIPLIERS ........................................................................56
TABLE 3-4 EARLY DESIGN EFFORT MULTIPLIERS .......................................................................................57
TABLE 3-5 PLANS AND REQUIREMENTS ACTIVITY DISTRIBUTION ...........................................................59
TABLE 3-6 PRODUCT DESIGN ACTIVITY DISTRIBUTION ...........................................................................59
TABLE 3-7 PROGRAMMING ACTIVITY DISTRIBUTION .............................................................................60
TABLE 3-8 INTEGRATION AND TEST ACTIVITY DISTRIBUTION ..................................................................60
TABLE 3-9 WORK BREAKDOWN STRUCTURE FOR A MEDIUM SIZE PROJECT ...........................................60
TABLE 3-10 SOFTWARE COST ESTIMATION MODEL TYPES ......................................................................62
TABLE 4-1 COMMUNICATION OVERHEAD PERCENTAGE AS A GIVEN TEAM SIZE .....................................68
TABLE 4-2 COMMUNICATION PATHS ADDED TO COMMUNICATION OVERHEAD ...........................................68
TABLE 4-3 COPMO AND COMMUNICATION OVERHEAD ...........................................................................69
TABLE 5-1 PLANS AND REQUIREMENTS ACTIVITY DISTRIBUTION ...........................................................76
TABLE 5-2 PRODUCT DESIGN ACTIVITY DISTRIBUTION ...........................................................................76
TABLE 5-3 PROGRAMMING ACTIVITY DISTRIBUTION ..............................................................................77
TABLE 5-4 INTEGRATION AND TEST ACTIVITY DISTRIBUTION ..................................................................77
TABLE 5-5 PLANS AND REQUIREMENTS PHASE FOR A 40 KSLOC PROJECT ...........................................78
TABLE 5-6 COMPLETE WORK BREAKDOWN STRUCTURE FOR EXTENDED EXAMPLE ..............................78
TABLE 5-7 WORK BREAKDOWN STRUCTURE MAPPING .............................................................................79
TABLE 5-8 ADJUSTED WORK BREAKDOWN STRUCTURE ...........................................................................81
TABLE 5-9 EXAMPLE TEAM SIZES ...........................................................................................................91
TABLE 7-1 RANDOMIZING TO TREATMENTS ............................................................................................117
TABLE 7-2 RESEARCH MODEL ...............................................................................................................119
TABLE 7-3 EXPERIMENTAL TASKS OVERVIEW ......................................................................................126
LIST OF FIGURES

FIGURE 1-1 TYPICAL PROJECT RESOLUTIONS ................................................................. 2
FIGURE 1-2 THREE-TIER MODEL ................................................................. 13
FIGURE 1-3 TWO-TIER MODEL ................................................................. 14
FIGURE 1-4 ONE-TIER MODEL ................................................................. 14
FIGURE 1-5 RESEARCH MODEL ................................................................. 17
FIGURE 1-6 NEW SOFTWARE COST ESTIMATION MODEL ........................................... 18
FIGURE 1-7 DESIGN SCIENCE RESEARCH MODEL ............................................... 20
FIGURE 5-1 MODEL OVERVIEW ................................................................. 74
FIGURE 5-2 EFFORT BREAKDOWN FOR THREE-TIER ........................................... 83
FIGURE 5-3 TWO-TIER EFFORT BREAKDOWN ........................................... 86
FIGURE 5-4 ONE-TIER PROCESS STRUCTURE ........................................... 87
FIGURE 5-5 THREE-TIER MODEL ................................................................. 88
FIGURE 5-6 TWO-TIER MODEL ................................................................. 89
FIGURE 5-7 ONE-TIER MODEL ................................................................. 90
FIGURE 5-8 EFFORT BREAKDOWN FOR THREE-TIER ........................................... 93
FIGURE 5-9 TWO-TIER EFFORT BREAKDOWN ........................................... 100
FIGURE 6-1 SCREENSHOT OF ESTIMATING SOFTWARE SIZE .................................. 111
FIGURE 6-2: SCREENSHOT OF DEVELOPED TOOL - SIMULATION RESULTS ............ 112
FIGURE 6-3: SCREENSHOT OF DEVELOPED TOOL - AFTER OPTIMIZATION ............ 113
FIGURE 7-1 DESIGN SCIENCE RESEARCH MODEL ............................................... 115
FIGURE 8-1 EMpirICAL RESEARCH MODEL ........................................................ 128
The Impacts of the Handoffs on Software Development:

A Cost Estimation Model

Michael Jay Douglas

ABSTRACT

Effective software cost estimation is one of the most challenging and important activities in software development. The software industry does not estimate projects well. Poor estimation leads to poor project planning with resulting schedule overruns, inadequate staffing, low system quality, and many aborted projects. Research on software estimation is needed to build more accurate models of the key aspects of software development. The goals of research in this dissertation are to investigate and improve the modeling of team size and project structures in current software estimation methods.

Mathematical models for estimating the impacts of project team size and three variations of project structure are developed. These models accept the outputs of the COCOMO II software estimation tool, allow variation in both team size and project structure, and produce more detailed project estimates. This new extended model of COCOMO II is implemented in a decision support tool for software estimators called PSEstimate.
Following the design science research paradigm, the artifact is evaluated with an experiment with experienced software project managers. Three treatment groups: a manual (no tool) group, a COCOMO II group, and a PSEstimate group, completed two multipart software cost estimation tasks. The accuracy and consistency of the cost and schedule estimates, the participants’ confidence in their estimates, and their satisfaction with and perceived usefulness of the cost estimation tool are measured.

The experimental results support most of the hypotheses of the dissertation. For most tasks, individuals aided by computer-based decision support tools produce more accurate project effort estimates and are more confident in their estimates than manual estimators. There are no significant differences between the three groups on schedule estimation. A possible explanation is that experienced estimators in the manual group compensate for the inaccuracy of their effort estimates by adding time to their schedule estimates.

The research contributions are new mathematical models for software estimation based on project team size and structure; a decision support tool (PSEstimate) that incorporates these models; and the experimental results that demonstrate improvements in software estimation by experienced project managers when the new models and tool are applied in practice.
CHAPTER 1 INTRODUCTION

When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the stage of science.

- William Thomson (Lord Kelvin), 1891

1.1 Introduction

Software cost estimation remains an important unsolved practical problem in software engineering (Lewis 2001). Software cost estimation has failed, in most cases, to accurately predict the actual costs or the time needed to develop the system (Vijayakumar 1997). Project managers have the responsibility to make accurate estimations of cost and effort, but without good software cost estimation tools, the effectiveness of software project management is reduced (Agarwal, Kumar et al. 2001). A good software cost estimation model can significantly help software project managers make informed decisions on how to manage resources, control and plan a project, and deliver a project on time, on schedule, and on budget (Chen, Menzies et al. 2005). The problems in estimation are exacerbated by continued changes in software technologies. Thus, software cost estimation models require constant modification to stay current (Jones 2002). Further research in software cost estimation is clearly needed.

In the United States, more than 250 billion dollars is spent each year on IT application development (The Standish Group 2003), but in 1994, only 16.2% of
software development projects were completed both on-time and on-budget (Standish Group Inc. 1994). Almost ten years later, only 32% of projects are successful (The Standish Group 2003). Some companies can expect that a typical software development project will be delivered a year late at double the budget (Paulk 1995). Figure 1-1 illustrates typical project resolutions, and highlights how late many projects are delivered. 22% of the projects in this data set took more than twice as long to complete than was originally expected.

![Figure 1-1 Typical Project Resolutions](McConnell 2000)

Poor project planning and management results in companies taking a collective loss of $80 billion annually on new software projects that eventually get cancelled (King 1997). Cancelled projects are especially problematic given that projects are commonly cancelled in the later stages of development after significant resources have been
expended on behalf on the project. The average cancelled project in the United States is about a year behind schedule and has consumed 200 percent of its expected budgeted by the time it has been cancelled (Jones 1993).

A goal of project managers is to guide to completion a software development project. A successful software development project will deliver to the users a system desired by the customers. If the people that financially support the software development and the users of the software are satisfied, the system can be considered successful. Part of the desired system could be parameters such as: total system development cost, scheduled delivery date, functionality, and quality. The project manager needs to supervise the software development project so that the desired system is delivered. Reasonable estimates of cost, schedule, and staff are critical guides that help the project manager successfully control the software development activities.

1.2 Software Cost Estimation Difficulties

Estimating software development costs with accuracy is very difficult. The most common approach for improving software cost estimates is to use empirical models (Mukhopadhyay, Vicinanza et al. 1992). The predictive accuracy of software cost estimation models is not satisfactory, since model-based estimates are generally within 25% of the actual cost or schedule, one-half of the time (Ferens and Christensen 2000). This means that more than one-half of estimates are off by more than 25 percent, when comparing the actual versus estimated metric. When poor results are found using software cost estimation models, many researchers suggest calibrating the parameters of a model to a specific environment (Kemerer 1987; van Genuchten and Koolen 1991;
Andolfi 1996). However, results from calibrating software cost estimation models show that the predictive accuracy does not always improve (Ferens and Christensen 1998).

The additional goal of being able to predict the costs and schedule at the beginning of the project can prove to be more challenging. “Early prediction of completion time is absolutely essential for proper advance planning and aversion of the possible ruin of a project” (Pillai and Nair 1997 p.485). Nevertheless, using the entire suite of available software cost estimation models, researchers find that there is no evidence that software models are effective at estimating projects at an early stage of system development (Vijayakumar 2002). Yet, estimation does not stop because it is inaccurate. Instead of using a model, software cost estimation continues to be most commonly conducted by experts, sometimes using a Bayesian approach to manage the uncertainty (Stamelos, Angelis et al. 2003).

McConnell suggests there is a lack of understanding as to what developing software means. The difficulty in creating good software cost estimation models is directly related to the lack of understanding about software development.

The problems in developing today’s and tomorrow’s systems are overwhelming; they require many different types of problems to be solved. No other scientific or engineering discipline relies on a single technique for addressing problems, so why are we, so-called professional engineers (and computer scientists), stupid enough to think that our field is fundamentally different in this respect? So, what do we need to do? First, industrial management has to understand that software engineering is not an engineering discipline like so many others (yet) and that standards, methods, and tools are all likely to be wrong (once we really understand what developing software means) (McConnell 2000 p.17).
The current software cost estimation tools have not yet reached a level of accuracy required for proper advanced planning. Research is needed to improve the understanding of software development and then use that knowledge gained to create better software cost estimation models.

1.3 Software Cost Estimation Models Help

A software cost estimation model provides a formal method for estimating software costs and schedule. Because of the lack of predictive validity, some project managers believe that using formal methods to estimate software costs is a wasted effort and instead use intuitive judgment (Paulk 1995) or external sources, such as senior project managers desires for cost estimates (Agarwal, Kumar et al. 2001). Senior management needs are not usually based on the capabilities of the development staff. These needs are therefore subject to schedule and budget overruns.

Even with the predictive problems of software cost estimation models, the models prove to be better than any alternative method of estimation. For example, simple statistical models have been shown to be superior to using human judgment even though the statistical models were created by the humans (Paulk 1995). Consistent answers when given the same input are one reason there is an advantage of using models over humans. An incomplete model is better than no model at all; therefore, research is conducted to improve models rather than using other methods of estimation, such as expert opinion.
1.4 Attributes of a Good Model

There are three requirements for a software cost estimation model that will make accurate predictions of software effort and schedule. The first requirement is that the estimation model is built on a solid foundation of prior research and empirically tested. Software cost estimation models have two problems. First, “The domain of software effort estimation lacks a strong causal model based on deep principles and is situated within an often-changing, highly context-dependent task environment” (Mukhopadhyay, Vicinanza et al. 1992 p.156). Second, attempts at validating software cost estimation models have been largely unsuccessful (Mukhopadhyay, Vicinanza et al. 1992).

Since there is a lack of theoretical support describing the complicated process of how software development impacts software development costs, using historical data as a basis for software cost estimation is very insightful. Having an organization collect data during software development is the first step in trying to improve estimates. Boeing Information Systems used historical data and drastically increased the quality of software estimates. Without historical data, the variance in effort ranged from -145% to +20%, whereas with historical data the variance was reduced to -20% to +20% (Vu 1997). Boeing Information Systems still encountered cost overruns, but moving from 145% to only 20% was a big improvement. By measuring and documenting the software development process, future estimates are based on empirical data rather than pure speculation.

The second requirement of a good estimation model is that the development process follows a repeatable process. A software development organization that follows a
repeatable process is more mature because a higher amount of discipline is instilled into software development activities. The maturity of an organizations software process influences its ability to meet costs, quality, and schedule target (Curtis 1992). In 1994, 75% of all software organizations did not use a disciplined approach to development software. “The immature software organization is reactionary and managers are usually focused on fighting the fires that a more mature process might have prevented” (Curtis 1992 p.2). Having project managers react to contingencies, rather than planning and controlling the project, only makes project planning more difficult. Research has found that the inability to estimate software development accurately is the fault of an immature organization (Curtis 1992). The best predictor of cost in an immature organization is the capability of the staff (Paulk 1995). Heroic efforts by an individual are needed in order for an immature organization to deliver projects within planned targets. Software cost estimation models have limited use in immature organizations. However, the value of software cost estimation models increases as the organization becomes more mature. For that reason, it is not surprising that most high maturity companies use cost models for their software cost estimation (Paulk, Goldenson et al. 2000).

The most common method available to project managers for increasing the quality of the organization’s software development processes is to use the Capability Maturity Model. The Carnegie Mellon Software Engineering Institute’s Capability Maturity Model (1995) (CMM) is a framework for improving the software development process based on the concepts of Total Quality Management and continuous improvement. Research has shown that the predictability, control, and the effectiveness
of the processes are significantly improved by adopting the CMM (Humphrey, Snyder et al. 1991; Lipke and Butler 1992; Dion 1993; Paulk, Weber et al. 1993). By adopting key process areas, software development processes mature, allowing for an improvement in software development.

Another model of software process quality improvement is ISO 9001. ISO 9001 was created at the same time the CMM was created in 1987. The US Department of Defense sponsored the CMM where as the International Organization for Standardization in Geneva, Switzerland created the ISO 9001 model. ISO 9001 and more specifically ISO-9000-3, which governs the software development process, are commonly needed by businesses that want to develop and sell software in the European Union. Both the CMM and ISO 9001 embody the philosophy, “To estimate the time and cost of next time, you must know and be able to repeat what you did last time” (Putnam and Myers 1997 p.105).

On August 11, 2000, a new process model, the CMMI-SE/SW Version 1.0, officially replaced the CMM. The Capability Maturity Model Integration (CMMI) was created to support process and product improvement, and to reduce redundancy and eliminate inconsistency experienced by those using multiple standalone models. The CMMI combines all relevant process models into one product suite.

The ISO 9001:2000 standard makes obsolete the preceding ISO 9001 standards. Organizations that are ISO 9001 compliant have to update their quality system and be recertified at the new ISO 9001:2000 standard to conduct business in the European Union. The continual improvement of process models highlights the importance of
having a repeatable process. With the continual improvement of process models, software development estimation can advance.

A third method of process improvement that can be applied to software is Six Sigma. A process that has achieved Six Sigma will produce no more than 3.4 defects per million opportunities (Harry and Lawson 1992).

The third requirement for a good estimation model is that the model includes relevant factors that vary with project metrics. This dissertation argues that two relevant factors, process structure and inter-group coordination are missing from current software cost estimation models.

1.5 The Software Handoff

To advance software cost estimation, models must include one major activity of software development, the software handoff. A software handoff can explain differences in inter-group coordination between different process models. The software handoff is introduced and this dissertation will explain how the software handoff affects software development.

A software handoff occurs when one person or group’s software-development-lifecycle-work-product output is given to another person or group as input to another work-product. Examples of a software handoff include the analysts’ requirements document being given to the designers, the designers’ system design being given to the programmers, and the programmers’ code being given to the tester. Unless one person comes up with an idea for a system, creates the requirements, designs the system, implements the design, tests the code, and uses the final system, a software handoff will
occur. The term handoff invokes an analogy to both football and air traffic control. When an airplane moves from one controller to another, it is “handed off” to the next responsible controller. The term handoff is also used in wireless networking terminology when one call moves from one cell tower to another cell tower because of movement in the wireless device. With a software handoff, an artifact moves from one person or group to another.

The software handoff creates a potential communication problem in software development. A software handoff can be thought of as an information flow. “It is clear that information flow impacts productivity (because developers spend time communicating) as well as quality (because developers need information from one another in order to carry out their tasks well)” (Seaman and Basili 1997 p.550). “Communication problems occurred in the transition between phases when groups transferred intermediate work products to succeeding groups” (Curtis, Krasner et al. 1998 p.1281). The software handoff is one of the culprits of communication problems during software development.

The software handoff is a process loss and leads to inefficiency, but software handoffs can be anticipated during development and can be managed. Properly managing the handoff will increase efficiency. The effects of the software handoff are most commonly seen in integration testing when rework is needed to fix misunderstandings caused by communication problems during development. Since the handoff is required for all large systems, proper management is required. Software handoffs have different magnitudes. Handing off 100,000 lines of code is a large handoff compared to handing
off only 1000 lines of code. Some software development processes, such as a project that has many different specialized groups all working together, have more handoffs than other processes. The number of handoffs in a project can be controlled by the way the project team is structured. If an analyst does both requirements definition and design, this eliminates the handoff of the requirements document to the design group.

Software handoffs are unavoidable during software development. Any software development process that requires coordination between groups is going to have software handoffs. More interfaces mean more software handoffs. Bigger software development projects are going to have bigger software handoffs. The amount of information that must be communicated in the handoff is another aspect of the software handoff.

Different software development projects are going to need different process structures based on the size of the project, the number of people working on development, and the amount of schedule time to complete development. Creating an order entry website will probably not need the same process structure that a large military project needs for system development.

1.6 The Software Handoff and Team Size

Up to a point, a larger team allows more work to be done in a given amount of time. However, as teams get larger, the complexity of the software handoff grows. At some point, creating a bigger team will no longer be efficient. There exists an equilibrium point which maximizes the efficiency of the work to be done.

The team size of a project group will affect the software handoff. Twenty people handing over an artifact to twenty other people is different from one person handing an
artifact to one other person, even if the artifacts are the same. Splitting up development tasks between more teams requires more handoffs, but the handoffs are smaller. For every software development project, the process of development will dictate a process structure, and the process structure will dictate the number of handoffs.

1.7 Software Handoff and Process Structure

“A software group should have between five and eight members. The overall design should be portioned into successively smaller chunks, until the development group has a chunk of software to develop that minimizes intra-group and inter-group communications. The chunks should be designed to hide difficult design decisions” (Simmons 1991 p.461).

Since Simmons suggests separating difficult design decisions, the V-Model of software development is used to partition the activities of software development into different groups. This dissertation details three different structures based on the V-Model with each structure having different amounts of partitioning.

This dissertation will study the impact of the software handoff on three different types of software development process structures. The first structure is a Three-Tiered model as shown in Figure 1-2. The boxes in the figure represent different development groups. In the three-tiered group, there are requirements, design, implementation/unit test, integration test, and customer acceptance groups. Each group will have a variable number of team participants with the minimum number being one.
Figure 1-3 shows a Two-Tier model. In this model, the requirements and design teams are combined to form the analysis and design team. The customer acceptance and integration test teams are also combined to form the integration/customer acceptance team. Reducing from five to three groups allow for a reduction of software handoffs.
Figure 1-3 Two-Tier Model

Figure 1-4 shows a One-Tier model. In this model, all system development activities take place in one group. There is no formal software handoff, but very little process to organize complexity. Also, for large groups, communication costs are higher in the One-Tier model with the same number of staff as compared to the other groups.

Figure 1-4 One-Tier Model
1.8 Inter-group Coordination

Inter-group coordination is a CMM Level 3 key process area. According to the CMM (Paulk, Weber et al. 1993), inter-group coordination is used to establish a means for the software engineering group to participate actively with other engineering groups so the project is better able to satisfy the customer’s needs effectively and efficiently. Examples of engineering groups that need to be coordinated with customers and end-users are: software engineering, software estimating, system test, software quality assurance, software configuration management, contract management, and documentation support. Communication problems during software development should be addressed by inter-group coordination. Inter-group coordination includes the technical working interfaces and interactions between groups. The software handoff is a way to understand inter-group coordination. Inter-group coordination is planned and managed to ensure that quality and integrity exists throughout the entire software development process.

To have satisfied the requirements of the inter-group coordination key process, measurement must be made to the system under development to ensure proper inter-group coordination. Examples of measurement activities include: measuring the actual effort and the resources expanded by the software engineering group for support to other engineering groups, and measuring the actual effort and other resources expanded by the other engineering groups in support of the software engineering groups.

One example of an inter-group coordination activity includes when representatives of the project engineering groups conduct periodic reviews and
interchanges. These interchanges are software handoffs. By studying software handoffs, more knowledge about software development can be understood.

1.9 Research Questions

This dissertation is based on three research questions. The research questions guide this dissertation through the nine chapters.

Research Question 1: Can a software cost estimation model be built that reflects the effect of both inter-group coordination and intra-group communication?

Research Question 2: Can a software cost estimation tool be built for project managers that implements inter-group coordination, intra-group communication and process structure?

Research Question 3: Does an experiment demonstrate the effectiveness of the new software cost estimation model?

Figure 1-5 displays the research model used in this dissertation. The research model is derived from the previous research questions. Three different types of relevant factors will be studied. A baseline where no support is given is the first type of model support. The second type is allowing for project size support. The third is a model that provides support for inter-group coordination and software handoffs. The experimental effectiveness of the estimation model is measured by five variables. These variables are accuracy, consistency, confidence, satisfaction and perceived usefulness.
Method of Estimation

- No Model
- State-of-the-practice model (COCOMO II)
  State-of-the-practice model that includes the effects of inter-group coordination and intra-group communication

Accuracy of Software Estimate
Consistency of Software Estimates
Confidence of Software Estimates
Satisfaction of Estimation Technique
Perceived Usefulness of Estimation Technique

Figure 1-5 Research Model

1.10 New Software Cost Estimation Model

Figure 1-6 shows the new software cost estimation model that has been developed. To estimate a project some basic information about a project is needed. These project characteristics focus on describing the size of the project. The first characteristic is system size. In addition, any factor that can make the project easier or harder to conduct also needs to be quantified. The effort multipliers and scale factors are the methods in this model to quantify the different difficulties.
The project characteristics are then entered into the COCOMO II algorithm. COCOMO II returns an effort estimate in man-months, a schedule estimate in months, and a detailed work breakdown structure that will quantify how much effort each particular software development activity will need. Next, different process structures with configurable team sizes are used to come up with a modified effort and schedule estimate. A new measure, staff loading, was also created. This measure represents the percentage of time that the groups in the two-tier and three-tier processes are assigned to tasks.

Figure 1-6 New Software Cost Estimation Model

1.11 Research Paradigm

Information Systems research can be broken down into two complementary paradigms. The first paradigm is behavioral science. A goal of the behavioral science
paradigm is to develop and verify theories of individual and organizational behavior. The behavioral science paradigm follows a natural science orientation where researchers measure the naturally-occurring or evoked behavior of individuals, groups of people, and organizations. Individuals or groups of individuals working to form an organizational unit together are typically the unit of analysis in the behavioral science paradigm. Managerial and organizational issues are studied in this paradigm.

Managerial and organizational issues are important, however the technological aspects of IS are equally as important. The behavioral science paradigm does not work well when applied to technological aspect of IS. For example, an efficient way to store and retrieve data does not occur in nature. A researcher can not just study individuals to extract methods to efficiently retrieve data. A different approach is needed.

The second paradigm in Information Systems is the design science paradigm. The design science paradigm stresses “design” as an approach to create knowledge. The late Herbert Simon’s *The Sciences of the Artificial (1969)* explained the importance of Design. Studying artificial objects (man-made) rather than natural objects or phenomenon can solve many problems that a behavioral approach cannot. For example, instead of studying individuals to find a way to efficiently store and retrieve data, designing a system will produce much more knowledge. In design science research, the artifact is important. In Information Systems, modeling, building, designing, and implementing an artifact can create knowledge.

Figure 1-7 shows the design science paradigm model that this dissertation is based upon.
This research is conducted under the design science paradigm. The design science paradigm has two different fundamental goals. The first goal is the construction phase, where artifacts are produced to solve a specific problem. The second goal is the evaluation phase, where the produced artifacts are evaluated. A project management tool is developed in the construction phase. The tool instantiates the research model depicted in Figure 1-6. During the construction phase, rigor is applied by using prior research and tools. Relevance is applied in the construction phase by using current problems that organizations have with current software cost estimation models.

The evaluation phase is conducted by testing the developed project management tool. An experimental design is used to show that the model developed improves estimation in a laboratory setting.
1.12 Contributions

Improving the process of developing software not only makes the organization more mature, but also can lead to cost savings (Fenton 1993). By introducing the concept of a software handoff and its effect in software development, better processes can be devised. Utilizing better processes will lead to more mature organizations. By building a cost estimation model that includes process structure and team size, better estimates can be used by software cost estimators.

For software development managers, the new software cost estimation model provides a better model than any currently available. The project management tool that implements the new software cost estimation model can be used to support improved estimation. By helping a project manager efficiently manage the software handoff, project management is improved.

The contributions also improve not only organizations, but also the knowledge base of software development. By modeling the software handoff, the impact of inter-group coordination on software development can be described and studied in greater detail than previously possible.

1.13 Dissertation Format

The format of the remainder of the dissertation is as follows. Chapter 2 provides a detailed literature review on the field and progress of software cost estimation. The goal of this chapter is to show the progress and problems encountered in software cost estimation. Chapter 3 details the COCOMO II cost estimation model. COCOMO II represents the state-of-the-practice in software cost estimation. From this chapter, an
understanding of how to estimate software projects is presented. Chapter 4 addresses the conceptual development of communication overhead. Communication is an important aspect in software cost estimation that is missing from current software estimation models. This chapter will present the theoretical and mathematical development of inter-group coordination and intra-group communication. Chapter 5 details an extended software cost estimation model. This new extended model builds on COCOMO II as presented in Chapter 3 and the communication overhead discussion presented in Chapter 4. At the end of this chapter, the new extended model is developed and introduced in a tool for project managers called PSEstimate. Chapter 6 shows the tool PSEstimate in use and the type of problems it can solve. Chapter 7 presents the experimental validation of the new extended software cost estimation model, and in this chapter the experimental design is outlined. Hypotheses are presented based the research questions introduced in Chapter 7. Chapter 8 presents the empirical results from the experimental validation and a discussion of these results. Chapter 9 concludes the dissertation and presents the contributions of this work.
CHAPTER 2 LITERATURE REVIEW

Only in software do people cling to the illusion that it’s OK to come up with estimates of the future, even though you’ve never measured anything in the past.

- Tom DeMarco (Brady and DeMarco 1994)

2.1 Introduction

With the invention of the electronic computer circa 1945 and the first high-level programming language, FORTRAN, circa 1955, people wanted to know the cost of developing a software project. The problem of software cost estimation became relevant around 1975 when software development methodologies emerged (Nemecek 2001). This chapter reviews the relevant literature on software cost estimation. Based on the literature, three ideas concerning the state of the art of software cost estimation will be expressed; there is a lack of a theoretical framework for estimation, very limited progress has been made in estimation, finally, drastic changes in modeling are needed to improve estimation.

The first point is that software cost estimation is plagued by a lack of a theoretical framework. Without a theoretical framework, the causes of cost in a software project are difficult to verify. A theoretical framework enumerates important metrics that need to be collected for software cost estimation models. “Even today, industry surveys indicate only about 25 percent of application development organizations have a formal metrics program” (Yourdon 1994). Because a theoretical framework is lacking, construct
development is not conducted with software cost estimation measures. Instead of properly developing constructs from a theoretical framework, significant correlations from statistical methods are used in software cost estimation models. By measuring many variables and using a “shotgun” approach, where correlations are run between all variables to see if any correlations are significant, eventually some variables will be found to have significant correlations even though the relationship might only be a spurious correlation. In addition, it is unclear if the significant correlations found are the artifact of violating the assumptions of a particular statistical method.

The second point is very little progress has been made on the problem of trying to devise high-quality software metrics that model cost. Software development is a very difficult task to understand; estimating software costs is even more difficult. Software cost estimation models are not much better today than they were over 20 years ago. We are rarely able to predict accurately the cost of any software development project (Nemecek 2001). Some researchers claim that “no prediction technique has proved consistently accurate, even when we relax the accuracy criterion to merely require that a technique generates useful predictions” (Kadoda, Cartwright et al. 2000). Software engineering has seen a shortage of competent software developers with an increasing amount of work to be done; this phenomenon is commonly called the “software crisis” (Amoroso and Zawacki 1992). Much of the work on software cost estimation follows the work done to solve the “software crisis” problem. Many attempts have been made to increase software developers’ productivity, but theoretical frameworks to explain productivity are rare. With a lack of a theoretical framework, empirical evidence is
sometimes ignored. Solid empirical findings, such as an increase in productivity can be realized by giving software developers an office with at least 90 square feet (DeMarco and Lister 1999) are rarely used in practice (Jones 1988). Software cost estimation is dependent on the subfield of software engineering, software measurement and the metrics developed. Unfortunately, software measurement has a very poor empirical knowledge base because of inappropriate or inadequate use of measurement. Many empirical findings are suspect mainly “because of their poor experimental design and lack of adherence to proper measurement principles” (Fenton 1993 p.141). Even though much work has been done to improve software development, very little progress in the way of practical or theoretical contributions actually enhances software cost estimates.

The third point is that the field of software cost estimation will never mature unless drastic changes are applied. The field of software engineering or software development is different from all other fields. With over 25 years of work on software cost estimation, most estimates are at best, guesses. The popular advice of taking the best software cost estimate and double it shows the difficulty in using an atheoretical approach to software cost estimation. Nemecek (2001) tells the story of a project manager, who after winning a large software development contract was asked how the estimate for effort was derived. The project manager summed the worst-case estimates for the project and then multiplied effort by 400%. When the project was completed, it still ran over budget (Nemecek 2001). A drastic change in software cost estimation will force a change to using a theoretical approach to software cost estimation. Researchers
claim that a simple theoretical framework was shown to be better than the most popular software cost estimation model (Smith, Hale et al. 2001).

It takes a diverse skill set to provide solutions to the software cost estimation problem. Competency in three main fields, Software Engineering, Management, and Statistics are needed. According to Jones, universities do not properly prepare their graduates for immediate assimilation into commercial software development. About 1 year of remedial training and $15,000 to $25,000 in training must be spent before an entry-level graduate software engineer can be entrusted with commercial-grade software projects in a major company. At the same time, the curriculum of software managers is lagged by 5 year behind the state of the art (Jones 1998).

With neither new graduates nor software managers having up-to-date knowledge on software cost estimation, champions’ support for a strong estimation program is difficult to achieve. Since there is such as long learning curve for both entry-level graduates and software managers, tools that provide a decision support system are needed. Managers will need state-of-the-art tools to help them manage their jobs. Furthermore, research has shown that tools that explain “why” or provide cognitive support to an answer are more preferable than the tools that just provide the outcome solution (Sengupta and Abdel-Hamid 1993). Software managers prefer the cognitive support that theoretical models can provide.

2.2 Cost Estimation Needs

Software cost estimation tools are needed to help manage all but trivial system development projects. An accurate estimate can be used by management to support estimating the cost of proposed new system, perform design-to-cost analysis, schedule the personnel and resources needed throughout the development, and monitor the
progress of the project (Adrangi 1987; Cover 1988). Since capital to invest in software
development projects is scarce, companies prioritize development projects on some sort
of cost/benefit analysis. A valid cost estimate will allow a company to develop the best
software development given a limited amount of capital. Scheduling personnel and
resources is an important activity for software cost estimation tools. Knowing how many
people will be needed and the amount of time required to develop the project will allow
management to provide the resources required to develop the software. Monitoring the
progress of the project is important to know if the project is on track or if it is falling
behind schedule.

Valid software cost estimations also allow other parts of a business to be more
productive. Sales and marketing need estimates when the project will be completed in
order to be effective. Many times software is marketed but no product ever ships.
Manufacturing delays causes an inefficient use of time for many people in an
organization.

Having valid software cost estimations is important to an organization. Software
that is late, over budget, or is of poor quality creates a major distraction to an
organization that develops software. Even very successful companies have problems
delivering large software projects on time and on budget. The only real solution is to have
organization use valid software cost estimations.

Even though cost estimation tools are needed, the solution is not so simple. It is
difficult to extract the variables that influence effort. While it may seem simple to
measure the productivity of the team and assume the team will have that same
productivity on other projects, it was shown that factors beyond the control of a software
development team have a significant impact on the productivity of a software
development team (Leavitt 1977). As the same team moves from project to project, two
different productivities will be seen.

In 1979, Larry Putnam considered software cost estimation an “intelligent
guessing game” and warned against software pitfalls such as cost overrun, schedule
slippages, and interdepartmental communication breakdowns. He said the poor project
estimation is one of the major problems in software development and attributed these
failures to the fact that software management and development is a science still in its
infancy (Scannell 1979). In 1988 and 1989, software was still being delivered late, over
budget, with poor quality and missing features, therefore an empirical study was
conducted to see why, and the major problem was underestimation of effort (van
Gunuchten 1991). Tasks were found to be more complex than initially estimated;
therefore, frequent budget and schedule overruns were common. Another study found
that software managers fail to learn from their mistakes by continuing to undersize
software size (Abdel-Hamid and Madnick 1989). Today, cost overruns and late software
are still common. To a point, the “intelligent guessing game” continues.

2.3 Cost Estimation Solutions

Software cost estimation lacks a strong theoretical foundation. Practioners rather
than researchers are leading the work conducted in software cost estimation. When the
task is to create an estimate for a particular company, theory is not considered. By
reviewing the history of software cost estimation, many potential problems that are
difficult to be solved by people in the field can be addressed using a theoretical foundation.

Software project cost estimation started by understanding that the bigger a project was to be developed, the more effort and the longer it would take to develop. Managers assumed that productivity rates of programmers were constant. Software development was thought to be linear, to do twice the work, you need twice the time. Therefore, the size of the system needed to be estimated, and using the productivity rates of the programmers, the schedule, and the number of people to develop the system could be calculated. If the schedule needed to be shortened, more programmers were added to the system. Brooks showed though that effort and schedule could not be directly interchanged (Brooks 1975).

Putnam wrote that the phenomenology of the software development process is not known, but data suggests a clear time-varying pattern (1978). Norden (1970) applied Lord Rayleigh’s distribution, to describe the projected labor needed during the stages of hardware development. Putnam applied Norden’s concepts to software development (1978). Putnam’s Software Equation was the result of the Rayleigh curve applied to software development and is summarized as follows “It has been discovered that there is a fundamental relationship in software development between the number of source statements in the system and the effort, development time, and the state of technology being applied to the project” (Putnam and Fitzsimmons 1979).

The Programmed Review of Information for Costing and Evaluation – Software (PRICE S), developed in 1977 by Martin Marietta Price Systems, was the first complex
commercially available software cost estimation tool. PRICE S is a proprietary cost estimation model developed by Lockheed Martin. To use this cost estimation model, a company would have to hire a Lockheed Martin consultant to conduct the cost estimation. Government agencies such as NASA, IRS, U.S. Air Force, U.S. Army, U.S. Navy, etc, as well as private companies have used PRICE S (NASA 2002 p. 35).

TRW Defense and Space Systems Group wanted a software estimation model that was developed with a well-defined set of criteria. In addition, the cost estimation model was required to be related to actual software project dynamics and the majority of the cost model, not based on poorly calibrated subjective factors (Boehm and Wolverton 1980). From this development work, COCOMO (Boehm 1981) was designed. By using a database of metrics built from software development projects, a regression was conducted to relate project size with project effort. Cost estimation of software development and control of cost during development is cited as being difficult because of a lack of useful cost history figures, therefore a software-cost database was developed to support cost estimation (Dekker and van den Bosch 1983). A software metric based on the sum of the number of files, flows, and processes in the system was found to be valid and reliable for a database of 20 different systems (van der Poel and Schach 1983). In this study, the researchers attempted to show that the cost of developing a system is directly proportional to its size, the size and cost of software can be accurately estimated early in the software development process, and the size of the software and the cost can be used to determine the efficiency of the development process.
Boehm’s book on Software Engineering Economics detailed five different software cost estimation techniques including algorithmic cost estimation, expert judgment, cost estimation based on previous experience, price-to-win cost estimation, and top-down/bottom-up costing (Boehm 1984). The argument made was that it is important to use an economic-based perspective to software engineering. By applying an economics-based perspective to cost estimation, the technical aspects of a software project can be analyzed in relation to the resource constraints that characterize the software engineering environment. Therefore, by way of the duality principle (Musgrave and Rasche 1977), a better estimate will be found than by just looking at technical aspects or resource constraints alone. Cost estimation models not built with the duality principle in mind, have a weakness in having spurious correlations if using regression. Research such as (van der Poel and Schach 1983) and (Dekker and van den Bosch 1983) that are not based on the duality principle provide little empirical evidence because their promising results are probably based on spurious correlations. Very shortly after Boehm argued for an economics-based perspective to software engineering, a study that looked at both the resources and the workload of a system was published (Italiani 1984). Italiani analyzed the performance of a software staff based “conventional experience,” “relative capacity,” and a new construct, “working environment quality coefficient.” By creating a workload matrix involving development activities, Italiani created a theory, productive capacity of a software development system, to support software cost estimation. Unfortunately the impact of this work is limited.
Even with the new economics-based perspective to cost estimation, the backlog of software development projects was steadily increasing with cost overruns and schedule slippages costing companies real money. There were no standardized or reliable methods for cost estimation and project control therefore a better understanding of process was thought to be the answer (Raja 1985). Raja explains how the Rayleigh model for software development can be effectively used for cost control and project management. By combining concepts from statistics, performance evaluation, critical path method, and software engineering, a project size can be estimated as a function of total project effort and development time. Before this effort was always the dependent variable with size being the independent variable as shown in Equation 2.1. Raja made software size the dependent variable with effort and development time being independent variables as shown in Equation 2.2. Raja through his modeling asked the question, with a given amount of effort, what sized projects could be built.

$$\text{Effort} = \beta_1 \times \text{Size} + \beta_0$$

Equation 2.1 Effort Equation

$$\text{Size} = \beta_2 \times \text{Effort} + \beta_3$$

Equation 2.2 Size Equation

A study (Kitchenham and Taylor 1985) was done to determine the effectiveness of the Putnam’s Rayleigh curve model and COCOMO with 33 software development projects. Kitchenham found that neither the Putnam nor the COCOMO model adequately
fitted the data when looking at software size, effort expended, and the time required for development.

By 1985, several cost estimation models were proposed, but very little external empirical validation was successfully completed on any of the proposed models. Modern systems were becoming more software intensive with software development definitely being on the critical path for system delivery. Other areas of software development were becoming mature, but cost estimation made little progress. Software is not a repetitive task like creating an automobile; instead, software is developed rather than built, therefore traditional experimental methods, which are common with agriculture and assembly line production, are very difficult to conduct with software development. Without experimental methods, it is hard to verify cause and effect during software development. Instead of developing and testing theories, best practices were used instead. By sharing best practices in software cost estimation allowed the field to slowly progress.

Many of the failures of software cost estimation have been because of the difficulty in measuring a software development system (Verner and Tate 1987). With size being the major variable to describe a software development system and the difficulty to measure a system size accurately, failure in software cost estimation can be understood. The usual way of measuring a system size is using lines of code (LOC). A popular quip summarizes the inadequacy of using lines of code as a measure of software size. “To estimate software development costs on the basis of LOC is analogous to estimating home construction costs based on the number of nails or bricks to be used” (Callisen and Colborne 1984). However, using lines of code is a poor measure because programmers
can easily manipulate the metric. Function Point Analysis (FPA) is one attempt to solve the sizing problem in software development. Some cost estimation models use different methods of sizing to mitigate the weaknesses of using lines of code as a size metric. Function Points are an improvement over lines of code, but fundamental flaws in the construction of function points prevent them from being valid measures (Kitchenham, Pfleeger et al. 1995; Kitchenham 1997).

Another method to solve the software-sizing problem is to calibrate the software estimation model. By using historical data, which may or may not be reliable measures representing software size, effort, and development time, better estimations were thought to be possible. The PRICE S model had a formal established methodology of calibrating productivity indexes with historical data. With the methodology, the organization was calibrated rather than the model (Park 1988). The advantages of calibrating a software cost estimation model were shown (Cuelenaere, van Genuchten et al. 1987). Software cost estimation is important because software continues to be large part of the cost of modern systems, therefore based on the state of estimating, there was a request for more efficient software cost models (Ferens 1988). Human, technical, environmental, and political reasons all can affect the effort and time required to develop a system so there was a claim that software cost estimation will never be an exact science (Navlakha 1990). Through an experiment, Navlakha showed the importance of customizing a software cost estimation model to an organizational environment.

The software cost estimation field was revitalized with object-oriented development. Using object-oriented development, the method of software sizing became
more accurate because the strong link between specifications and implementation (Laranjeira 1990). The number of classes and methods in an object-oriented system provides more insight into the project size than just lines of code. With object-oriented development, software metrics became a popular avenue of research. There was interest to develop new metrics around the new paradigm in programming. Many of the metrics developed were highly correlated with software size, and this provided no support to the software-sizing problem.

A major work done by Abdel-Hamid (1991) provided many insights into software development by using a novel approach for researching software engineering. By using a dynamic simulation model, various inputs were allowed to change over time. The field of Calculus was now being applied to software development instead of multiple regressions from statistics. By building a model of the software development environment, variables that affected software development were very explicitly described. An integrated theoretical framework to software development was built. Many interesting finding came out of this research (Abdel-Hamid 1988; Abdel-Hamid 1988; Abdel-Hamid 1989; Abdel-Hamid 1992; Abdel-Hamid 1993; Abdel-Hamid, Sengupta et al. 1994; Abdel-Hamid, Sengupta et al. 1999), but results of the work have yet to be integrated into modern software cost estimation models. Relevant findings such as communication overhead in software development, schedule pressure, learning curves in software development, productivity lost to training new employees, task underestimation, and the effects of turnover in system development are not modeled in most software cost estimation models; even though they are shown in the simulation to drastically affect effort and
schedule. The knowledge created from software dynamics has not yet been used to
develop better software cost estimation models. Abdel-Hamid even describes how the
interdependency of projects results in a fundamental deficiency in the formulation of
current generation cost estimation tools (1993). Abdel-Hamid believes that the reason
software cost estimation model have low portability is because of the lacking of the
models to quantify the effect of managerial decisions on cost (1987). Two identical
projects can be conducted by two different organizations but most cost estimation models
will not provide different estimates to the effort and schedule, even though the first
project might have three times the amount of employees as the second project. Current
cost estimation models have poor linkages to the real world of software development.
The lack of cost estimation models built on theoretical frameworks is the reason.

The Minimum Software Cost Model (MSCM) (Hu, Plant et al. 1998) is software
cost estimation model built from economic production theory and systems optimization
theory. In particular, the MSCM was derived from the Cobb-Douglas production
function. Using Kemerer’s data set of 14 projects (1987), the MSCM was declared to be
superior to all other software cost estimation models. Unfortunately, all but two of the
projects in the database were COBOL systems so this does not help in estimating modern
object-oriented systems.

A study used four new constructs as inputs to software cost estimation. Team size,
concurrency, intensity, and fragmentation where shown to have goodness of fit and
quality of estimation superior to that of the COCOMO model, while being more
parsimonious (Smith, Hale et al. 2001).
Team Size

Team size is an important construct because Brooks (1975) showed that managers often employ additional people to late projects in order to rescue the project. However, the additional communication and training needed cause the project to become late. Brooks’ Law was later empirically validated (Sengupta, Abdel-Hamid et al. 1999). It was shown that big teams cause negative effects during development (Fried 1991), and Putnam (1985) stresses using a small team approach for production of reliable systems. According to Smith and all, “Although no prior research has been found that directly explores the relationship between team size and development effort, these related finding support an expected negative relationship between the two” (Smith, Hale et al. 2001). No software cost estimation model specifically model the size of a team into the calculation. Hence, public dataset do not provide the amount of people that worked in a team.

Intensity

Smith et al (2001) also devised a construct called intensity, which measures the degree of schedule compression. It is thought that high developer productivity requires single-minded work time, and for each interruption, immersion time is required to restore the high productivity. This is the main reason that a private office increases a developer’s productivity. Having a developer working on a single task should result in higher productivity. However, Putnam warns that schedule compression increases communication noises, which introduces ambiguities into the development process, and results in lower productivity as people interrupt each other to resolve the ambiguities (Putnam 1985). If given too much time to complete work, the work will scale to fill the
allotted time. Intensity is not included as a factor in software cost estimation models even though research has shown that it is an important driver of productivity, which affects costs.

Concurrency

Concurrency is the degree to which team members work together or independently on a portion of the software project. The degree to which people work together is shown to be critical to team performance (Guinan, Coprider et al. 1998). Yet software cost estimation models do not include a measure of concurrency in the model. COCOMO II does include a qualitative measure of team interactions. A software development team is rated on a scale from having very difficult interactions to seamless interactions. Higher the scale, the larger the effort is needed. Concurrency instead explains if people are working together or independently.

Fragmentation

The last construct advanced by Smith is fragmentation. Fragmentation examines the degree to which a team’s time is broken up over multiple modules. While it is understandable that fragmentation leads to decreased efficiency, managers argue that cross-pollination of ideas ensure consistent approaches on multiple modules (Reinertsen 2000). A person that works 80 hours per week on a module with no fragmentation cannot easily increase the amount of hours worked on that module whereas someone that only works 20 hours per week. Forcing developers to work to a rate of full-time utilization
only guarantees queues and delays. Nevertheless, software cost estimation models do not include a measure of fragmentation.

2.4 Empirical Model Building

The majority of software cost estimation models that have been developed are empirically based (Cover 1988). Most models are a variation of a basic effort equation as shown in Equation 2.3.

\[ E = c \times a^b \]

Equation 2.3 Basic Effort Equation

In the basic effort equation, “E” stands for effort, and “a” is normally the size of the project, usually in lines of code. Both “c” and “b” are constants established through an analytical technique, usually regression.

Historical data usually consisted of effort, project size, and project duration. From this historical data, software cost estimation models first tried to relate project size and effort. It was generally agreed that bigger projects should take more effort to develop. What was not known was whether a project twice as big as another would take twice the effort to develop. This economies and diseconomies of scale of software development was the first empirical task software cost estimation models tried to answer. It was important to explore whether the relationship between project size and effort was linear. By collecting project data that included how much effort was required and the total size of the software project, a multiple regression was conducted with the dependent variable being effort and the independent variable being software size.
Later it was found that software size and effort did not have a linear relationship, except for very small software development projects. Boehm first showed that there were diseconomies of scales in software development (Boehm 1981). Instead of using a linear relationship to model software size and effort, a nonlinear relationship was used to fit the data better. Using a nonlinear relationship to model size with effort resulted in the first modern cost estimation tool, Barry Boehm’s Constructive Cost Model (COCOMO) (Boehm 1981) was created.

2.5 COCOMO

COCOMO includes three different types of cost models; these types are basic, intermediate, and detailed. All three models used thousands of lines of delivered source code or KSLOC as a measure of software size. The differences between the three models were accuracy. To be more accurate, the model required more information.

The simplest cost model was Basic COCOMO, but it provided the most unreliable results of the three, but only simple information was needed as input. The model is shown in Equation 2.4.

\[ \text{Effort} = a \times (KLOC)^b \]

Equation 2.4 Basic COCOMO Equation

There were only three parameters and the software size as the input to the equation. The output, Effort, was given in man-months. One man-month equals one person working for a month.
The equation for schedule for Basic, Intermediate and Detailed are all the same. Schedule explains the number of calendar months it will take the software project to complete.

\[ Schedule = 2.5 \times Effort^{c} \]

Equation 2.5 COCOMO Effort Equation

Three constants are needed to come up with a numerical answer for effort and schedule. The project’s development type first has to be known, and then the constants can be found by referring to Table 2-1.

Three different project types are defined by COCOMO, Organic, semidetached and embedded. In organic mode, the software development team is small. Usually only small (less than 50 KSLOC) projects are developed by an organic team. Most people developing the software have experience and thorough understanding of the system will contribute to the organizations objectives.

Semidetached mode is a compromise between organic and embedded. Typically projects that are in the semidetached mode are no bigger than 300 KSLOC.

In embedded mode, the project needs to fit within tight constraints and these are the most difficult software development projects developed. A missile system would be a type of embedded software development project.

<table>
<thead>
<tr>
<th>Development Type</th>
<th>Constant a</th>
<th>Constant b</th>
<th>Constant c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organic</td>
<td>2.4</td>
<td>1.05</td>
<td>0.38</td>
</tr>
<tr>
<td>Semidetached</td>
<td>3.0</td>
<td>1.12</td>
<td>0.35</td>
</tr>
<tr>
<td>Embedded</td>
<td>3.6</td>
<td>1.20</td>
<td>0.32</td>
</tr>
</tbody>
</table>

Table 2-1 Basic COCOMO Constants
Intermediate COCOMO was more accurate by adding 15 more parameters. These 15 cost drivers characterize product attributes, computer attributes, personnel experience, and software tools and practices. A project that has a higher complexity will have a higher cost driver; therefore, the project takes more effort and time to complete.

Intermediate COCOMO has the following equation for effort:

\[
\text{Effort} = EAF \times a \times (KLOC)^b
\]

Equation 2.6 Intermediate COCOMO Effort Equation

A new variable, EAF, represents the product of the 15 cost drivers. In order not to overestimate effort when using Intermediate COCOMO with effort multipliers, the value for the “a” constant changes from the Basic COCOMO model. The constants “b” and “c” however are the same. For organic the new constant is a=3.2; semidetached a=3.0; and embedded a=2.8.

Detailed COCOMO is very similar to Intermediate COCOMO but instead uses the 15 different cost drivers through each phase of the software development lifecycle. This way a cost driver can focus on a specific phase rather than having to apply to the whole project. By individually estimating each phase, for a project with new programmers and very experienced designers, the effort for the implementation phase will increase whereas the effort for the design phase will be reduced.

2.6 COCOMO II

Over 15 years after releasing COCOMO, COCOMO II (Boehm 2000) was developed. This major work updated an outdated software cost estimation model that COCOMO had become. Both COCOMO and COCOMO II used a database of project in
which multiple regressions were run in order to create scale and effort multipliers. Just before COCOMO II came out, the projects in the COCOMO database were almost 20 years old. The software cost estimation model was not reflecting many improvements in productivity. Today, COCOMO II has over 100 commercial implementations, and is the most widely used software cost estimation tool. COCOMO II is discussed in more detail in chapter three.

2.7 Other Modern Software Cost Estimation Tools

While COCOMO II is the most used modern cost estimation tool, several tools also exist. Using the Basic COCOMO formula, but with different values for the constants, there are three different cost estimation models, Walston-Felix (1977), Bailey-Basili (1983), and Doty (Herd, Postak et al. 1977). Using a simple regression between function points and effort resulted in Albrecht-Gaffney (1983), Kemerer (1987), and Matson, Barret and Meltichamp (1994) cost estimation models. Putnam’s SLIM model (1978; Putnam and Myers 1992) is different from all other cost estimation models in terms of equation form, but outputs values are very close to COCOMO II.

2.8 New Findings Not Assimilated Into Software Cost Estimation Models

Angelis et al (2001) used recent data collected by the International Software Benchmarking Standards Group to create a software cost estimation model. This data set consisted of historical data from many different types of organizations. They conducted a regression with the basic effort equation as the model. The results showed that 44% of the variance of was explained when predicting effort with size. A categorical regression was
conducted with many variables, but the variable, maximum team size, was found to be significant. With the maximum team size placed into the model, the explained variance doubled to around 88%.

Using dimensional analysis is common in fields like Physics, Chemistry, or Math where units matter. “Dimensional analysis is a method of comparing the dimensions of the physical quantities occurring in a problem to find relationships between the quantities without having to solve the problem completely” (Random House 1998). Equation checking is part of dimensional analysis. In this step, the formula’s theoretical derivation is checked based on algebra. If the units on both sides of the equation are equal, the equation is said to be commensurable. If the units are not equal, for example, if apples are on one side of the equation, and oranges are on the other side, the equation is said to be incommensurable. After studying all the software cost estimation models, “Conventional software models can not be correct because each is incommensurate” (Nemecek 2001). Predicting effort with size using regression is not a valid theoretical derivation.

Another study was done to look at the sensitivity of COCOMO II (Musilek, Pedrycz et al. 2002). After conducting three types of sensitivity analysis including mathematical analysis of the effort equation, Monte Carlo simulation, and error propagation, the size variable in COCOMO II was found to be very sensitive followed by the effort multipliers. The exponential factor has little impact of error. The authors suggest using fuzzy set of inputs to software size whereby giving the project manager a spectrum of effort estimations rather than a single point estimate.
Neural networks also have been used to predict effort. In this particular study (Idri, Khoshgoftaar et al. 2002), size plus four effort multipliers were placed in the neural network. This study used the COCOMO dataset and the researchers claimed that the “results are acceptable”. Although, understanding and interpreting the resulting neural network was found to be very difficult.

Estimating by analogies or case-based-reasoning is another technique used to predict effort. The use of analogies as a technique was suggested over 20 years ago (Boehm 1981). The effectiveness of case-based-reasoning greatly relies on the underlying dataset used for analogies. Case-based-reasoning is a type of cluster analysis and inherits the weakness of any cluster analysis methodology.

“Cluster analysis is the name for a group of multivariate techniques whose primary purpose is to group objects based on the characteristics they possess. Cluster analysis classifies objects (e.g., respondents, products, or other entities) so that each object is very similar to others in the cluster with respect to some predetermined selection criterion. The resulting clusters of objects should then exhibit high internal (within-cluster) homogeneity and high external (between-cluster) heterogeneity. Thus, if the classification is successful, the objects within clusters will be close together when plotted geometrically, and different clusters will be far apart” (Hair, Anderson et al. 1998 p.473).

Case-based-reasoning is often used in task domains that have no strong theoretical models and where the domain rules are incomplete, ill-defined, and inconsistent (Mukhopadhyay, Vicinanza et al. 1992). The number of possible project factors is a problem for many software cost estimation models. Over 74 different project factors have been identified (Wrigley and Dexter 1987). Predetermining some set of project factors then running a cluster-type analysis on a published data set usually yields favorable
results. Consider the case-based-reasoning model called Estor. “Estor did not perform quite as well as the human expert, but it did outperform existing algorithmic model on the data set” (Mukhopadhyay, Vicinanza et al. 1992 p.167). Estor estimates averaged 52% within actual estimates when COCOMO averaged 618% within actual estimates. The goal of software cost estimation is not to predict the cost of historical data, but rather to predict the cost of new projects. The authors write, “To be fair, Estor would almost certainly fail to accurately estimate project from different environment (e.g. embedded military systems) with additional domain knowledge” (Mukhopadhyay, Vicinanza et al. 1992 p.167). “Estimates of the accuracy of prediction obtained from a training set are always optimistic. To get a more realistic estimate of the accuracy of prediction you either have to use a new, independent data set or adopt a jack-knife approach” (Samson, Ellison et al. 1997 p.59).

An important study was conducted to show the causes of estimating error. Only one managerial practice, which was the use of the estimate in performance evaluation of software managers and professionals, was shown to increase accuracy of estimates. Software cost estimation models were shown to be no help. The authors write

“… It is unexpected that the application of the algorithmic basis failed to predict estimating accuracy. Apparently, the use of complex statistics, software, and standards do not facilitate more accurate estimates. Such a finding does not imply that software managers and professionals should shun algorithm-based estimating techniques. However it intimates instead that they recognize their shortcomings: Specifically, the employment of algorithm-based estimating methods did not improve the accuracy of cost estimates for subjects in this research. When using such methods, software managers and professionals probably need to be very careful to avoid the impression in other managers and users that they can guarantee meeting algorithm-based estimated” (Lederer and Prasad 1998).
By holding estimators responsible for their estimates is probably the only way software cost estimation is going to improve. Once people are responsible for their estimates, substandard models will not be tolerated.

2.9 Empirical Datasets

Empirical validation of software cost estimation models using regression depend on the quality of the datasets available. COCOMO II has the best dataset of projects with “161 carefully-collected” projects (Boehm and Sullivan 2002). However, the dataset is proprietary and not published. COCOMO only needed 63 projects to have the same predictive accuracy as COCOMO II, which is being within 30% of the actual metric, 75% of the time (Boehm 1981). The larger required dataset need by COCOMO II shows the difficulty of using regression to develop cost estimation models. Empirically validating a cost estimation model using a regression approach with the following datasets is not very convincing. Two example empirical dataset are presented in Appendix A and Appendix B.

2.10 Other Validation Approaches

When experts estimate software costs without any formal algorithmic technique, they outperform software cost estimation models. The mean error rates of the experts’ predictions still ranged from 32 to 1107 percent (Lederer and Prasad 1998). Experts have a better idea at estimating software parameters than software cost estimation models, so the knowledge experts have has yet to be transferred into a software cost estimation tool. In the absence of empirical data, professional judgment should be used. The Delphi
method is a method to capture and properly document the knowledge being shared from an engineer’s expert opinion (NASA 2002 p.39). Using experts to validate a software cost estimation tool with a technique such as the Delphi method solves the problem of having large empirical datasets, but finding capable experts a problem. Unfortunately, according to Andy Prince, “Everyone is an expert on cost. Get used to it” (NASA 2002 p. 170).

2.11 Conclusions

Software cost estimation remains a difficult problem. With current estimates that still result in millions of dollars being spent in projects running over budget, the need to have better estimates will continue. There are some new ideas that can be used to make a better software cost estimation model, mainly the work on team size and task assignment. Even though many hundreds of variables have been proposed as inputs into software cost estimation, none of the variables has shown external empirical validity. Yet there is a need to build better models, and future software cost estimation models are going to have to be manager oriented. Since “software cost estimation is the process of predicting the amount of effort required to build a software system and is a fundamental managerial planning activity” (Nemecek 2001). Software cost estimation is more than just about the size of the project. Having 10 people or 100 people working on a project makes a big difference because of the mythical man-month (Brooks 1975).

This dissertation will provide a drastic change to the field of software cost estimation by placing the most logical driver of cost missing from current generation models, configuration of workforce and team size, into the formula.
CHAPTER 3 COST ESTIMATION IN COCOMO II

“We shall not fail or falter; we shall not weaken or tire...Give us the tools and we will finish the job.”
- Sir Winston Churchill, BBC radio broadcast, Feb 9, 1941

3.1 Introduction

This chapter describes the first two constructs used in this dissertation, project characteristics and COCOMO II outputs. This chapter will review how COCOMO II models differences in project characteristics to estimate effort, schedule, and staffing needed to conduct the particular software development project. The equations used by COCOMO II to produce the outputs are described.

3.2 Project Characteristics

It is safe to assume that no two software projects are alike. Given any two software development projects, differences can be found between the projects. Therefore, it is important to identify and quantify the significant differences among software development projects. Project characteristics are the independent variable in software cost estimation models, meaning differences in project characteristics create changes in the dependent variables, effort, and schedule.
Software Size

The first project characteristic that was modeled was software size. Common sense leads researchers to theorize that larger software development projects will take more effort and more time to complete than small projects.

COCOMO II uses a measure of software size in the algorithm to calculate effort and schedule. COCOMO II uses thousands of delivered source lines of code (KSLOC) as a measure of software size. Measuring KSLOC is not universal. With the same source code there are different methods for counting KSLOC. For example, the following simple code can be counted in many ways:

```c
int x, y, z; x=3; y=4; z=2; int xyz = x+y+z;
```

The line above code can be considered one line of code, or five, depending on the counter. The same functionality can be rewritten to be seven lines of code as shown below:

```c
int x; (1)
int y; (2)
int z; (3)
int x=3; (4)
int y=4; (5)
int z=2; (6)
int xyz = x+y+z; (7)
```

An alternative is to code as follows with one line of code:

```c
int xyz=3+4+2; (1)
```
Since programmers have control over how they implement the code, large variations are susceptible to the KSLOC measurement. Another solution was devised to get around the problem that exists using KSLOC. By measuring functionality rather than lines of code, the same logic can be used to argue that bigger programs require more effort still applies. A project with more functionality will require more effort and schedule time to complete versus a project with less functionality.

Information systems are commonly sized by functionality, like number of graphical user interface screen or reports. Function points are used instead of lines of code to measure software size.

COCOMO II’s internal algorithms only use KSLOC in estimating effort and schedule. A process called backfiring is used to convert function points into SLOC. COCOMO II can convert from unadjusted function points to lines of code based on programming language used to implement the function points. Table 3-1 shows the conversion factors for different programming languages.
<table>
<thead>
<tr>
<th>Programming Language</th>
<th>SLOC per Unadjusted Function Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>38</td>
</tr>
<tr>
<td>Ada 83</td>
<td>71</td>
</tr>
<tr>
<td>Ada 95</td>
<td>49</td>
</tr>
<tr>
<td>AI Shell</td>
<td>49</td>
</tr>
<tr>
<td>APL</td>
<td>32</td>
</tr>
<tr>
<td>Assembly – Basic</td>
<td>320</td>
</tr>
<tr>
<td>Assembly – Macro</td>
<td>312</td>
</tr>
<tr>
<td>Basic – ANSI</td>
<td>64</td>
</tr>
<tr>
<td>Basic – Complied</td>
<td>91</td>
</tr>
<tr>
<td>Basic – Visual</td>
<td>32</td>
</tr>
<tr>
<td>C</td>
<td>128</td>
</tr>
<tr>
<td>C++</td>
<td>53</td>
</tr>
<tr>
<td>Cobol (ANSI 85)</td>
<td>91</td>
</tr>
<tr>
<td>Database – Default</td>
<td>40</td>
</tr>
<tr>
<td>Fifth Generation Language</td>
<td>4</td>
</tr>
<tr>
<td>First Generation Language</td>
<td>320</td>
</tr>
<tr>
<td>Forth</td>
<td>64</td>
</tr>
<tr>
<td>Fortran 77</td>
<td>107</td>
</tr>
<tr>
<td>Fortran 95</td>
<td>71</td>
</tr>
<tr>
<td>Fourth Generation Language</td>
<td>20</td>
</tr>
<tr>
<td>High Level Language</td>
<td>64</td>
</tr>
<tr>
<td>HTML 3.0</td>
<td>15</td>
</tr>
<tr>
<td>Java</td>
<td>53</td>
</tr>
<tr>
<td>Jovial</td>
<td>107</td>
</tr>
<tr>
<td>Lisp</td>
<td>64</td>
</tr>
<tr>
<td>Machine Code</td>
<td>640</td>
</tr>
<tr>
<td>Modula 2</td>
<td>80</td>
</tr>
<tr>
<td>Pascal</td>
<td>91</td>
</tr>
<tr>
<td>PERL</td>
<td>21</td>
</tr>
<tr>
<td>PowerBuilder</td>
<td>16</td>
</tr>
<tr>
<td>Prolog</td>
<td>64</td>
</tr>
<tr>
<td>Query – Default</td>
<td>13</td>
</tr>
<tr>
<td>Report Generator</td>
<td>80</td>
</tr>
<tr>
<td>Second Generation Language</td>
<td>107</td>
</tr>
<tr>
<td>Simulation – Default</td>
<td>46</td>
</tr>
<tr>
<td>Spreadsheet</td>
<td>6</td>
</tr>
<tr>
<td>Third Generation Language</td>
<td>80</td>
</tr>
<tr>
<td>Unix Shell Scripts</td>
<td>107</td>
</tr>
<tr>
<td>Visual Basic 5.0</td>
<td>29</td>
</tr>
<tr>
<td>Visual C++</td>
<td>34</td>
</tr>
</tbody>
</table>

Table 3-1 Unadjusted FP to SLOC Conversion Ratios  
(Boehm 2000, p 20)
To convert from unadjusted function points to SLOC simply multiply the unadjusted function point estimate by the appropriate conversion ratio for the programming language in which development will occur.

There is a lower bound on software size as to what COCOMO II can estimate. COCOMO II has been calibrated for projects bigger than two KSLOC; therefore, the model built in this dissertation will also not be able to calculate projects smaller than two KSLOC. Projects smaller than two KSLOC are typically completed by only one person, and the developer’s skill highly determines the effort and schedule required to develop the project.

Scale Factors

Researchers use more than just software size to quantify a software development project. Differences in projects with the same software size lead researchers to add another component to the description of a project. By using the concept of a scale factor, the software size can adjust to circumstances that cause more or less effort needed for the same software size. For example, this allows two projects, both 40 KSLOC, to have different effort estimates based on scale factors.

COCOMO II has five scale factors that account for the economies and diseconomies of scale in software development projects. When there are economies of scale, doubling the software size will result in effort being less than double the original. Whereas when diseconomies of scale are present for a software project, doubling the project size will results in more than double of the original project effort being needed to complete the project.
COCOMO II uses Equation 3.1 to calculate if a project has economies or diseconomies of scale.

\[ E = B + 0.01 \times \sum_{j=1}^{s} SF_j \]

Equation 3.1 Economy of Scale Equation

In Equation 3.1, B is a constant and for COCOMO II.2000 the value is 0.91. If the value of E is equal 1.0 then the economies of scale and diseconomies of scale are in balance. If the value of E is less than 1.0 then the project has economies of scale. If the value of E is greater than 1.0 then the project has diseconomies of scale.

If the highest and lowest scale factors are applied to Equation 3.1, the result is that the economy of scale equation ranges from 0.91 to 1.2262. COCOMO II’s accuracy depends on correctly identifying the proper scale factors for a project.

Effort Multipliers

In addition to scale factors, there are other set of variables that are thought to help increase the quantifying of project characteristics. Effort multipliers are used as the third type of project characteristics along with software size and scale factors. COCOMO II
has two different sets of effort multipliers that should be used at different times. The first set is the Post-Architecture effort multipliers. The seventeen effort multipliers are to be used after the software architecture has been designed. The Early Design effort multipliers are an alternative to the Post-Architecture effort multipliers. The Early Design effort multipliers are best used when a high-level model is needed to explore architectural alternatives or incremental development strategies, whereas the Post-Architecture effort multipliers are best used when more detailed information about the architecture is available and a more accurate estimation is needed (Boehm 2000 p. 12). COCOMO II provides for either type of multiplier to be used.

The following table lists the quantitative values for each effort driver. The scale is divided by very low, low, nominal, high, very high, and extra high.
<table>
<thead>
<tr>
<th>Drivers</th>
<th>Description</th>
<th>VL</th>
<th>L</th>
<th>N</th>
<th>H</th>
<th>VH</th>
<th>XH</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELY</td>
<td>Required Software Reliability</td>
<td>0.82</td>
<td>0.92</td>
<td>1.00</td>
<td>1.10</td>
<td>1.26</td>
<td>n/a</td>
</tr>
<tr>
<td>DATA</td>
<td>Database Size</td>
<td>n/a</td>
<td>0.90</td>
<td>1.00</td>
<td>1.14</td>
<td>1.28</td>
<td>n/a</td>
</tr>
<tr>
<td>CPLX</td>
<td>Product Complexity</td>
<td>0.73</td>
<td>0.87</td>
<td>1.00</td>
<td>1.17</td>
<td>1.34</td>
<td>1.74</td>
</tr>
<tr>
<td>RUSE</td>
<td>Developed for Reusability</td>
<td>n/a</td>
<td>0.95</td>
<td>1.00</td>
<td>1.07</td>
<td>1.15</td>
<td>1.24</td>
</tr>
<tr>
<td>DOCU</td>
<td>Documentatio n Match to Life-Cycle Needs</td>
<td>0.81</td>
<td>0.91</td>
<td>1.00</td>
<td>1.11</td>
<td>1.23</td>
<td>n/a</td>
</tr>
<tr>
<td>TIME</td>
<td>Execution Time Constraint</td>
<td>n/a</td>
<td>n/a</td>
<td>1.00</td>
<td>1.11</td>
<td>1.29</td>
<td>1.63</td>
</tr>
<tr>
<td>STOR</td>
<td>Main Storage Constraint</td>
<td>n/a</td>
<td>n/a</td>
<td>1.00</td>
<td>1.05</td>
<td>1.17</td>
<td>1.46</td>
</tr>
<tr>
<td>PVOL</td>
<td>Platform Volatility</td>
<td>n/a</td>
<td>0.87</td>
<td>1.00</td>
<td>1.15</td>
<td>1.30</td>
<td>n/a</td>
</tr>
<tr>
<td>ACAP</td>
<td>Analyst Capability</td>
<td>1.42</td>
<td>1.19</td>
<td>1.00</td>
<td>0.85</td>
<td>0.71</td>
<td>n/a</td>
</tr>
<tr>
<td>PCAP</td>
<td>Programmer Capability</td>
<td>1.34</td>
<td>1.15</td>
<td>1.00</td>
<td>0.88</td>
<td>0.76</td>
<td>n/a</td>
</tr>
<tr>
<td>PCON</td>
<td>Personnel Continuity</td>
<td>1.29</td>
<td>1.12</td>
<td>1.00</td>
<td>0.90</td>
<td>0.81</td>
<td>n/a</td>
</tr>
<tr>
<td>APEX</td>
<td>Applications Experience</td>
<td>1.22</td>
<td>1.10</td>
<td>1.00</td>
<td>0.88</td>
<td>0.81</td>
<td>n/a</td>
</tr>
<tr>
<td>PLEX</td>
<td>Platform Experience</td>
<td>1.19</td>
<td>1.09</td>
<td>1.00</td>
<td>0.91</td>
<td>0.85</td>
<td>n/a</td>
</tr>
<tr>
<td>LTEX</td>
<td>Language and Tool Experience</td>
<td>1.20</td>
<td>1.09</td>
<td>1.00</td>
<td>0.91</td>
<td>0.84</td>
<td>n/a</td>
</tr>
<tr>
<td>TOOL</td>
<td>Use of Software Tools</td>
<td>1.17</td>
<td>1.09</td>
<td>1.00</td>
<td>0.90</td>
<td>0.78</td>
<td>n/a</td>
</tr>
<tr>
<td>SITE</td>
<td>Multisite Development</td>
<td>1.22</td>
<td>1.09</td>
<td>1.00</td>
<td>0.93</td>
<td>0.86</td>
<td>0.80</td>
</tr>
<tr>
<td>SCED</td>
<td>Required Development Schedule</td>
<td>1.43</td>
<td>1.14</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Table 3-3 Post-Architecture Effort Multipliers
<table>
<thead>
<tr>
<th>Drivers</th>
<th>Description</th>
<th>XL</th>
<th>VL</th>
<th>L</th>
<th>N</th>
<th>H</th>
<th>VH</th>
<th>XH</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCPX</td>
<td>Product Reliability and Complexity</td>
<td>0.49</td>
<td>0.60</td>
<td>0.83</td>
<td>1.00</td>
<td>1.33</td>
<td>1.91</td>
<td>2.72</td>
</tr>
<tr>
<td>RUSE</td>
<td>Developed for Reusability</td>
<td>n/a</td>
<td>n/a</td>
<td>0.95</td>
<td>1.00</td>
<td>1.07</td>
<td>1.15</td>
<td>1.24</td>
</tr>
<tr>
<td>PDIF</td>
<td>Platform Difficulty</td>
<td>n/a</td>
<td>n/a</td>
<td>0.87</td>
<td>1.00</td>
<td>1.29</td>
<td>1.81</td>
<td>2.61</td>
</tr>
<tr>
<td>PERS</td>
<td>Personnel Capability</td>
<td>2.12</td>
<td>1.62</td>
<td>1.26</td>
<td>1.00</td>
<td>0.83</td>
<td>0.63</td>
<td>0.50</td>
</tr>
<tr>
<td>PREX</td>
<td>Personnel Experience</td>
<td>1.59</td>
<td>1.33</td>
<td>1.22</td>
<td>1.00</td>
<td>0.87</td>
<td>0.74</td>
<td>0.62</td>
</tr>
<tr>
<td>FCIL</td>
<td>Facilities</td>
<td>1.43</td>
<td>1.30</td>
<td>1.10</td>
<td>1.00</td>
<td>0.87</td>
<td>0.73</td>
<td>0.62</td>
</tr>
<tr>
<td>SCED</td>
<td>Required Development Schedule</td>
<td>n/a</td>
<td>1.43</td>
<td>1.14</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Table 3-4 Early Design Effort Multipliers

The effort multipliers were designed to be independent factors, but the literature has shown the factors are often interrelated (Briand, El Emam et al. 1998). Briand also shows that even though some COCOMO factors appear to be useful and significant, they only play a minor role in explaining project effort because the impact on different models goodness of fit is weak. The conclusion of Briand’s research is that the effort multipliers described in this section might not be the correct variables. Nevertheless, from all the possible set of variables to use, COCOMO II uses the variables described in the section.

3.3 COCOMO II Outputs

This section describes the outputs from COCOMO II. The outputs are the dependent variables. Effort and schedule are the most common dependent variables. However, a lesser-known variable, the work breakdown structure, plays an important role too.

Development Effort

Estimating development effort is the main goal of software cost estimation. The common unit of measure of effort is man-months or the politically-correct person-
months. One person-month represents one person working for a month. The more person-months required the more effort is required to complete the project. An estimate in person-months can be easily converted into person-years, person-days, or person-hours by the appropriate multiplication factor. COCOMO II will provide all estimates of effort in man-months.

Project Duration

Project duration is a very important dependent variable in software cost estimation. Along with knowing the cost, knowing how long a project will take to conduct is a practical concern of project managers. COCOMO II will provide an estimate of the project duration in months. This estimate can be converted into different time units by the appropriate multiplication factor.

Work Breakdown Structure

COCOMO II provides a unique work breakdown structure based on the project size, effort estimate, and schedule estimate. By breaking down the whole project into three main activities, which are product design, programming, and integration and test, the amount of time needed to conduct requirements and analysis, product design, programming, test planning, verification and validation, project office, quality assurance, and manuals for each phase can be estimated.

As the project size and scale factors change, the work breakdown structure will also change. A sample work break down structure for a medium project (32K SLOC)
with a size exponent (diseconomy of scale) of 1.12 is shown in Table 3-9. COCOMO II derives the work breakdown structure from a table based on the relevant factors.

<table>
<thead>
<tr>
<th>Size Exponent</th>
<th>Size</th>
<th>Overall Phase Percentage</th>
<th>Requirements Analysis</th>
<th>Product Design</th>
<th>Programming</th>
<th>Test Planning</th>
<th>V &amp; V</th>
<th>Project Office</th>
<th>CM / QA</th>
<th>Manuals</th>
</tr>
</thead>
<tbody>
<tr>
<td>E = 1.05</td>
<td>S, I, M, L</td>
<td>6</td>
<td>46</td>
<td>20</td>
<td>3</td>
<td>6</td>
<td>15</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>E = 1.12</td>
<td>S</td>
<td>7</td>
<td>47</td>
<td>16</td>
<td>2.5</td>
<td>6.5</td>
<td>15.5</td>
<td>3.5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>E = 1.20</td>
<td>S</td>
<td>7</td>
<td>45</td>
<td>17</td>
<td>2.5</td>
<td>7</td>
<td>17.5</td>
<td>3</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

S: 2 KSLOC; I: 8 KSLOC; M: 32 KSLOC; L: 128 KSLOC; VL: 512 KSLOC

Table 3-5 Plans and Requirements Activity Distribution

<table>
<thead>
<tr>
<th>Size Exponent</th>
<th>Size</th>
<th>Overall Phase Percentage</th>
<th>Requirements Analysis</th>
<th>Product Design</th>
<th>Programming</th>
<th>Test Planning</th>
<th>V &amp; V</th>
<th>Project Office</th>
<th>CM / QA</th>
<th>Manuals</th>
</tr>
</thead>
<tbody>
<tr>
<td>E = 1.05</td>
<td>S, I, M, L</td>
<td>16</td>
<td>12.5</td>
<td>40</td>
<td>14</td>
<td>5</td>
<td>6</td>
<td>11</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>E = 1.12</td>
<td>S</td>
<td>17</td>
<td>12.5</td>
<td>41</td>
<td>12.5</td>
<td>5</td>
<td>6</td>
<td>13</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>E = 1.20</td>
<td>S</td>
<td>17</td>
<td>12.5</td>
<td>41</td>
<td>12.5</td>
<td>6</td>
<td>7</td>
<td>13</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

S: 2 KSLOC; I: 8 KSLOC; M: 32 KSLOC; L: 128 KSLOC; VL: 512 KSLOC

Table 3-6 Product Design Activity Distribution
<table>
<thead>
<tr>
<th>Size Exponent</th>
<th>E = 1.05</th>
<th>E = 1.12</th>
<th>E = 1.20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>S</td>
<td>I</td>
<td>M</td>
</tr>
<tr>
<td>Overall Phase Percentage</td>
<td>68 65 62 59</td>
<td>64 61 58 55 52</td>
<td>60 57 54 51 48</td>
</tr>
<tr>
<td>Requirements Analysis</td>
<td>5 5 5 5</td>
<td>4 4 4 4 4</td>
<td>3 3 3 3 3</td>
</tr>
<tr>
<td>Product Design</td>
<td>10 10 10 10</td>
<td>8 8 8 8 8</td>
<td>6 6 6 6 6</td>
</tr>
<tr>
<td>Programming</td>
<td>58 58 58 58</td>
<td>56.5 56.5 56.5 56.5</td>
<td>55 55 55 55 55</td>
</tr>
<tr>
<td>Test Planning</td>
<td>4 4 4 4</td>
<td>4 4.5 5 5.5 6</td>
<td>4 5 6 7 8</td>
</tr>
<tr>
<td>V &amp; V</td>
<td>6 6 6 6</td>
<td>7 7.5 8 8.5 9</td>
<td>8 9 10 11 12</td>
</tr>
<tr>
<td>Project Office</td>
<td>6 6 6 6</td>
<td>7.5 7 6.5 6 5.5</td>
<td>9 8 7 6 5</td>
</tr>
<tr>
<td>CM / QA</td>
<td>6 6 6 6</td>
<td>7 6.5 6.5 6.5 6</td>
<td>8 7 7 7 6</td>
</tr>
<tr>
<td>Manuals</td>
<td>5 5 5 5</td>
<td>6 6 5.5 5 5</td>
<td>7 7 6 5 5</td>
</tr>
</tbody>
</table>

Table 3-7 Programming Activity Distribution

<table>
<thead>
<tr>
<th>Size Exponent</th>
<th>E = 1.05</th>
<th>E = 1.12</th>
<th>E = 1.20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>S</td>
<td>I</td>
<td>M</td>
</tr>
<tr>
<td>Overall Phase Percentage</td>
<td>16 19 22 25</td>
<td>19 22 25 28 31</td>
<td>22 25 28 31 34</td>
</tr>
<tr>
<td>Requirements Analysis</td>
<td>3 3 3 3</td>
<td>2.5 2.5 2.5 2.5 2.5</td>
<td>2 2 2 2 2</td>
</tr>
<tr>
<td>Product Design</td>
<td>6 6 6 6</td>
<td>5 5 5 5 5</td>
<td>4 4 4 4 4</td>
</tr>
<tr>
<td>Programming</td>
<td>34 34 34 34</td>
<td>33 35 37 39 41</td>
<td>32 36 40 44 48</td>
</tr>
<tr>
<td>Test Planning</td>
<td>2 2 2 2</td>
<td>2.5 2.5 3 3 3.5</td>
<td>3 3 4 4 5</td>
</tr>
<tr>
<td>V &amp; V</td>
<td>34 34 34 34</td>
<td>32 31 29.5 28.5 27</td>
<td>30 28 25 23 20</td>
</tr>
<tr>
<td>Project Office</td>
<td>7 7 7 7</td>
<td>8.5 8 7.5 7 6.5</td>
<td>10 9 8 7 6</td>
</tr>
<tr>
<td>CM / QA</td>
<td>7 7 7 7</td>
<td>8.5 8 8 8 7.5</td>
<td>10 9 9 9 8</td>
</tr>
<tr>
<td>Manuals</td>
<td>7 7 7 7</td>
<td>8 8 7.5 7 7</td>
<td>9 9 8 7 7</td>
</tr>
</tbody>
</table>

Table 3-8 Integration and Test Activity Distribution

| Requirements & Analysis | 12.50% | 4% | 2.5% |
| Product Design | 41% | 8% | 5% |
| Programming | 13% | 56.5% | 37% |
| Testing Planning | 5.5% | 5% | 3% |
| V & V | 7% | 8% | 29.5% |
| Project Office | 11% | 6.5% | 8% |
| QA | 2.5% | 6.5% | 8% |
| Manuals | 7.5% | 5.5% | 7.5% |
| Phase Percentage of Total Effort | 17% | 58% | 25% |

Table 3-9 Work Breakdown Structure for a Medium Size Project
3.4 Model Types

With the independent variables and dependent variables described, the next step is to describe the relationship among all the variables. A model is needed to describe how the independent variables affect the dependent variables. In the literature, there are four common models used to relate software size to effort. In addition, research is conducted to identify the causes of economies or diseconomies of scale. On one hand, fixed overhead costs such as project management may not directly increase with system size; therefore, larger projects can realize economies of scale. On the other hand, some overhead activities, such as documentation, increase in excessive proportion to project size. As projects increase, the amount of work required for documentation increases more rapidly leading to diseconomies of scale.

From the software cost estimation literature, it is unclear if economies or diseconomies of scale exist. Most likely, mixed economies of scale exist, but it is difficult to know at which project size economies of scale can no longer be realized.

Kitchenham found that the relationship between effort and size is rather linear since the tendency of the constant $b$ in the log-linear model is to be 1.0 (Kitchenham 1992). By ignoring economies of scale and diseconomies of scale, the linear model was argued as being the best method to describe size and effort. Further research has shown that economies and diseconomies of scale exist in software development (Banker, Chang et al. 1994). Banker concludes that the log-linear relationship is too limited to model size on effort. Hu tested the linear, quadratic, log-linear, and translog model and found that
the quadratic model provided the most plausible relationship between effort and size (Hu 1997).

<table>
<thead>
<tr>
<th>Model Specification</th>
<th>Model Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \text{Effort} = a + (b \times \text{Size}) ]</td>
<td>Linear Model</td>
</tr>
<tr>
<td>[ \text{Effort} = a + (b \times \text{Size}) + (c \times \text{Size}^2) ]</td>
<td>Quadratic Model</td>
</tr>
<tr>
<td>[ \text{Effort} = e^a \times \text{Size}^b ]</td>
<td>Log-linear Model</td>
</tr>
<tr>
<td>[ \text{Effort} = e^a \times \text{Size}^b \times \text{Size}^{c \ln \text{Size}} ]</td>
<td>Translog Model</td>
</tr>
</tbody>
</table>

Table 3-10 Software Cost Estimation Model Types

(Briand, El Emam et al. 1998)

Briand et al (1998) states that the most plausible model to explain the costs of space and military projects is to use log-linear model involving KLOC, team size, and three COCOMO factors: reliability requirements (RELY), storage constraints (STOR) and execution time constraints (TIME).

COCOMO II uses the log-linear model to relate software size to effort. The model allows projects to have economies and diseconomies of scale through scale multipliers. It is unclear though if this is the best plausible model to describe the size/effort relationship.

3.5 Effort Estimation

With the software cost estimation model type picked for COCOMO II along with the independent and dependent variables, the next step in estimating effort is to instantiate the model. The formula to estimate effort in person-month is given in Equation 3.2
In Equation 3.2, PM stands for the total effort in person-months. A is a constant, which for COCOMO II, the value is 2.94. Size represents the estimated project size in thousands of source lines of code (KLOC). The effort multipliers as shown by EM are all multiplied together. In addition, B is a constant, for COCOMO II the value is 0.91. Finally, the five scale factors (SF) are summed together. The result is the effort in person-months.

3.6 Schedule

The amount of time to develop the software product is the schedule or project duration. The equation to estimate the project duration is shown in Equation 3.3. In Equation 3.3, C and D are constants, which for COCOMO II is 3.67 and 0.28 respectively. PM is the effort in person-months calculated from the previous section. SF is the summed scale factors. TDEV is the project duration in months.

\[
TDEV = C \times (PM)^F
\]

\[
where \ F = D + 0.2 \times 0.01 \times \sum_{j=1}^{5} SF_j
\]

Equation 3.3 Schedule Estimation
3.7 Staffing

COCOMO II calculates staffing by taking the effort estimate divided by the schedule estimate.

\[ \text{Staffing} = \frac{PM}{TDEV} \]

Equation 3.4 Staffing Equation

One underlying assumption of COCOMO II is that higher team size results in lower productivity, but the direct effects of team size are not specifically modeled by COCOMO II. In addition, there is no support in COCOMO II for increasing or decreasing the staffing estimate. Team size is thought to be indirectly captured by factors already modeled, such as project size, (Conte, Dunsmore et al. 1986) but not explicitly modeling team size leaves no support for changing the staffing estimate. Briand et al (1998) states that after product size, team size is the strongest factor influencing project cost, but COCOMO II treats it as a dependent variable rather than an independent variable. This dissertation will address this large weakness.

3.8 COCOMO II Overview

COCOMO II provides a rich structure to characterize software projects though scale factors and effort multipliers. Also using lines of code or function points as a measure of size, a software project can be parameterized in detail. COCOMO II provides a detailed estimation of product activity though the work breakdown structure. The effort and schedule estimate along with the work breakdown structure will be used as inputs to improve the issues raised about staffing to build a new cost estimation model.
CHAPTER 4 COMMUNICATION OVERHEAD

I will pay more for the ability to deal with people than any other ability under the sun.  
- John D. Rockefeller

4.1 Introduction

“Professional programmers spend considerable time communicating with others in their organization, both individually and as part of a group. Thus the analysis of communication problems—for example, groups not realizing they are even supposed to communicate, misunderstandings about a shared issue, conflicting views from different groups, or changes in project personnel—is a key element in understanding how to better support the software development process” (Rosson 1996 p.194).

Just as there are losses in productivity due to lack of motivation, there are also losses because of communication. This loss is commonly called communication overhead. This chapter details the derivation of communication overhead used in this dissertation.

4.2 Communication Overhead Definition

Communication overhead is the “average team member’s drop in productivity below his nominal productivity as a result of team communication, where communication includes verbal communication, documentation, and any additional work, such as that due to interfaces” (Abdel-Hamid and Madnick 1991 p.93). Such communication overhead is not needed when software is developed by a single person, but as additional people are added to a team, the communication overhead rises.
“... it is necessary that each individual spend part of his time communication with each of the other team members. For example, the designer must confer with the coder to resolve any questions the code may have about the design; both of these must talk to the individual testing the code to give him the benefit of their experience with the program; each of these must talk to the documentor to assure that the documentation is proper and complete; and so on” (Tausworthe 1977).

As more people are added to a software development project, the number of possible communication paths grows not linearly, but polynomially. Since communication paths are a function of communication overhead, communication overhead also grows exponentially. Brooks detailed this relationship, saying as the team size (n) increases, communication overhead increases in proportion to $n^2$ (Brooks 1975; Abdel-Hamid and Madnick 1991). Brooks argued for the drop in productivity as team size increases stating the following:

1. As the team size increase, there is greater need to coordinate the activities of the group, thus increasing overhead at the expense of code production.
2. As members are added to a team, the new members must acquaint themselves with the overall project design and with previously completed work before they can begin to contribute to the project. (Conte, Dunsmore et al. 1986 p.258)

The number of communication paths that exist in a team with n people is shown in Equation 4.1.

$$\text{CommunicationPaths} = \frac{(n)(n-1)}{2}$$

Equation 4.1 Communication Paths for n People

If a group of 30 people were in a team, there is a possibility of $\frac{30 \times (30-1)}{2}$, or 435 communication paths between all people. Abdel-Hamid found that for a team of 30
people, the communication overhead is more than 50%. Out of an 8-hour day, more than 4 hours of the day will be spent communicating. Typical communication activities include meetings, phone calls, documentation, and artifact reviews.

During software development, if needed communication is not done, problems will arise from misunderstandings and will eventually have to be corrected. On the other hand, communication that is not needed can also occur, leading to no foreseeable benefit to the software development project.

Since communication overhead can take up such as large percentage of time during software development, some people suggest small, agile teams, that consist of no more than 10 people (Paulk 2001). With small teams, communication overhead is reduced, leading to more efficient software development. However, some software projects cannot be completed in a reasonable period with 10 people or less. In these projects, communication and communication overhead play an important role in the project success.

Instead of limiting the number of team members on a software development project, another method is to implement a process structure that limits communication paths between individuals. By breaking the project team into smaller groups and restricting the number of communication paths between team members, communication overhead can be reduced.

4.3 Quantifying Communication Overhead

Abdel-Hamid quantified the relationship between team size and communication overhead. Table 4-1 shows the communication overhead percentage for given team sizes.
Table 4-1 Communication Overhead Percentage as a Given Team Size

(Abdel-Hamid and Madnick 1991 p.94)

To find a team size not listed, interpolation is used between the two closest points.

To provide a better way of finding a team size not listed, mapping team size to number of communication paths using Equation 4.1 provides more detail. Table 4-2 shows the addition of adding communication paths to Table 4-1.

Table 4-2 Communication Paths Added To Communication Overhead

Conducting a regression with communication overhead being the dependent variable and communication paths being the independent variable leads to Equation 4.2.

\[
CommunicationOverhead = 0.001248269 \times CommunicationPaths
\]

Equation 4.2 Prediction Equation for Communication Overhead

The regression equation has very high explanatory power with \( R^2 \) being greater than 0.99. Having a \( R^2 \) at 1.00 is the maximum possible. Therefore the equation is very good at modeling Communication Overhead based on Communication Paths.

Once more than 30 people are on a team, the empirical evidence on communication overhead is sparse. In order not to estimate with Equation 4.2 beyond the
data that the equation was modeled, any teams bigger than 30 people or 435 paths will assume a communication overhead of 54%.

4.4 Cooperating Program Model - COPMO

Team size was a major factor whose significance was not fully analyzed therefore Thebaut (Thebaut and Shen 1984) proposed a software cost estimation model assuming additional effort is needed for when there is large number of people in teams on a project. The equation developed assumed that staff provides diseconomies of scale rather than software size.

\[
Effort = a + bS + cP^d
\]

Equation 4.3 COPMO Equation

In the previous equation, a, b, c, and d are constants that need to be determined from empirical data, S is the program size in thousands of lines of code, and P is the average personnel level (staff) over the life of the project.

Communication overhead is modeled with the last term in the equation. By replacing c with 1.5, and d with 2.0, the communication overhead follows Brooks’ suggestion. Calculating the last term of the equation with team size produces the following table.

<table>
<thead>
<tr>
<th>Team Size</th>
<th>Communication Overhead</th>
<th>Increase in % of Overhead</th>
<th>COPMO</th>
<th>Increase in % of COPMO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.00</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>0.02</td>
<td>-</td>
<td>37.5</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>0.06</td>
<td>3.00</td>
<td>150</td>
<td>3.00</td>
</tr>
<tr>
<td>15</td>
<td>0.14</td>
<td>1.25</td>
<td>337.5</td>
<td>1.25</td>
</tr>
<tr>
<td>20</td>
<td>0.24</td>
<td>0.78</td>
<td>600</td>
<td>0.78</td>
</tr>
<tr>
<td>25</td>
<td>0.38</td>
<td>0.56</td>
<td>937.5</td>
<td>0.56</td>
</tr>
<tr>
<td>30</td>
<td>0.54</td>
<td>0.44</td>
<td>1350</td>
<td>0.44</td>
</tr>
</tbody>
</table>

Table 4-3 COPMO and Communication Overhead
Table 4-3 shows that the increase in COPMO for a given team size has the same increase in communication overhead. This further provides evidence of the $n^2$ relationship between team size and communication overhead.

4.5 Communication Overhead Contributions

By including communication overhead to software cost estimation models, the work of Thebaut and Abdel-Hamid can be continued. Thebaut was interested in looking at the average staffing level throughout the project where Abdel-Hamid was interested in the instantaneous staffing level during the project. Neither of the researchers looked at how the structure of the project team interacted with communication. The regression equation developed here along with the equation for the number of communication paths given a certain number of people will be used to create a new software cost estimation model.
5.1 Introduction

This chapter details the creation of a new software cost estimation model based on COCOMO II, software development process structure, and team size. The outputs of COCOMO II which include effort, schedule and project duration, and the work breakdown structure are summarized, and then are further explained with various process structures: one-tier, two-tier, or three-tier, along with team size to improve the estimates for effort and schedule. A new metric is created called staff loading that quantifies what percentage of time staff is actively working through development. Different completed software development projects are run through the new software cost estimation tool to illustrate the impact of the software handoff on software development.
5.2 Model Overview

The new software cost estimation model performs five steps in order to create new estimates. The five steps are initially summarized than are further explained throughout the chapter.

The first step in the new software cost estimation is to calculate the outputs from COCOMO II. COCOMO II includes many project differences in its cost estimation model. The differences in projects allow COCOMO II to yield a scale factor and an effort multiplier for each particular project. Along with the project size, scale factor, and effort multiplier, COCOMO II can produce an estimate for effort, duration, staff size, and a work breakdown structure. Chapter 3 describes COCOMO II and the calculations formed in detail.

The second step is preparing the work breakdown structure and effort estimate to be input into the new cost estimation model. The effort estimate is adjusted to include the effects of the planning and requirements phase. In addition, the work breakdown structure is mapped into the different process structures. The work breakdown structure provides information about how long different software development activities will take. The process structure explains which group conducts the particular software development activities. Combining the process structure with the work breakdown structure will inform the model to which group does how much work.

The third step is to include staffing as an independent variable. With staffing moving from a dependent variable in COCOMO II to an independent variable in the new
software cost estimation model, the staffing for each process structure must be included. Populating the process structure with staffing information is thus the third step.

The fourth step is to calculate coordination and communication costs based on the staffing and the combined work breakdown structure and process structure. A new effort estimate will be created.

The fifth step is to calculate a new schedule estimate based on the new effort estimate along with the staffing and process structure. Many other software cost estimation models have difficulty in estimating project duration, but with this new model, the estimate is rather straightforward.
Calculate Outputs from COCOMO II
Work breakdown Structure
Effort Estimate

Step 1

Adjust the work breakdown structure
and effort estimate to include the
planning and requirements phase

Step 2a

Convert the work breakdown
structure into a % of effort of total
rather than % of phases

Step 2b

Map the work breakdown structure
into the three different process
structures

Step 2c

Populate the three different process
structures with staffing information

Step 3

Calculate new effort estimate based
on coordination and communication
costs

Step 4

Calculate new schedule for the three
different process structures with the
staffing information

Step 5

Figure 5-1 Model Overview

5.3 Extended Example Information

An extended example is used throughout this chapter to show the workings of the model. For the extended example, a medium sized project consisting of 40 KSLOC is used. The default scale multipliers (1.12) and effort multipliers (1.00) are also used. Working with this example will show how the five different steps of the model create a new estimate for both effort and schedule.
5.4 Using the COCOMO II Outputs

The first step that is conducted in the new cost estimation model is to calculate the needed outputs from COCOMO II. Chapter 3 provides details on how COCOMO II estimates effort, schedule, staffing, and a work breakdown structure. This cost estimation model specifically needs the effort estimate and the work breakdown structure from COCOMO II. COCOMO II provides the effort estimate in man-months and derives the work breakdown structure from tables Table 5-1, Table 5-2, Table 5-3, and Table 5-4. Table 5-1 is the planning and requirements phase of software development. This phase is where the software specification is created. Table 5-2 is the product design phase. This phase is where the requirements specification is turned into a valid software design. Table 5-3 is the programming phase. This phase is where the software design is implemented into code. Finally, Table 5-4 is the integration and test phase. This phase is where the developed software is tested. All the numbers in the work breakdown structure represent percentages.
### Table 5-1 Plans and Requirements Activity Distribution

<table>
<thead>
<tr>
<th>Size Exponent</th>
<th>E = 1.05</th>
<th>E = 1.12</th>
<th>E = 1.20</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size</strong></td>
<td>S, I, M, L</td>
<td>S, I, M, L, VL</td>
<td>S, I, M, L, VL</td>
</tr>
<tr>
<td><strong>Overall Phase Percentage</strong></td>
<td>6</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td><strong>Requirements Analysis</strong></td>
<td>46</td>
<td>48</td>
<td>47</td>
</tr>
<tr>
<td><strong>Product Design</strong></td>
<td>20</td>
<td>16</td>
<td>16.5</td>
</tr>
<tr>
<td><strong>Programming</strong></td>
<td>3</td>
<td>2.5</td>
<td>3.5</td>
</tr>
<tr>
<td><strong>Test Planning</strong></td>
<td>3</td>
<td>2.5</td>
<td>3</td>
</tr>
<tr>
<td><strong>V &amp; V</strong></td>
<td>6</td>
<td>6</td>
<td>6.5</td>
</tr>
<tr>
<td><strong>Project Office</strong></td>
<td>15</td>
<td>15.5</td>
<td>14.5</td>
</tr>
<tr>
<td><strong>CM / QA</strong></td>
<td>2</td>
<td>3.5</td>
<td>3</td>
</tr>
<tr>
<td><strong>Manuals</strong></td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

S: 2 KSLOC; I: 8 KSLOC; M: 32 KSLOC; L: 128 KSLOC; VL: 512 KSLOC

### Table 5-2 Product Design Activity Distribution

<table>
<thead>
<tr>
<th>Size Exponent</th>
<th>E = 1.05</th>
<th>E = 1.12</th>
<th>E = 1.20</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size</strong></td>
<td>S, I, M, L</td>
<td>S, I, M, L, VL</td>
<td>S, I, M, L, VL</td>
</tr>
<tr>
<td><strong>Overall Phase Percentage</strong></td>
<td>16</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td><strong>Requirements Analysis</strong></td>
<td>40</td>
<td>41</td>
<td>41</td>
</tr>
<tr>
<td><strong>Product Design</strong></td>
<td>14</td>
<td>12</td>
<td>12.5</td>
</tr>
<tr>
<td><strong>Programming</strong></td>
<td>5</td>
<td>4.5</td>
<td>5</td>
</tr>
<tr>
<td><strong>Test Planning</strong></td>
<td>6</td>
<td>6</td>
<td>6.5</td>
</tr>
<tr>
<td><strong>V &amp; V</strong></td>
<td>11</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td><strong>CM / QA</strong></td>
<td>2</td>
<td>3</td>
<td>2.5</td>
</tr>
<tr>
<td><strong>Manuals</strong></td>
<td>7</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

S: 2 KSLOC; I: 8 KSLOC; M: 32 KSLOC; L: 128 KSLOC; VL: 512 KSLOC
Using the information about the extended example, COCOMO II calculates the effort to be 169.9 man-months for the given 40 KSLOC project. With the size exponent being \( E = 1.12 \) in the extended example, and the since 40 KSLOC is closer to 32 KSLOC rather than 128 KSLOC, the M column under the \( E = 1.12 \) section is used. The correct numbers for the plans and requirements phase that are used for the extended example are highlighted in Table 5-5.
Table 5-5 Plans and Requirements Phase for a 40 KSLOC project

The other three tables are also selected to create a complete work breakdown structure for the extended example. The complete work breakdown structure is shown in Table 5-6.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Plans and Requirement</th>
<th>Product Design</th>
<th>Programming Activity</th>
<th>Integration and Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement &amp; Analysis</td>
<td>46</td>
<td>12.5</td>
<td>4</td>
<td>2.5</td>
</tr>
<tr>
<td>Product Design</td>
<td>17</td>
<td>41</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Programming</td>
<td>4.5</td>
<td>13</td>
<td>56.5</td>
<td>37</td>
</tr>
<tr>
<td>Test Planning</td>
<td>3.5</td>
<td>5.5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>V &amp; V</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>29.5</td>
</tr>
<tr>
<td>Project Office</td>
<td>13.5</td>
<td>11</td>
<td>6.5</td>
<td>7.5</td>
</tr>
<tr>
<td>Quality Assurance</td>
<td>3</td>
<td>2.5</td>
<td>6.5</td>
<td>8</td>
</tr>
<tr>
<td>Manuals</td>
<td>5.5</td>
<td>7.5</td>
<td>5.5</td>
<td>7</td>
</tr>
<tr>
<td>Phase Totals</td>
<td>7</td>
<td>17</td>
<td>58</td>
<td>25</td>
</tr>
</tbody>
</table>

Table 5-6 Complete Work Breakdown Structure for Extended Example

A notation is needed to represent the cells in the previous table. Each phase will be denoted by an abbreviation for the phase. Plans and Requirements is PR, Product Design is PD, Programming Activity is PA, and Integration and Test is IT. A subscript is used to denote the activity rows. Requirements & Analysis is row 1, with each Manuals being row 8. The activity is added as a subscript to the phase to get a variable in the form: \( \text{Phase}_{\text{Activity}} \). The phase total for each phase is notated by the given phase with total.
as the subscript. Table 5-7 shows the complete enumeration of the work breakdown structure using the described notation.

<table>
<thead>
<tr>
<th>Phases</th>
<th>Plans and Requirement</th>
<th>Product Design</th>
<th>Programming Activity</th>
<th>Integration and Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activities</td>
<td>Equ</td>
<td>Ex</td>
<td>Equ</td>
<td>Ex</td>
</tr>
<tr>
<td>Requirement &amp; Analysis</td>
<td>$PR_1$</td>
<td>46</td>
<td>$PD_1$</td>
<td>12.5</td>
</tr>
<tr>
<td>Product Design</td>
<td>$PR_2$</td>
<td>17</td>
<td>$PD_2$</td>
<td>41</td>
</tr>
<tr>
<td>Programming</td>
<td>$PR_3$</td>
<td>4.5</td>
<td>$PD_3$</td>
<td>13</td>
</tr>
<tr>
<td>Test Planning</td>
<td>$PR_4$</td>
<td>3.5</td>
<td>$PD_4$</td>
<td>5.5</td>
</tr>
<tr>
<td>V &amp; V</td>
<td>$PR_5$</td>
<td>7</td>
<td>$PD_5$</td>
<td>7</td>
</tr>
<tr>
<td>Project Office</td>
<td>$PR_6$</td>
<td>13.5</td>
<td>$PD_6$</td>
<td>11</td>
</tr>
<tr>
<td>Quality Assurance</td>
<td>$PR_7$</td>
<td>3</td>
<td>$PD_7$</td>
<td>2.5</td>
</tr>
<tr>
<td>Manuals</td>
<td>$PR_8$</td>
<td>5.5</td>
<td>$PD_8$</td>
<td>7.5</td>
</tr>
<tr>
<td>Phase Total</td>
<td>$PR_{TOTAL}$</td>
<td>7</td>
<td>$PD_{TOTAL}$</td>
<td>17</td>
</tr>
</tbody>
</table>

Table 5-7 Work breakdown structure mapping

5.5 Modeling the Work Breakdown Structure in Process Structures

The first step in mapping the work breakdown structure (shown as Step 2a in Figure 5-1) to different process structures is to adjust the effort estimate to include the plans and requirements phase. From Table 5-7 $PD_{TOTAL} + PA_{TOTAL} + IT_{TOTAL} = 100$.

COCOMO II’s effort output only includes the product design, programming activity, and integration and test phases. To include the plans and requirements phase, $PR_{TOTAL}$ must be added to the COCOMO II effort estimate. To include the plans and requirement phase the following equation is used:

$$Effort_{Total} = (1 + \frac{PR_{TOTAL}}{100}) \times COCOMOIIEffortEstimate$$
Along with adjusting the effort to include the plans and requirements phase, the work breakdown structure must be changed so \( PR_{TOTAL} + PD_{TOTAL} + PA_{TOTAL} + IT_{TOTAL} = 100 \).

The algorithm to convert the four phase totals to equal 100 is shown below:

\[
X = PR_{TOTAL} + PD_{TOTAL} + PA_{TOTAL} + IT_{TOTAL}
\]

\[
PR_{TOTAL} = \frac{PR_{TOTAL}}{X}
\]

\[
PD_{TOTAL} = \frac{PD_{TOTAL}}{X}
\]

\[
PA_{TOTAL} = \frac{PA_{TOTAL}}{X}
\]

\[
IT_{TOTAL} = 100 - PR_{TOTAL} - PD_{TOTAL} - PA_{TOTAL}
\]

The work breakdown structure four phase totals now sum to 100, but adding \( PR_1 \) though \( PR_8 \) equals 100 instead of \( PR_{TOTAL} \). The activities in each phase are adjusted by the phase total to indicate the percentage of work that activity will take place for the whole project rather than just the phase. By multiplying the activities in each phase with the phase total, the conversion is made.
Let $I$ be an Index Set of Activities where $|I| = 8$.

$$PR_i = PR_i \times \frac{PR_{TOTAL}}{100} \quad i \in I$$

$$PD_i = PD_i \times \frac{PD_{TOTAL}}{100} \quad i \in I$$

$$PA_i = PA_i \times \frac{PA_{TOTAL}}{100} \quad i \in I$$

$$IT_i = IT_i \times \frac{IT_{TOTAL}}{100} \quad i \in I$$

<table>
<thead>
<tr>
<th>Phases</th>
<th>Plans and Requirement</th>
<th>Product Design</th>
<th>Programming Activity</th>
<th>Integration and Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activities</td>
<td>Equ</td>
<td>Ex</td>
<td>Equ</td>
<td>Ex</td>
</tr>
<tr>
<td>Requirement &amp; Analysis</td>
<td>$PR_1$</td>
<td>2.99</td>
<td>$PD_1$</td>
<td>1.9875</td>
</tr>
<tr>
<td>Product Design</td>
<td>$PR_2$</td>
<td>1.105</td>
<td>$PD_2$</td>
<td>6.519</td>
</tr>
<tr>
<td>Programming</td>
<td>$PR_3$</td>
<td>0.2925</td>
<td>$PD_3$</td>
<td>2.067</td>
</tr>
<tr>
<td>Test Planning</td>
<td>$PR_4$</td>
<td>0.2275</td>
<td>$PD_4$</td>
<td>0.8745</td>
</tr>
<tr>
<td>V &amp; V</td>
<td>$PR_5$</td>
<td>0.455</td>
<td>$PD_5$</td>
<td>1.113</td>
</tr>
<tr>
<td>Project Office</td>
<td>$PR_6$</td>
<td>0.8775</td>
<td>$PD_6$</td>
<td>1.749</td>
</tr>
<tr>
<td>Quality Assurance</td>
<td>$PR_7$</td>
<td>0.195</td>
<td>$PD_7$</td>
<td>0.3975</td>
</tr>
<tr>
<td>Manuals</td>
<td>$PR_8$</td>
<td>0.3575</td>
<td>$PD_8$</td>
<td>1.1925</td>
</tr>
<tr>
<td>Phase Total</td>
<td>$PR_{TOTAL}$</td>
<td>6.5</td>
<td>$PD_{TOTAL}$</td>
<td>15.9</td>
</tr>
</tbody>
</table>

Table 5-8 Adjusted Work Breakdown Structure

The adjusted work breakdown structure now reflects a project that will have four phases of development. The next step is to map the adjusted work breakdown structure into the three different process structures. With three different process structures, there will be three different mappings.
There are three types of places that the thirty-two different cells can be mapped into. A cell can be mapped into a main group box. This mapping represents the fact that only one group will do the work without working with other groups. Examples include the implementation/unit testing group writing the software, and the design group, designing the software. Another place to map is between two groups. This mapping represents a handoff. The implementation/unit testing team giving the code to the testing group is an example of a handoff. The third mapping is general overhead. Cells that do not map into the first two mappings belong in the third. Project management is a good example of a mapping that belongs in the third group.

5.6 Mapping of the Three-Tier Process Structure

The first structure to be mapped is the Three-Tier Structure. The three-tier structure provides five different main boxes and includes requirements, design, implementation/unit testing, integration testing, and customer acceptance. Figure 5-2 shows the mapping of the work breakdown structure into the three-tier process structure. Each individual mapping is discussed in this section.
Figure 5-2 Effort Breakdown for Three-Tier

$PR_1$ is the requirements and analysis activity of the plans and requirements phase. This phase is where the initial requirements of the systems are developed from the customers. $PR_1$ is conducted by the requirements team therefore is mapped to the requirements box in the three-tier process structure. The requirements team also start to plan for quality assurance at the beginning of the project, $PR_7$ is also mapped to the requirements team process structure.

While the requirements are being collected, the initial customer acceptance test plan can be created ($PR_8$). Part of this test plan is the manual for the system.
After the initial requirements are created, the requirements must be handed over to the design group. Product Design and Programming done in the plans and requirements phase is very high level usually consisting of initial prototypes that will be eventually discarded. The requirements team transfers to the design team the requirements document along with the initial product design \((PR_2)\) and initial programming \((PR_3)\).

With the requirements document from the requirements group, the design group can start on the designing the system, \(PD_1\). Any questions for the requirements group or updates to the requirements will occur through the requirements and design group handoff; \(PD_1\) represents this activity. The design group will also conduct the initial test planning \((PR_4)\) and verification and validation activities \((PR_5)\).

Once the design is created, two major activities occur. First, the Integration test plan \((PD_4, PD_5)\) is handed off from the design group to the integration testing team. Second, the detailed design \((PD_3)\) created by the design group is handed off to the implementation/unit testing group.

If there are any questions about the requirements when creating the customer acceptance test plan, \(PD_7\) maps the extra quality assurance activity. The quality assurance activity could cause changes in the requirements though. But at this point the plans and requirement and product design phase is complete. The programming activity phase is ready to start.
The implementation/unit testing group starts developing the code \((PA_3)\). Changes to the requirements propagate through the requirements and design groups \((PA_i)\) and through the design and implementation/unit testing group \((PA_j)\). With the detailed design already complete, the design group continues working on the integration test plan \((PA_k, PA_m, PA_n)\).

At this point the programming phase is complete and the final phase, integration and test start. Any final changes to the requirements are propagated through to the design group \((IT_i)\) and the implementation/unit test group \((IT_j)\). The handoff of code from the integration/unit testing group to the integration testing \((IT_3)\) is a large task. In this activity all rework is done. Testing can commence once the code is given to the integration team. The final integration test plan \((IT_4)\) is conducted by the integration testing team \((IT_5)\). Once the code is tested, the integrated system is delivered for the customer acceptance team for testing and delivery \((IT_7)\).

In the three-tier structure, there was no specific place to map the project office activities \((PR_6, PD_6, PA_6, IT_6)\) and manuals \((PD_8, PA_8, IT_8)\). These activities are mapped as general overhead that will add to the completion of all software development activities.

5.7 Mapping of the Two-Tier Process Structure

The mapping for a Two-Tier Structure is next. Three main boxes are used. By using the mapping for the three-tier process structure and combining the requirements and design team to create the requirements/design and combining the integration testing
and customer acceptance group to create the integration/customer acceptance group the two-tier process structure is formed. Figure 5-3 shows the two-tier process structure.

Figure 5-3 Two-Tier Effort Breakdown

5.8 Mapping of the One-Tier Process Structure

The final process structure is the one-tier process structure. Since there is only one place for the work to be done, the one and only box contains all the mappings. This can also be seen by combining the requirements/design, integration/customer acceptance, and implementation/unit testing into one box.
5.9 Populating Staffing into the Process Structures

Unlike COCOMO II, the new cost estimation model will be able to include the effects of changing the staff size. Staffing is now modeled as an independent variable, rather than a dependent variable that is the result of effort divided by schedule. The team size can be adjusted in the model from a minimum of one person to however many is wanted. The project manager is no longer limited in knowing the staffing must exactly match what COCOMO II suggests or the estimate will not be valid. If COCOMO II requires ten people, but only seven are available, simply putting in the seven people will adjust the scheduled project duration.

Changing staffing for a team changes both intra-group communication and inter-group coordination. Intra-group communication is calculated directly from the size of the team. Bigger teams are going to need more intra-group communication. A staff meeting with five people will take more effort than a meeting with just three people. Since the amount of communication overhead for a given team size is known, intra-group communication is well understood. Inter-group coordination occurs when two different
teams need to coordinate information. Having bigger teams results in more inter-group coordination in addition to intra-group communication.

Three different process structures are presented in this dissertation. The one-tier process structure has only one team. The two-tier process structure has three teams; the requirements/design team, the implementation/unit testing team, and the integration/customer acceptance team. The three-tier process structure has five teams: requirements, design, implementation/unit testing, integration testing, and customer acceptance teams.

A method is needed to refer to the different teams in the three process structures. Step three of Figure 5-1 is to populate the three different process structures with staffing information.

![Diagram of Three-Tier Model]

Figure 5-5 Three-Tier Model
The three-tier process structure is first enumerated. With the three-tier model, five different variables that represent the number of staff in each team are needed. From the three-tier model, the following variables are created: $\text{TeamSize}_{\text{Requirements}}$, $\text{TeamSize}_{\text{Design}}$, $\text{TeamSize}_{\text{Implementation}}$, $\text{TeamSize}_{\text{IntegrationTesting}}$ and $\text{TeamSize}_{\text{AcceptanceTest}}$.

Figure 5-6 Two-Tier Model

Next, the two-tier process structure is enumerated. The variables to represent the different teams are

$\text{TeamSize}_{\text{RequirementsDesign}}$, $\text{TeamSize}_{\text{Implementation}}$, $\text{TeamSize}_{\text{IntegrationCustomerAcceptance}}$.
Lastly, the one-tier structure is enumerated. The variable to represent the single team is $TeamSize_{OneTeam}$.

At this point, each team in each of the three process structures is given a variable name, and these variables names are used in the next step to calculate effort. Based on the extended example described earlier in this chapter, the process structures are going to be populated. Table 5-9 shows a possible method of populating the process structure teams with staff.
<table>
<thead>
<tr>
<th>Process Structure</th>
<th>Team Name</th>
<th>Team Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Three-Tier</td>
<td>$\text{TeamSize}_{\text{Requirements}}$</td>
<td>2</td>
</tr>
<tr>
<td>Three-Tier</td>
<td>$\text{TeamSize}_{\text{Design}}$</td>
<td>2</td>
</tr>
<tr>
<td>Three-Tier</td>
<td>$\text{TeamSize}_{\text{Implementation}}$</td>
<td>2</td>
</tr>
<tr>
<td>Three-Tier</td>
<td>$\text{TeamSize}_{\text{IntegrationTesting}}$</td>
<td>2</td>
</tr>
<tr>
<td>Three-Tier</td>
<td>$\text{TeamSize}_{\text{AcceptanceTest}}$</td>
<td>2</td>
</tr>
<tr>
<td>Two-Tier</td>
<td>$\text{TeamSize}_{\text{RequirementsDesign}}$</td>
<td>3</td>
</tr>
<tr>
<td>Two-Tier</td>
<td>$\text{TeamSize}_{\text{Implementation}}$</td>
<td>3</td>
</tr>
<tr>
<td>Two-Tier</td>
<td>$\text{TeamSize}_{\text{IntegrationCustomerAcceptance}}$</td>
<td>4</td>
</tr>
<tr>
<td>One-Tier</td>
<td>$\text{TeamSize}_{\text{OneTeam}}$</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 5-9 Example Team Sizes

5.10 Effort Calculation

Step 4 of Figure 5-1 is to calculate the effort needed for each process structure based on the staffing information.

5.11 Three-Tier Structure

Describing how the estimation for the three-tier structure is implemented is discussed in this section. The three-tier structure has five different places to put staff members. Staff members can be placed in requirements, design, implementation and unit test, integration test, or acceptance test. The algorithm assumes at least one staff member is assigned to each functional group, but any number of staff members can be present.

The communication overhead for each team size is calculated based on Equation 4.1 and Equation 4.2.
\[ CommunicationPaths = \frac{(n)(n-1)}{2} \]

Equation 5.1 Communication Paths for n People

\[ CommunicationOverhead = 0.001248269 \times CommunicationPaths \]

Equation 5.2 Prediction Equation for Communication Overhead

For example, for a team of 10 people, the communication overhead would be 0.056. For a group of 10 people there will be almost 6% more effort required to complete the task than if a single person did it alone. The next step is multiply the original COCOMO II effort estimate by each work breakdown cell to get a numerical estimation of effort in each cell.

The phases of the work breakdown structure must be mapped into the group that does the work. As shown in Figure 5-8, the mapping from the work breakdown into function groups is shown. Arcs between groups are shared tasks and will affect the combination of the teams. The box labeled as general overhead are activities that are not particularly done by any group. As more total staff are added to the project, this overhead grows.
To calculate effort, two steps are required. First, the intra-group communication is calculated. Then, the inter-group coordination is required. To calculate intra-group communication, the communication overhead is calculated for each group based on the staff size of the group. The total amount of work that is conducted in the group is multiplied by the communication overhead.

The following equation takes a group staff size, and calculates the communication overhead that will result with the given staff size.

Figure 5-8 Effort Breakdown for Three-Tier
Using the Communication Effort Multiplier equation, the effort increase due to intra-group communication is calculated.

\[
CEM(n) = 1 + (0.001248269 \times \frac{(n)(n-1)}{2})
\]

Equation 5.3 Effort Multipliers Due To Intra-Group Communication

Inter-group Coordination:

\[
\begin{align*}
\text{EffortMultiplier}_{\text{Requirements}} &= CEM \left( TeamSize_{\text{Requirements}} \right) \\
\text{EffortMultiplier}_{\text{Design}} &= CEM \left( TeamSize_{\text{Design}} \right) \\
\text{EffortMultiplier}_{\text{Implementation}} &= CEM \left( TeamSize_{\text{Implementation}} \right) \\
\text{EffortMultiplier}_{\text{IntegrationTesting}} &= CEM \left( TeamSize_{\text{IntegrationTesting}} \right) \\
\text{EffortMultiplier}_{\text{AcceptanceTest}} &= CEM \left( TeamSize_{\text{AcceptanceTest}} \right)
\end{align*}
\]

\[
\begin{align*}
\text{CommunicationEffortMultiplier}_{\text{Requirements&Design}} &= \text{EffortMultiplier}_{\text{Requirements}} + \text{EffortMultiplier}_{\text{Design}} \\
\text{CommunicationEffortMultiplier}_{\text{Design&Implementation}} &= \text{EffortMultiplier}_{\text{Design}} + \text{EffortMultiplier}_{\text{Implementation}} \\
\text{CommunicationEffortMultiplier}_{\text{Implementation&IntegrationTesting}} &= \text{EffortMultiplier}_{\text{Implementation}} + \text{EffortMultiplier}_{\text{IntegrationTesting}} \\
\text{CommunicationEffortMultiplier}_{\text{IntegrationTesting&AcceptanceTesting}} &= \text{EffortMultiplier}_{\text{IntegrationTesting}} + \text{EffortMultiplier}_{\text{AcceptanceTest}} \\
\text{CommunicationEffortMultiplier}_{\text{Requirements&AcceptanceTesting}} &= \text{EffortMultiplier}_{\text{Requirements}} + \text{EffortMultiplier}_{\text{AcceptanceTest}} \\
\text{CommunicationEffortMultiplier}_{\text{Design&IntegrationTesting}} &= \text{EffortMultiplier}_{\text{Design}} + \text{EffortMultiplier}_{\text{IntegrationTesting}}
\end{align*}
\]
EffortMultiplier_{All} = CommunicationEffortMultiplier(TeamSize_{Requirements} + TeamSize_{Design} + TeamSize_{Implementation} + TeamSize_{IntegrationTesting} + TeamSize_{AcceptanceTest})

Effort_{Requirements} = PR_{1} + PR_{7}
Effort_{Design} = PR_{4} + PR_{5} + PD_{2}
Effort_{Implementation} = PA_{3}
Effort_{IntegrationTesting} = IT_{5}
Effort_{AcceptanceTest} = 0
Effort_{Requirements&Design} = PR_{2} + PR_{3} + PD_{1} + PA_{1} + IT_{1}
Effort_{Design&Implementation} = PD_{3} + PA_{3} + IT_{2}
Effort_{Implementation&IntegrationTesting} = IT_{7}
Effort_{IntegrationTesting&AcceptanceTesting} = IT_{7}
Effort_{Requirements&AcceptanceTesting} = PR_{8} + PD_{7}
Effort_{Design&IntegrationTesting} = PD_{4} + PA_{4} + IT_{4} + PD_{5} + PA_{5} + PA_{7}
Effort_{All} = PR_{6} + PD_{6} + PA_{6} + IT_{6} + PD_{8} + PA_{8} + IT_{8}

Equation 5.4 Tier-Three Effort Mapping Equations
\[ \text{TierThreeEffortMultiplier} = \]
\[ \text{EffortMultiplier}_{\text{Requirements}} \times \text{Effort}_{\text{Requirements}} + \]
\[ \text{EffortMultiplier}_{\text{Design}} \times \text{Effort}_{\text{Design}} + \]
\[ \text{EffortMultiplier}_{\text{Implementation}} \times \text{Effort}_{\text{Implementation}} + \]
\[ \text{EffortMultiplier}_{\text{IntegrationTesting}} \times \text{Effort}_{\text{IntegrationTesting}} + \]
\[ \text{EffortMultiplier}_{\text{AcceptanceTest}} \times \text{Effort}_{\text{AcceptanceTest}} + \]
\[ \text{EffortMultiplier}_{\text{Requirements&Design}} \times \text{Effort}_{\text{Requirements&Design}} + \]
\[ \text{EffortMultiplier}_{\text{Design&Implementation}} \times \text{Effort}_{\text{Design&Implementation}} + \]
\[ \text{EffortMultiplier}_{\text{Implementation&IntegrationTesting}} \times \text{Effort}_{\text{Implementation&IntegrationTesting}} + \]
\[ \text{EffortMultiplier}_{\text{IntegrationTesting&AcceptanceTesting}} \times \text{Effort}_{\text{IntegrationTesting&AcceptanceTesting}} + \]
\[ \text{EffortMultiplier}_{\text{Requirements&AcceptanceTesting}} \times \text{Effort}_{\text{Requirements&AcceptanceTesting}} + \]
\[ \text{EffortMultiplier}_{\text{Design&IntegrationTesting}} \times \text{Effort}_{\text{Design&IntegrationTesting}} + \]

Finally,

\[ \text{TierThreeEffortEstimate} = \text{TierThreeEffortMultiplier} \times \text{COCOMOII Effort Estimate} \]

Schedule Calculation

To calculate the project duration the formula of effort divided by people is used. The TierThreeEffortEstimate from the previous section is used to represent the effort, and the number of people in a particular group is used for the people. Development effort that is not directly related to a particular team group is added as overhead. The equations that setup the schedule calculation are shown below:
Overhead_{Requirements} = TierThreeEffortEstimate \times (1 + PR_{6})

Overhead_{Design} = TierThreeEffortEstimate \times (1 + PD_{6} + PD_{8})

Overhead_{Programming} = TierThreeEffortEstimate \times (1 + PA_{6} + PA_{8})

Overhead_{Testing} = TierThreeEffortEstimate \times (1 + IT_{6} + IT_{8})

Time for Plans and Requirement Phase:

\begin{align*}
\text{Time}_{PR_{1}} &= \frac{PR_{1} \times \text{Overhead}_{Requirements}}{\text{TeamSize}_{Requirements}} \\
\text{Time}_{PR_{2}} &= \frac{PR_{2} \times \text{Overhead}_{Requirements}}{\text{TeamSize}_{Requirements} + \text{TeamSize}_{Design}} \\
\text{Time}_{PR_{3}} &= \frac{PR_{3} \times \text{Overhead}_{Requirements}}{\text{TeamSize}_{Requirements} + \text{TeamSize}_{Design}} \\
\text{Time}_{PR_{4}} &= \frac{PR_{4} \times \text{Overhead}_{Requirements}}{\text{TeamSize}_{Design}} \\
\text{Time}_{PR_{5}} &= \frac{PR_{5} \times \text{Overhead}_{Requirements}}{\text{TeamSize}_{Requirements}} \\
\text{Time}_{PR_{8}} &= \frac{PR_{8} \times \text{Overhead}_{Requirements}}{\text{TeamSize}_{Requirements} + \text{TeamSize}_{AcceptanceTest}}
\end{align*}

Time for Product Design Phase:

\begin{align*}
\text{Time}_{PD_{1}} &= \frac{PD_{1} \times \text{Overhead}_{Design}}{\text{TeamSize}_{Requirements} + \text{TeamSize}_{Design}} \\
\text{Time}_{PD_{2}} &= \frac{PD_{2} \times \text{Overhead}_{Design}}{\text{TeamSize}_{Design}} \\
\text{Time}_{PD_{3}} &= \frac{PD_{3} \times \text{Overhead}_{Design}}{\text{TeamSize}_{Design} + \text{TeamSize}_{Implementation}} \\
\text{Time}_{PD_{4}} &= \frac{PD_{4} \times \text{Overhead}_{Design}}{\text{TeamSize}_{Design} + \text{TeamSize}_{IntegrationTesting}} \\
\text{Time}_{PD_{5}} &= \frac{PD_{5} \times \text{Overhead}_{Design}}{\text{TeamSize}_{Design} + \text{TeamSize}_{IntegrationTesting}} \\
\text{Time}_{PD_{6}} &= \frac{PD_{6} \times \text{Overhead}_{Design}}{\text{TeamSize}_{Requirements} + \text{TeamSize}_{AcceptanceTest}}
\end{align*}
Time for Programming Activity Phase:

\[
\begin{align*}
\text{Time}_{\text{PA}_1} &= \frac{PA_1 \times \text{Overhead}_{\text{programming}}}{\text{TeamSize}_{\text{requirements}} + \text{TeamSize}_{\text{design}}} \\
\text{Time}_{\text{PA}_2} &= \frac{PA_2 \times \text{Overhead}_{\text{programming}}}{\text{TeamSize}_{\text{design}} + \text{TeamSize}_{\text{implementation}}} \\
\text{Time}_{\text{PA}_3} &= \frac{PA_3 \times \text{Overhead}_{\text{programming}}}{\text{TeamSize}_{\text{implementation}}} \\
\text{Time}_{\text{PA}_4} &= \frac{PA_4 \times \text{Overhead}_{\text{programming}}}{\text{TeamSize}_{\text{design}} + \text{TeamSize}_{\text{integration testing}}} \\
\text{Time}_{\text{PA}_5} &= \frac{PA_5 \times \text{Overhead}_{\text{programming}}}{\text{TeamSize}_{\text{design}} + \text{TeamSize}_{\text{integration testing}}} \\
\text{Time}_{\text{PA}_6} &= \frac{PA_6 \times \text{Overhead}_{\text{programming}}}{\text{TeamSize}_{\text{design}} + \text{TeamSize}_{\text{integration testing}}}
\end{align*}
\]

Time for Integration and Test Phase:

\[
\begin{align*}
\text{Time}_{\text{IT}_1} &= \frac{IT_1 \times \text{Overhead}_{\text{testing}}}{\text{TeamSize}_{\text{requirements}} + \text{TeamSize}_{\text{design}}} \\
\text{Time}_{\text{IT}_2} &= \frac{IT_2 \times \text{Overhead}_{\text{testing}}}{\text{TeamSize}_{\text{design}} + \text{TeamSize}_{\text{implementation}}} \\
\text{Time}_{\text{IT}_3} &= \frac{IT_3 \times \text{Overhead}_{\text{testing}}}{\text{TeamSize}_{\text{implementation}} + \text{TeamSize}_{\text{integration testing}}} \\
\text{Time}_{\text{IT}_4} &= \frac{IT_4 \times \text{Overhead}_{\text{testing}}}{\text{TeamSize}_{\text{design}} + \text{TeamSize}_{\text{integration testing}}} \\
\text{Time}_{\text{IT}_5} &= \frac{IT_5 \times \text{Overhead}_{\text{testing}}}{\text{TeamSize}_{\text{integration testing}}} \\
\text{Time}_{\text{IT}_6} &= \frac{IT_6 \times \text{Overhead}_{\text{testing}}}{\text{TeamSize}_{\text{integration testing}}}
\end{align*}
\]

To calculate the schedule, the tasks that are on the critical path are added together. Adding all the times will assume no parallelism, whereas only taking the longest task assumes complete parallelism. Normally, software development projects are somewhere between the two poles. By taking the tasks that are on the critical path leads to a schedule estimate.

\[
\text{TierThreeSchedule} = \\
\text{Time}_{\text{PR}_1} + \text{Time}_{\text{PR}_2} + \text{Time}_{\text{PD}_1} + \text{Time}_{\text{PD}_2} + \\
\text{Time}_{\text{PD}_3} + \text{Time}_{\text{PA}_3} + \text{Time}_{\text{IT}_3} + \text{Time}_{\text{IT}_5} + \text{Time}_{\text{IT}_6}
\]

98
5.12 Two-Tier Structure

Describing how the estimation for the two-tier structure is implemented is discussed in this section. The two-tier structure has three different places to put staff members. Staff members can be placed in Analysis and Design group, the Implementation group or the System Testing group. The algorithm assumes at least one staff member is assigned to each functional group, but any number of staff members can be present.

The two-tier process structure is a simplified case of the three-tier team structure. The top two tiers are compressed into one tier, but the implementation tier stays the same. With fewer teams in which to put people, bigger team sizes are expected using the same amount of people as in the three-tier structure. The communication overhead is bigger, but the work might get quicker depending on the team size. The effort breakdown is again used for the remainder of the algorithm.
Figure 5-9 Two-Tier Effort Breakdown

Effort Calculation

\[
EffortMultiplier_{RequirementsDesign} = CommunicationEffortMultiplier(TeamSize_{RequirementsDesign})
\]

\[
EffortMultiplier_{Implementation} = CommunicationEffortMultiplier(TeamSize_{Implementation})
\]

\[
EffortMultiplier_{IntegrationCustomerAcceptance} = CommunicationEffortMultiplier(TeamSize_{IntegrationCustomerAcceptance})
\]
\[ \text{EffortMultiplier}_{\text{RequirementsDesign\&Implementation}} = \text{CommunicationEffortMultiplier}\left(\text{TeamSize}_{\text{RequirementsDesign}} + \text{TeamSize}_{\text{Implementation}}\right) \]

\[ \text{EffortMultiplier}_{\text{RequirementsDesign\&IntegrationCustomerAcceptance}} = \text{CommunicationEffortMultiplier}\left(\text{TeamSize}_{\text{RequirementsDesign}} + \text{TeamSize}_{\text{IntegrationCustomerAcceptance}}\right) \]

\[ \text{EffortMultiplier}_{\text{Implementation\&IntegrationCustomerAcceptance}} = \text{CommunicationEffortMultiplier}\left(\text{TeamSize}_{\text{Implementation}} + \text{TeamSize}_{\text{IntegrationCustomerAcceptance}}\right) \]

\[ \text{Effort}_{\text{RequirementsDesign}} = PR_1 + PR_2 + PR_3 + PR_4 + PR_5 + PR_7 + PD_4 + PD_5 + PA_4 + IT_1 \]
\[ \text{Effort}_{\text{Implementation}} = PA_5 \]
\[ \text{Effort}_{\text{IntegrationCustomerAcceptance}} = IT_5 + IT_7 \]
\[ \text{Effort}_{\text{RequirementsDesign\&Implementation}} = PD_3 + PA_3 + IT_2 \]
\[ \text{Effort}_{\text{RequirementsDesign\&IntegrationCustomerAcceptance}} = PD_4 + PD_5 + PA_4 + PA_5 + IT_4 + PD_7 + PA_7 + PR_8 \]
\[ \text{Effort}_{\text{Implementation\&IntegrationCustomerAcceptance}} = IT_5 \]
\[ \text{Effort}_{\text{All}} = PR_6 + PD_6 + PA_6 + IT_6 + PD_8 + PA_8 + IT_8 \]

\[ \text{TierTwoEffortMultiplier} = \text{EffortMultiplier}_{\text{RequirementsDesign}} \times \text{Effort}_{\text{RequirementsDesign}} + \]
\[ \text{EffortMultiplier}_{\text{Implementation}} \times \text{Effort}_{\text{Implementation}} + \]
\[ \text{EffortMultiplier}_{\text{IntegrationCustomerAcceptance}} \times \text{Effort}_{\text{IntegrationCustomerAcceptance}} + \]
\[ \text{EffortMultiplier}_{\text{RequirementsDesign\&Implementation}} \times \text{Effort}_{\text{RequirementsDesign\&Implementation}} + \]
\[ \text{EffortMultiplier}_{\text{RequirementsDesign\&IntegrationCustomerAcceptance}} \times \text{Effort}_{\text{RequirementsDesign\&IntegrationCustomerAcceptance}} + \]
\[ \text{EffortMultiplier}_{\text{Implementation\&IntegrationCustomerAcceptance}} \times \text{Effort}_{\text{Implementation\&IntegrationCustomerAcceptance}} + \]
\[ \text{EffortMultiplier}_{\text{All}} \times \text{Effort}_{\text{All}} \]
Finally, TierTwoEffortEstimate = \textit{TierTwoEffortMultiplier} \times \text{COCOMOII Effort Estimate}

**Schedule Calculation**

To calculate project duration, the formula of effort divided by people is used. The TierTwoEffortEstimate from the previous section is used to represent the effort, and the number of people in a particular group is used for the people. Development effort that is not directly related to a particular team group is added as overhead. The equations that setup the schedule calculation are shown below:

\[
\begin{align*}
\text{Overhead}_{\text{Requirements}} &= \text{TierTwoEffortEstimate} \times (1 + PR_6) \\
\text{Overhead}_{\text{Design}} &= \text{TierTwoEffortEstimate} \times (1 + PD_6 + PD_8) \\
\text{Overhead}_{\text{Programming}} &= \text{TierTwoEffortEstimate} \times (1 + PA_6 + PA_8) \\
\text{Overhead}_{\text{Testing}} &= \text{TierTwoEffortEstimate} \times (1 + IT_6 + IT_8)
\end{align*}
\]

**Time for Plans and Requirement Phase:**

\[
\begin{align*}
\text{Time}_{\text{PR}_1} &= \frac{PR_1 \times \text{Overhead}_{\text{Requirements}}}{\text{TeamSize}_{\text{RequirementsDesign}}} \\
\text{Time}_{\text{PR}_2} &= \frac{PR_2 \times \text{Overhead}_{\text{Requirements}}}{\text{TeamSize}_{\text{RequirementsDesign}}} \\
\text{Time}_{\text{PR}_3} &= \frac{PR_3 \times \text{Overhead}_{\text{Requirements}}}{\text{TeamSize}_{\text{RequirementsDesign}}} \\
\text{Time}_{\text{PR}_4} &= \frac{PR_4 \times \text{Overhead}_{\text{Requirements}}}{\text{TeamSize}_{\text{RequirementsDesign}}} \\
\text{Time}_{\text{PR}_5} &= \frac{PR_5 \times \text{Overhead}_{\text{Requirements}}}{\text{TeamSize}_{\text{RequirementsDesign}}} \\
\text{Time}_{\text{PR}_6} &= \frac{PR_6 \times \text{Overhead}_{\text{Requirements}}}{\text{TeamSize}_{\text{RequirementsDesign}} \times \text{TeamSize}_{\text{IntegrationCustomerAcceptance}}}
\end{align*}
\]
Time for Product Design Phase:

\[
\begin{align*}
\text{Time}_{\text{PD}_1} &= \frac{PD_1 \times \text{Overhead}_{\text{Design}}}{\text{TeamSize}_{\text{RequirementsDesign}}} \\
\text{Time}_{\text{PD}_2} &= \frac{PD_2 \times \text{Overhead}_{\text{Design}}}{\text{TeamSize}_{\text{RequirementsDesign}}} \\
\text{Time}_{\text{PD}_3} &= \frac{PD_3 \times \text{Overhead}_{\text{Design}}}{\text{TeamSize}_{\text{RequirementsDesign}} \times \text{TeamSize}_{\text{Implementation}}} \\
\text{Time}_{\text{PD}_4} &= \frac{PD_4 \times \text{Overhead}_{\text{Design}}}{\text{TeamSize}_{\text{RequirementsDesign}} \times \text{TeamSize}_{\text{Implementation}} \times \text{TeamSize}_{\text{IntegrationCustomerAcceptance}}} \\
\text{Time}_{\text{PD}_5} &= \frac{PD_5 \times \text{Overhead}_{\text{Design}}}{\text{TeamSize}_{\text{RequirementsDesign}} \times \text{TeamSize}_{\text{Implementation}} \times \text{TeamSize}_{\text{IntegrationCustomerAcceptance}}} \\
\text{Time}_{\text{PD}_6} &= \frac{PD_6 \times \text{Overhead}_{\text{Design}}}{\text{TeamSize}_{\text{RequirementsDesign}} \times \text{TeamSize}_{\text{IntegrationCustomerAcceptance}}} \\
\end{align*}
\]

Time for Programming Activity Phase:

\[
\begin{align*}
\text{Time}_{\text{PA}_1} &= \frac{PA_1 \times \text{Overhead}_{\text{Programming}}}{\text{TeamSize}_{\text{RequirementsDesign}}} \\
\text{Time}_{\text{PA}_2} &= \frac{PA_2 \times \text{Overhead}_{\text{Programming}}}{\text{TeamSize}_{\text{RequirementsDesign}} \times \text{TeamSize}_{\text{Implementation}}} \\
\text{Time}_{\text{PA}_3} &= \frac{PA_3 \times \text{Overhead}_{\text{Programming}}}{\text{TeamSize}_{\text{Implementation}}} \\
\text{Time}_{\text{PA}_4} &= \frac{PA_4 \times \text{Overhead}_{\text{Programming}}}{\text{TeamSize}_{\text{RequirementsDesign}} \times \text{TeamSize}_{\text{Implementation}} \times \text{TeamSize}_{\text{IntegrationCustomerAcceptance}}} \\
\text{Time}_{\text{PA}_5} &= \frac{PA_5 \times \text{Overhead}_{\text{Programming}}}{\text{TeamSize}_{\text{RequirementsDesign}} \times \text{TeamSize}_{\text{IntegrationCustomerAcceptance}}} \\
\end{align*}
\]
Time for Integration and Test Phase:

\[
\text{Time}_{IT_1} = \frac{IT_1 \times \text{Overhead}_{testing}}{\text{TeamSize}_{requirements/design}}
\]

\[
\text{Time}_{IT_2} = \frac{IT_1 \times \text{Overhead}_{testing} + \text{TeamSize}_{implementation}}{\text{TeamSize}_{requirements/design}}
\]

\[
\text{Time}_{IT_3} = \frac{IT_1 \times \text{Overhead}_{testing} + \text{TeamSize}_{integration/customer acceptance}}{\text{TeamSize}_{implementation}}
\]

\[
\text{Time}_{IT_4} = \frac{IT_1 \times \text{Overhead}_{testing} + \text{TeamSize}_{integration/customer acceptance}}{\text{TeamSize}_{requirements/design}}
\]

\[
\text{Time}_{IT_5} = \frac{IT_1 \times \text{Overhead}_{testing} + \text{TeamSize}_{integration/customer acceptance}}{\text{TeamSize}_{integration/customer acceptance}}
\]

To calculate the schedule, the tasks that are on the critical path are added together.

The schedule equation for Two-Tier is equivalent to the Three-Tier schedule calculation.

\[
\text{TierTwoSchedule} = \text{Time}_{PR_1} + \text{Time}_{PD_1} + \text{Time}_{PD_2} + \text{Time}_{PD_3} + \text{Time}_{PD_4} + \text{Time}_{PD_5} + \text{Time}_{PR_2} + \text{Time}_{IT_3} + \text{Time}_{IT_4} + \text{Time}_{IT_5}
\]

5.13 One-Tier Structure

By adding the impact of the team size on the total effort, the one-tier calculation for effort follows:

\[
\text{TierOneEffortEstimate} = \text{CommunicationEffortMultiplier}(\text{TeamSize}_{OneTeam}) \times \text{COCOMOII Effort Estimate}
\]
The schedule is:

\[
\text{TierOneSchedule} = \frac{\text{TierOneEffortEstimate}}{\text{CommunicationEffortMultiplier(TeamSize}_{\text{OneTeam}})}
\]

5.14 Staff Loading

A new variable called staff loading is created by this cost estimation model. This variable represents the percentage of time that groups in the two-tier and three-tier are assigned to a task. In the one-tier structure, people can be thought to be always working on a task, so the staff loading is 100%. Each staff member in a one-tier process structure is always working on the critical path. If a staff member in a one-tier project is sick for a day, an extra day can be added to the end of the schedule if that time is not made up in another way.

With the two-tier and three-tier structure, work is not always conducted on the critical path, so the staff loading represents how much work effort is being planned for the critical path.

5.15 Optimization

The software cost estimation provides an optimization routine for each structure. Based on an objective function, the model runs different team size numbers in order to minimize the function. The default function is listed below:

\[
\text{OptimizationFunction} = \text{Minimize(Schedule)}
\]

The optimization function will try to staff the project in a way to minimize the amount of time the project takes to complete. At some point adding additional staff will
result in more overhead than the additional staff will provide in productivity. Right before this point is the optimal staffing point. In addition to finding the optimal staffing point, the optimization engine can also have to additional constraint. The first constraint specifies a minimum total staff. The optimization engine will find the optimal staffing point with a total staff that includes at least the minimum total staff. The second constraint is a maximum total staff. This works by setting a maximum total staff size that the optimization engine must honor. Both constraints can be used simultaneously to limit the solution space between a maximum and minimum number of total staff.

The algorithm is implemented in two different methods. The first method is a brute force optimization. This is used for both the one-tier and two-tier process structures. All possible combination of staff can be checked in under a second with a brute force approach. But optimizing a three-tier process structure is inefficient with a brute force approach. In some cases, an optimal result is expected to take many years so solve. So an external nonlinear solver is used to provide the optimization. Lingo 8.0 by LINDO Systems Inc. is used to solve exactly the same problem that was being attempted with the brute force attempt, but instead in a much more efficient way. The Pre-solver in Lingo can reduce the optimization problem so just solve in a few seconds. The Lingo script is available in Appendix C.
5.16 Conclusion

This chapter describes the building of the new software cost estimation model. The improvements over COCOMO II were shown in staff allocation optimization. All the equations needed for the algorithm to create the new estimates have been shown in the chapter along with a sample test case to show the algorithm in use. Improvements to cost estimation are possible with use of the new software cost estimation model. The following chapter will provide empirical support to validate the model described in this chapter.
CHAPTER 6 DECISION SUPPORT TOOL

Build a system that even a fool can use, and only a fool will want to use it.
— George Bernard Shaw

6.1 Example Test Run

This chapter describes how the PSEstimate tool works. This chapter estimates a sample project through the tool. Screen shots are provided to illustrate the tool at different parts through the estimation process.

COCOMO II Estimate

With the new software cost estimation model described, a sample project will show the models in use. A test case with software size being 40 KSLOC is used with the default COCOMO II effort multipliers and scale factors. COCOMO II estimates total effort to be 169.9 people-months. However, COCOMO II by default does not include the requirements phase of development. The effort and schedule required to build the requirements have to be added to COCOMO II estimate. In this case, the new effort from COCOMO II including requirements is 181.8 people-months. COCOMO II estimates that 9.5 people are required and the project will take 19.2 months.

One-Tier Estimate

The estimate for the one-tier estimate is 192 people-months. The difference between the COCOMO II estimate and the one-tier estimate is due to communication
overhead. COCOMO II’s 9.5 staff estimate is rounded to 10 people to result in a schedule of 19.2 months. However, if 13 people are used instead of 10, the effort increase to 199.5 people-months, but the schedule is reduced to 15.3 months. COCOMO II has limited support for changing the schedule.

Two-Tier Structure

The estimate for the two-tier structure is 185 people-months. Ten people are used as in the one-tier structure. However, the calculated schedule is 31 months. With three people placed in the analysis and design group, three in system testing, and four in implication, the model shows that software development will take much longer than COCOMO II estimates. If instead six people are placed in analysis and design, six in system testing, and nine in implementation, the total effort only increase to 198 people-months, but the schedule is reduced to 16 months.

Three-Tier Structure

The estimate for the three-tier structure is 184 people-months. Ten people are used to get this estimate. Two people are in requirements, two in design, two in implementation, two in integration testing, and two in acceptance testing. The calculated schedule is 52 months. But, if three people are in requirements, five in design, nine in implementation, six in integration testing, and one in acceptance testing, the total effort increases to 199 people-months, but the schedule is reduced to 16 months.
Conclusion

In all three cases, using the model described in this chapter, assigning different team size than COCOMO II suggested improves the schedule estimate for development. For the same data, the best process structure is to use a one-tier process structure with 13 people. There were no bad structures for the sample test case if the number of staff were assigned to each group optimally. Without good staff allocation, the three-tier structure will deliver a software project much later than is estimated by COCOMO II.

6.2 Tool Discussion

This next section shows the developed tool in use. Four different screen shots are used to show the developed new cost estimation project tool.

The first screenshot details the choices in project characteristics available. Both the lines of code or function point methodology is available for software sizing. In the following screenshot, the function point methodology is used with backfiring to come up with equivalent lines of code estimate. Using an estimate of 900 unadjusted function points of C++ yields 47700 lines of code. The five scale factors are also selectable. Finally, the option of using the early design or post-architecture effort multipliers is available.
The next screenshot shows the results of the estimation based on the project characteristics. COCOMO II estimates for effort, schedule, and staffing are estimated along with the derived effort multipliers, scale factors, and equivalent lines of code. The three different process structures are estimated based on a default-staffing algorithm.
Figure 6-2: Screenshot of Developed Tool - Simulation Results

The next screenshot shows the results of optimizing each process structure. There is a large improvement in schedule after optimization.
6.3 Tool Construction

PSEstimate was developed in C# using Visual Studio .NET 2002, Visual Studio .NET 2003, and finally Visual Studio 2005. The final version is compiled in Visual Studio 2005. As the technology changed the software was updated as needed. The program runs with the Microsoft’s .NET framework 2.0. The software uses Microsoft’s ClickOnce deployment method to be placed on the web. The ClickOnce web deployment forces users to navigate to the web server where PSEstimate was located. Any updates were automatically retrieved. This allowed used to be guaranteed to have the latest
version of the software. Approximately 10,000 lines of code were written in C# to implement the tool. The tool took the author approximately two years of full-time work to design, implement and test the tool. The tool uses external code, Lingo 8 API, for the tier-three non-linear solver. This code is called via Dynamic Link Library calls. Since the Lingo 8 API is written for Windows machines, the software only currently runs on Windows based machines.
CHAPTER 7 EXPERIMENTAL VALIDATION

7.1 Introduction

This chapter details the experimental validation used to assess the software cost estimation artifact and project management tool developed in this dissertation. Justifying and evaluating an artifact is an important step in the design science paradigm.

![Design Science Research Model](Image)

Figure 7-1 Design Science Research Model

(Hevner, March et al. 2004)
7.2 Study Rationale

According to McGrath, there are eight different research strategies available when designing a study. These strategies include laboratory experiments, experimental simulations, field experiments, field studies, computer simulations, formal theory, sample surveys and judgment tasks. Any particular type of study will have strengths and weaknesses when looking at three objectives: generalizability with respect to populations, precision in control and measurement of variables related to the behaviors of interest, and existential realism, for the participants, of the context within which those behaviors are observed (McGrath 1982). While each objective is important, it is impossible to maximize all three objectives simultaneously with one study. This problem is commonly known as McGrath’s three-horn dilemma.

A laboratory experiment can maximize control at the expense of both generalizability and reality. An experiment is the best study available to capture cause and effect. By using a control group and an experimental group, differences between the two groups can be attributed to the treatment, i.e. being in the control group or experimental group. A field experiment can maximize reality at the expense of control and generalizability. In a field experiment, a study is conducted in an organization. But, it is very hard to create controlled conditions and the results are represent a particular organization. A sample survey can maximize generalizability at the expense of both control and reality. A survey can be sent to a random sample of people, but control and reality are poor.
An experiment should strive to provide the most control as possible in order to show cause and effect. A key strength of the controlled experiment is that, since other possible effects are controlled, all variation in the dependent variable is attributed to the treatments.

Participants are randomly assigned into one of three treatment groups. This fact makes the study a true randomized controlled experiment. Random assignment is important from the experimental and data analysis standpoint. In this study a participant is not guided or placed into any treatment group based on any factor other than a random assignment therefore all participants have an equal chance of being assigned to any treatment group. Two coin tosses were used to randomize participants into groups. Participants were placed in a treatment group based they outcome as shown in Table 7-1.

<table>
<thead>
<tr>
<th>First Coin Toss</th>
<th>Second Coin Toss</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heads</td>
<td>Head</td>
<td>Manual</td>
</tr>
<tr>
<td>Heads</td>
<td>Tails</td>
<td>COCOMO II</td>
</tr>
<tr>
<td>Tails</td>
<td>Heads</td>
<td>PSEstimate</td>
</tr>
<tr>
<td>Tails</td>
<td>Tails</td>
<td>Repeat</td>
</tr>
</tbody>
</table>

Table 7-1 Randomizing to Treatments

Notice, if two tails are flipped, the whole procedure would start over to insure a participant had an equal chance of being in any one of the three groups.

The three possible treatment groups are no tools support or a manual group, COCOMO II, and PSEstimate. The no tool support group is the control group. This group will not be given any cost estimation models to help estimate software. The second
group, COCOMO II, is given a computer tool that supports estimating with COCOMO II. The third group, PSEstimate, is given the tool developed in this dissertation to help estimate software costs.

7.3 Institutional Review

The University of South Florida Institutional Review board is required to approve all studies that involve human subjects. The review board requires that all investigators have proper training in conducting studies with human subjects. This study has been approved by the institutional review board as IRB # 101906. The approval for the study is listed in Appendix D.

For this experiment, all participants were briefed on the study in general, able to read the consent form, answer any questions they had about the consent documents or study, and then signed that they acknowledged and gave consent to participant in the study. All participants were given a signed copy of the consent document for their own records.

7.4 Research Question

The research questions are repeated from Chapter 1:

Research Question 1: Can a software cost estimation model be built that models the effect of both inter-group coordination and intra-group communication?

Research Question 2: Can a software cost estimation tool be built for project managers that implements inter-group coordination, intra-group communication and process structure?
Research Question 3: Does an experiment demonstrate the effectiveness of the new software cost estimation model?

The empirical study focuses on the third research question. Will the experiment demonstrate the effectiveness of the software cost estimation model that was built in the previous chapters?

7.5 Hypotheses

Based on the third research question and the research model, five hypotheses were developed.
H1: Use of a software cost estimation tool for software development projects increases the accuracy of effort and schedule.

H2: Use of a software cost estimation tool for software development projects reduces the variation (increases the consistency) of estimates for effort and schedule.

H3: Users of a software cost estimation tool for software development projects are more likely to have an appropriate level of confidence in their estimates than estimators without support.

H4: Use of a software cost estimation tool for software development projects increases the satisfaction with the estimation technique.

H5: Use of a software cost estimation tool for software development projects increases the users’ perceived usefulness of the estimation technique.

7.6 Pretest

Several pretests were conducted during the development of the cost estimation tool. One pretest used twelve master students in MIS to estimate the staff and effort while using the PSEstimate tool. It was found that more work was needed in order for the tool to be used in an experimental setting. The participants were timed during this initial pretest and many participants needed more than 90 minutes to estimate two tasks. The main problem was when participants were trying to estimate staffing with Tier Three. Using a brute force algorithm to find an optimal staffing needed many calculations. A single staffing optimization scenario would not finish in less than one hour. The participants would either have to sit and wait or otherwise cancel the task. To fix this
problem a nonlinear solver was used that reduced the optimization problem to fewer than two seconds. This made the tool much more useful to estimators.

The second pretest was conducted on one MIS doctoral students to get an idea of the time needed to conduct the experiment with all the changed made from the first pretest. The pretest was successful; feedback was obtained about the software and the experimental materials. Slight changes in instructions were made to clarify what was expected in the experiment.

7.7 Pilot Test

After two pretests and many changes to the experimental task and materials, a pilot test was conducted. In the pilot all three experimental treatments were conducted. A total of four people went through the experiment. One person was in the manual group, one person in the COCOMO II group, and two people were in the PSEstimate group. The pilot data was not sufficient to do any kind of analysis, but one task was changed because several participants rated one task being impossible to estimate with the given information. This was an error and was corrected before the main data collection started.

7.8 Main Study

After several pretests and a pilot test, the main data collection was ready to occur. The first step was to find participants. Participants in the study were selected based on their prior knowledge about software cost estimation. Because of this, participants were recruited from local companies. Employees that were either project managers or team leaders were targeted. People currently working on a project management certification
were also deemed to be sufficient since work experience as a project manager is a
perquisite for the PMP certification. A graduate course in project management was a
targeted since estimation knowledge would be sufficient in this type of a course. Finally,
various faculty members with work experience as project managers were also targeted. In
the end, 34 participants completed the experiment.

The average participant was 34 years old. The oldest participant was 54 and the
youngest was 21. Twenty-four of the participants were male, ten were female. On
average participants had 15 years of full-time work experience with 12 years being IT
related. Five years of time in current position was the average for the participants.

7.9 Training

All participants were given a 45 minute presentation on software cost estimation
before participating in the experiment. During the briefing, experimental materials were
explained to the participants. After the briefing any remaining questions were answered
and then the participants were allowed to work on the experimental tasks.

7.10 Experimental Tasks

In this experiment, to obtain maximum control, all participants were given the
same experimental materials except one sheet of paper notifying the participants a
website to go to in order to download the software for the experiment. The COCOMO II
and PSEstimate groups were each given a different website to go to. The manual group
was given no additional information. The COCOMO II t and PSEstimate treatment page
can be seen in the experimental materials in Appendix E.
After initial training, all participants read an instruction sheet that thanked them for their participation and outlined the tasks. The welcome sheet can be seen in Appendix E.

The next page in the experimental materials was the Institutional Review Board Consent form. These three pages can be seen in Appendix E. The participants were allowed to keep a copy of the consent form. Included on this was the phone number of the investigators in case they had any additional questions or concerns.

After the participants filled out the consent form, the next step was to complete a pre-experiment questionnaire. This form can be seen in Appendix E. In this questionnaire, demographic information such as age, employment history, and previous estimating background was collected.

7.11 Experimental Task 1

After all the demographic information was collected, the participants were to start estimating the first task. The participants were asked to write the start time when starting the first Task. Task 1 was broken into three parts, Task 1a, Task 1b, and Task 1c. In Task 1a, a 30K project was to be estimated. The participants were told they had 12 people to work on the task, and everyone worked in a single group. The participants were given historical data from other similar projects, but there was not a project that was exactly related to this project. Participants were also given some qualitative information about the team being very experienced and working well together. They were also told that the project was not complex and the development platform was commonly used in the organization.
With this information, the participants had to estimate the effort in man-hours required to complete the development. They had to rate their confidence in the estimate for man-hours. In addition they had to give a best and worst case value for effort. Finally, they had to provide a rationale for their estimate. The participants also had to do the same for schedule.

Task 1b was the same task as Task 1a except for the staffing available for the project increased from 12 people to 24 people. The participants were asked to estimate the same set of questions for Effort and Schedule as they did in Task 1a.

In Task 1c, everything was the same as Task 1a except for the project was now bigger. The size went from 30K to 180K. Also the staffing was reduced to 11 instead of 12. The participants were asked to estimate effort and schedule. This task worked as a manipulation check because there was a very similar project in the historical data. In the historical data there was a 183K project versus the 180K project proposed. The participants should use this information to help them with estimation. After Task 1c was estimated, the participants were asked to write the stop time for this task in the materials. From this a total time on task measure can be calculated for Task 1.

7.12 Experimental Task 2

When starting Task 2, the participants were asked to record the time they starting working on the task. At the end of the Task 2, the participants were to record the stop time so the time on task for Task 2 can be calculated.

Experimental Task 2 was again broken into 3 subtasks, Task 2a, Task 2b, and Task 2c. All three subtasks had the exact setup except for the staffing arrangement. The
project was a larger project than the projects in Task 1a and 1b. The task was to make estimates about developing a financial system, which should require an e-commerce application that was 80K in size. The task was setup to be a financial system so it would require more effort than a normal project. In Task 2a, the staffing was set to be 30 people working in one project group. Also the historical data was explained to be not relevant since this project more than double the staff of any historical projects. The participants had to estimate the effort and schedule required to complete this estimation just like in the previous task.

In Task 2b, everything was the same from Task 2a except for the staffing structure changed. Instead of having one project group, three project groups were used. There were a requirement and design team, an implementation team, and a testing team. Instead of having 30 people in one group, the 30 people were broken into one of three groups. The first team was the requirements and design team. This team consists of 9 people. The second team was the implementation team and consists of 13 people. The third team is the testing team and consists of 8 people. Again, the participants were asked to estimate effort and schedule.

In Task 2c, another staffing structure change was made from Task 2a. In this case five different project groups were used to build the software system. The first team was the requirements team. The requirements team consists of 4 people. The second team was the design team. Design team consists of 6 people. The third team was the implementation team and consists of 12 people. The fourth team is the testing team and consists of 7 people. The fifth team is the customer acceptance team. One person will
perform all the customer acceptance activities. Notice, there is still a total of 30 people working on the system development. The participants were asked to estimate the effort and schedule required to conduct this project.

<table>
<thead>
<tr>
<th>Experimental Task</th>
<th>Project Size</th>
<th>Staffing</th>
<th>Historical Data Available?</th>
<th>Project Characteristics</th>
<th>Best Estimation Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>30K</td>
<td>12</td>
<td>Yes, but not exactly similar</td>
<td>Team very experiences, works well together; project not too complex and development platform common</td>
<td>Historical Data</td>
</tr>
<tr>
<td>1b</td>
<td>30K</td>
<td>24</td>
<td>Yes, but not exactly similar</td>
<td>same</td>
<td>PSEstimate</td>
</tr>
<tr>
<td>1c</td>
<td>180K</td>
<td>11</td>
<td>Yes, very similar</td>
<td>same</td>
<td>Historical Data</td>
</tr>
<tr>
<td>2a</td>
<td>80K</td>
<td>30 Tier1</td>
<td>Yes, but not relevant</td>
<td>Complex e-commerce application</td>
<td>PSEstimate</td>
</tr>
<tr>
<td>2b</td>
<td>80K</td>
<td>30 Tier2</td>
<td>Yes, but not relevant</td>
<td>same</td>
<td>PSEstimate</td>
</tr>
<tr>
<td>2c</td>
<td>80K</td>
<td>30 Tier3</td>
<td>Yes, but not relevant</td>
<td>Same</td>
<td>PSEstimate</td>
</tr>
</tbody>
</table>

Table 7-3 Experimental Tasks Overview

7.13 Post Experiment Questionnaire

After all the tasks were estimated, a post-experiment questionnaire was completed. This can be seen in Appendix E. The questionnaire is used to measure three main constructs and includes the manipulation check. The first sets of questions were to measure the participants the perceived usefulness of the estimation technique. The next questions were to measure the participants’ satisfaction about the experiment. Finally, the last sets of questions were used as additional manipulation checks.

Manipulation checks were needed to be conducted in order to show that a particular participant in a treatment group received the treatment. Manipulation checks
add to the rigor of the method of experimentation. By asking certain questions after the experiment provides information to show that the manipulation was effective.

Four questions were used in the manipulation check.

In making my software cost estimates, the technique I mainly used:

a. A Calculator
b. Spreadsheet
c. Historical Data
d. Historical Data along with COCOMO II
e. Historical Data and PSEstimate
f. Other (please specify) ________________________________

During the study, circle all of the following techniques that you used to make software cost estimates:

a. A Calculator
b. Spreadsheet
c. Historical Data
d. COCOMO II
e. PSEstimate
f. Other (please specify) ________________________________

My preferred method to estimate software cost is to use _________________ to come up with my estimates.

When conducting Task 2, how do you think the difference in structures changed the communication that occurred as the same thirty people moved smaller group

From these questions an analysis can be conducted on if a participant was placed in the COCOMO II group but did not end up using COCOMO II for the estimation task.
CHAPTER 8 RESULTS AND DISCUSSION

8.1 Introduction

This chapter reports the results of the empirical study presented in Chapter 7. In this chapter the hypotheses that are presented in both Chapter 1 and Chapter 7 are tested with a discussion on the results of the findings. Five main hypotheses are being tested; see the research model for an overview.

Figure 8-1 Empirical Research Model

128
8.2 Treatment Breakdown

The random assignment of the 34 participants to three groups resulted in a desirable breakdown. Twelve people were assigned into the manual group, eleven were assigned to COCOMO II, and another eleven were assigned into the PSEstimate group.

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Number of Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td>12</td>
</tr>
<tr>
<td>COCOMO II</td>
<td>11</td>
</tr>
<tr>
<td>PSEstimate</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 8-1 Treatment Breakdown

8.3 Data Analysis Overview

With a sample size of 34 participants, a nonparametric data analysis will be the most conservative. Several data analysis techniques were studied to find the best possible analysis that would not violate assumptions. Since there were three groups, a Krusal-Wallis One-Way Analysis of Variance by Ranks was a possible choice. The Krusal-Wallis test is an extension of the Mann-Whitney U test. The Mann-Whitney U test is limited to two groups, whereas the Krusal-Wallis test expands the analysis to N groups. The Krusal-Wallis test has four assumptions (Abell, Braselton et al. 1999):

1. Samples are independent, random samples, one for each of K populations, where the median of population \( i \) is denoted by \( M_i \), \( i=1,\ldots,k \).

2. The sample values are at least ordinate, categorical data.

3. The populations all have the same shape. (If the populations differ, this difference is only in location).
4. The populations each have a continuous distribution.

   Assumption one, two, and four are satisfied through the experimental design. But assumption three cannot be assumed to be satisfied. Particularly, Hypothesis Two will specifically test against the populations having different shapes.

   The parametric analysis like ANOVA will have even more challenging demands towards assumptions. The standard parametric and non-parametric tests can not be used; therefore a different test was required. The best analysis was found in SAS under the procedure MULTTEST. This procedure can use bootstrapping to get population estimates rather than rely on assumptions. The downside of this procedure is it may take much time when bootstrapping with large datasets a large number of times. With a modern computer bootstrapping 20,000 times was a trivial task for this dataset.

8.4 Expert Validation

   One expert in software cost estimation rated the tasks. These values will be used as the “correct” answer for the analyses conducted in this chapter. The results of the expert ratings are shown below:

<table>
<thead>
<tr>
<th>Expert</th>
<th>Task 1a</th>
<th>Task 1b</th>
<th>Task 1c</th>
<th>Task 2a</th>
<th>Task 2b</th>
<th>Task 2c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert 1</td>
<td>E</td>
<td>S</td>
<td>E</td>
<td>E</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>13</td>
<td>40</td>
<td>12</td>
<td>1582</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>180</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>160</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>150</td>
<td>19</td>
</tr>
</tbody>
</table>

   Table 8-2 Experts Ratings of Effort and Schedule for Tasks

8.5 Accuracy

   The first hypothesis is about accuracy of the three treatment groups. Each treatment group was using a different type of tool to do estimation, the first was no
support, and the second treatment group was using COCOMO II, and the third treatment group was using PSEstimate.

Hypothesis H1 is as follows:

H1: Use of a software cost estimation tool for software development projects increases the accuracy of effort and schedule.

It is important to break the hypothesis into two parts, one for effort and one for schedule. This creates:

H1a: Use of a software cost estimation tool for software development projects increases the accuracy of effort.

H1b: Use of a software cost estimation tool for software development projects increases the accuracy of schedule.

The first step to testing this hypothesis is to see if there is a difference among the groups in estimates for effort and schedule.

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Task 1A Effort</th>
<th>Task 1B Effort</th>
<th>Task 1C Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td>Mean = 28</td>
<td>Mean = 57</td>
<td>Mean = 133</td>
</tr>
<tr>
<td></td>
<td>Std. Dev = 14</td>
<td>Std. Dev = 127</td>
<td>Std. Dev = 85</td>
</tr>
<tr>
<td></td>
<td>Range = 5-50</td>
<td>Range = 1.5-456</td>
<td>Range = 2-250</td>
</tr>
<tr>
<td>COCOMOII</td>
<td>Mean = 53</td>
<td>Mean = 106</td>
<td>Mean = 247</td>
</tr>
<tr>
<td></td>
<td>Std. Dev = 33</td>
<td>Std. Dev = 183</td>
<td>Std. Dev = 89</td>
</tr>
<tr>
<td></td>
<td>Range = 20.5-132.5</td>
<td>Range = 8.73-633.9</td>
<td>Range = 160-430</td>
</tr>
<tr>
<td>PSEstimate</td>
<td>Mean = 63</td>
<td>Mean = 71</td>
<td>Mean = 288</td>
</tr>
<tr>
<td></td>
<td>Std. Dev = 25</td>
<td>Std. Dev = 33</td>
<td>Std. Dev = 126</td>
</tr>
<tr>
<td></td>
<td>Range = 30-112</td>
<td>Range = 16-130</td>
<td>Range = 75-550</td>
</tr>
</tbody>
</table>

Table 8-3 Results for Task 1 for Effort

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Task 2A Effort</th>
<th>Task 2B Effort</th>
<th>Task 2C Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td>Mean = 69</td>
<td>Mean = 75</td>
<td>Mean = 140</td>
</tr>
<tr>
<td></td>
<td>Std. Dev = 60</td>
<td>Std. Dev = 68</td>
<td>Std. Dev = 161</td>
</tr>
<tr>
<td></td>
<td>Range = 3.8-198</td>
<td>Range = 3.8-210</td>
<td>Range = 3.8-500</td>
</tr>
<tr>
<td>COCOMOII</td>
<td>Mean = 375</td>
<td>Mean = 355</td>
<td>Mean = 394</td>
</tr>
<tr>
<td></td>
<td>Std. Dev = 197</td>
<td>Std. Dev = 225</td>
<td>Std. Dev = 285</td>
</tr>
<tr>
<td></td>
<td>Range = 76.1-610.3</td>
<td>Range = 79.9-720.3</td>
<td>Range = 80-986.6</td>
</tr>
<tr>
<td>PSEstimate</td>
<td>Mean = 384</td>
<td>Mean = 353</td>
<td>Mean = 347</td>
</tr>
<tr>
<td></td>
<td>Std. Dev = 293</td>
<td>Std. Dev = 172</td>
<td>Std. Dev = 171</td>
</tr>
<tr>
<td></td>
<td>Range = 137-712</td>
<td>Range = 109-600</td>
<td>Range = 93-581</td>
</tr>
</tbody>
</table>

Table 8-4 Results for Task 2 for Effort
The results of bootstrapping 20,000 times with a seed of 1054 are shown in Table 8-5, with significant differences (p < .10) shown in boldface. The results show that there are significant differences in the effort estimations between PSEstimate and the manual groups for Tasks 1a, 1c, 2a and 2b; and between COCOMO II and the manual groups for Tasks 2a and 2b. No significant differences were found between the two groups using computer-based tools, COCOMO II and PSEstimate, in effort estimations.

<table>
<thead>
<tr>
<th>Contrast</th>
<th>T1A E</th>
<th>T1B E</th>
<th>T1C E</th>
<th>T2A E</th>
<th>T2B E</th>
<th>T2C E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual vs. COCOMO II</td>
<td>.23</td>
<td>.99</td>
<td>.13</td>
<td>.0008</td>
<td>.005</td>
<td>.09</td>
</tr>
<tr>
<td>Manual vs. PSEstimate</td>
<td>.02</td>
<td>1.00</td>
<td>.01</td>
<td>.0007</td>
<td>.005</td>
<td>.26</td>
</tr>
<tr>
<td>COCOMO II vs. PSEstimate</td>
<td>.99</td>
<td>.99</td>
<td>.99</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 8-5 Bootstrap p-vals for Effort

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Task 1A Schedule</th>
<th>Task 1B Schedule</th>
<th>Task 1C Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td>Mean = 11</td>
<td>Mean = 61</td>
<td>Mean = 23</td>
</tr>
<tr>
<td></td>
<td>Std. Dev = 11</td>
<td>Std. Dev = 180</td>
<td>Std. Dev = 16</td>
</tr>
<tr>
<td></td>
<td>Range = 3-40</td>
<td>Range = 1.25-631.5</td>
<td>Range = 3-71</td>
</tr>
<tr>
<td>COCOMOII</td>
<td>Mean = 10</td>
<td>Mean = 7</td>
<td>Mean = 20</td>
</tr>
<tr>
<td></td>
<td>Std. Dev = 5</td>
<td>Std. Dev = 4</td>
<td>Std. Dev = 3</td>
</tr>
<tr>
<td></td>
<td>Range = 2.1-17.4</td>
<td>Range = 1.5-10.8</td>
<td>Range = 15.5-24.5</td>
</tr>
<tr>
<td>PSEstimate</td>
<td>Mean = 10</td>
<td>Mean = 7</td>
<td>Mean = 35</td>
</tr>
<tr>
<td></td>
<td>Std. Dev = 7</td>
<td>Std. Dev = 5</td>
<td>Std. Dev = 33</td>
</tr>
<tr>
<td></td>
<td>Range = 2-23</td>
<td>Range = 1.3-17</td>
<td>Range = 10-127</td>
</tr>
</tbody>
</table>

Table 8-6 Results for Task 1 for Schedule

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Task 2A Schedule</th>
<th>Task 2B Schedule</th>
<th>Task 2C Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td>Mean = 16</td>
<td>Mean = 14</td>
<td>Mean = 45</td>
</tr>
<tr>
<td></td>
<td>Std. Dev = 22</td>
<td>Std. Dev = 18</td>
<td>Std. Dev = 85</td>
</tr>
<tr>
<td></td>
<td>Range = 1.80</td>
<td>Range = 1-65</td>
<td>Range = 1-300</td>
</tr>
<tr>
<td>COCOMOII</td>
<td>Mean = 20</td>
<td>Mean = 21</td>
<td>Mean = 21</td>
</tr>
<tr>
<td></td>
<td>Std. Dev = 12</td>
<td>Std. Dev = 12</td>
<td>Std. Dev = 12</td>
</tr>
<tr>
<td></td>
<td>Range = 3-48</td>
<td>Range = 4-42.5</td>
<td>Range = 6-45</td>
</tr>
<tr>
<td>PSEstimate</td>
<td>Mean = 28</td>
<td>Mean = 33</td>
<td>Mean = 35</td>
</tr>
<tr>
<td></td>
<td>Std. Dev = 36</td>
<td>Std. Dev = 34</td>
<td>Std. Dev = 34</td>
</tr>
<tr>
<td></td>
<td>Range = 4.7-133</td>
<td>Range = 9-133</td>
<td>Range = 9-133</td>
</tr>
</tbody>
</table>

Table 8-7 Results for Task 2 for Schedule
The results of bootstrapping 20,000 times with a seed of 1054 are shown in Table 8-8. There are no significant differences between groups in schedule estimations.

<table>
<thead>
<tr>
<th>Contrast</th>
<th>T1A S</th>
<th>T1B S</th>
<th>T1C S</th>
<th>T2A S</th>
<th>T2B S</th>
<th>T2C S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual vs. COCOMO II</td>
<td>1.00</td>
<td>.92</td>
<td>1.00</td>
<td>.99</td>
<td>.97</td>
<td></td>
</tr>
<tr>
<td>Manual vs. PSEstimate</td>
<td>1.00</td>
<td>.92</td>
<td>.86</td>
<td>.96</td>
<td>.44</td>
<td>1.00</td>
</tr>
<tr>
<td>COCOMO II vs. PSEstimate</td>
<td>1.00</td>
<td>1.00</td>
<td>.66</td>
<td>.99</td>
<td>.94</td>
<td>.99</td>
</tr>
</tbody>
</table>

Table 8-8 Bootstrap p-vals for Schedule

An additional test was conducted to test to see if there were differences in treatment groups in effort and schedule. Welch’s ANOVA is a test conducted when the assumptions of a parametric ANOVA are violated, particularly when the assumption of equal variance is violated.

<table>
<thead>
<tr>
<th>Task</th>
<th>Welch’s ANOVA for Effort</th>
<th>Welch’s ANOVA for Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1a</td>
<td>.0014</td>
<td>.92</td>
</tr>
<tr>
<td>Task 1b</td>
<td>.7727</td>
<td>.60</td>
</tr>
<tr>
<td>Task 1c</td>
<td>.0034</td>
<td>.35</td>
</tr>
<tr>
<td>Task 2a</td>
<td>&lt;.0001</td>
<td>.66</td>
</tr>
<tr>
<td>Task 2b</td>
<td>&lt;.0001</td>
<td>.29</td>
</tr>
<tr>
<td>Task 2c</td>
<td>.0105</td>
<td>.32</td>
</tr>
</tbody>
</table>

Table 8-9 Welch's ANOVA for Effort and Schedule

The results of the Welch’s ANOVA test are consistent with the results using the bootstrapping technique to determine whether the groups differed in their estimations.

With both techniques there were significant differences between the manual and PSEstimate group for Task 1a, Task 1c, Task 2a, and Task 2b for effort; and between the manual group and COCOMO II for Task 2b for effort. There were no significant differences in schedule between any of the groups with either bootstrapping or Welch’s ANOVA.
When combining treatments COCOMO II and PSEstimate to form a new group, tool versus the manual group or no tool the following results occur:

<table>
<thead>
<tr>
<th>Contrast</th>
<th>T1A E</th>
<th>T1B E</th>
<th>T1C E</th>
<th>T2A E</th>
<th>T2B E</th>
<th>T2C E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool vs. No Tool</td>
<td>.025</td>
<td>.999</td>
<td>.01</td>
<td>&lt;.0001</td>
<td>.0004</td>
<td>.059</td>
</tr>
</tbody>
</table>

Table 8-10 Bootstrap p-vals for Effort

<table>
<thead>
<tr>
<th>Contrast</th>
<th>T1A S</th>
<th>T1B S</th>
<th>T1C S</th>
<th>T2A S</th>
<th>T2B S</th>
<th>T2C S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool vs. No Tool</td>
<td>1.00</td>
<td>.92</td>
<td>1.00</td>
<td>1.00</td>
<td>.98</td>
<td>.97</td>
</tr>
</tbody>
</table>

Table 8-11 Bootstrap p-vals for Schedule

The tool did not make a significant difference between Task 1b and Task 2a. There was a difference for Task 1a, but when the manual group was given double the people, the group effectively doubled the effort. Even though the manual group underestimated Task 1a, the gross correction for doubling the staff brought the average in line with the other groups.

The fact that there are no significant differences between groups in schedule estimation is rather interesting. All treatment groups approached the same correct answer for the amount of time it took to complete a project. Even though individuals might not have a correct answer, the averaging of estimates led to a good estimate.

With significant differences found for effort, an analysis is conducted to see which group is the most accurate. There are two methods in which accuracy can be judged. The first is to measure which group has a raw mean or bootstrap mean closest to expert’s estimates. Because of the amount of bootstrapping, both raw means and bootstrap mean are equal. The second method is to measure the differences each participant is from the expert on a percentage score. The mean of each group raw
percentage score or a bootstrap mean can be measure to find the closest on to zero. This would signify the most accurate. The table that follows summarizes the results of all four data analysis techniques.

<table>
<thead>
<tr>
<th>Task</th>
<th>Expert</th>
<th>Raw/Bootstrapped Mean</th>
<th>Raw/Bootstrap Percentage Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>M  C  P</td>
<td>M  C  P</td>
</tr>
<tr>
<td>1a</td>
<td>Effort</td>
<td>60  28  53  63</td>
<td>54%  12%  6%</td>
</tr>
<tr>
<td></td>
<td>Schedule</td>
<td>13  11  10  10</td>
<td>14%  22%  25%</td>
</tr>
<tr>
<td>1b</td>
<td>Effort</td>
<td>40  57  106  71</td>
<td>43% -166% -76%</td>
</tr>
<tr>
<td></td>
<td>Schedule</td>
<td>12  61  7  7</td>
<td>-411% 44% 45%</td>
</tr>
<tr>
<td>1c</td>
<td>Effort</td>
<td>1582 133 247 288</td>
<td>92% 84% 81%</td>
</tr>
<tr>
<td></td>
<td>Schedule</td>
<td>40  23  20  35</td>
<td>43% 49% 14%</td>
</tr>
<tr>
<td>2a</td>
<td>Effort</td>
<td>180 69 375 384</td>
<td>62% -109% -114%</td>
</tr>
<tr>
<td></td>
<td>Schedule</td>
<td>20 16 20 28</td>
<td>21% 2% -38%</td>
</tr>
<tr>
<td>2b</td>
<td>Effort</td>
<td>160 75 355 353</td>
<td>53% -122% -121%</td>
</tr>
<tr>
<td></td>
<td>Schedule</td>
<td>19 14 21 33</td>
<td>24% -12% -73%</td>
</tr>
<tr>
<td>2c</td>
<td>Effort</td>
<td>150 140 394 347</td>
<td>-7% -162% -132%</td>
</tr>
<tr>
<td></td>
<td>Schedule</td>
<td>19 45 21 35</td>
<td>-138% -10% -86%</td>
</tr>
</tbody>
</table>

Table 8-12 Accuracy Results vs. Expert

The results provide mixed support for H1 for effort estimations. As already discussed, there were significant differences between the experimental groups in these estimations for some of the tasks. But when using the expert’s rating as a measure of accuracy, the PSEstimate group was both significantly different and more accurate than the manual group for Task 1a. Both of the computer-tool groups were both significantly different from the manual group on the more difficult Task 2a and 2b, but for Task 2a the tool-using groups estimated effort higher than the expert, and for Task 2b the manual group was more accurate.
8.6 Consistency

Hypothesis 2 is about testing the variation that exists between estimators. A consistent estimate is more desirable rather than a non-consistent estimate assuming both have the same accuracy.

H2: Use of a software cost estimation tool for software development projects reduces the variation (increases the consistency) of estimates for effort and schedule.

H2a: Use of a software cost estimation tool for software development projects reduces the variation (increases the consistency) of estimates for effort.

H2b: Use of a software cost estimation tool for software development projects reduces the variation (increases the consistency) of estimates for schedule.

To test this hypothesis two tests are used. The first analysis is the Levene Test. The Levene Test is a statistical test for homoscedasticity, where as it will check for equal variance of a measure across groups. Levene Test is known for its robustness for violations against normal data. A significant test supports the idea that the dispersion is different among the three groups. The results of the Levene Test are shown in Table 8-13.

<table>
<thead>
<tr>
<th>Task</th>
<th>P-Value Levene Effort</th>
<th>P-Value Levene Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1a</td>
<td>.21</td>
<td>.28</td>
</tr>
<tr>
<td>Task 1b</td>
<td>.45</td>
<td>.34</td>
</tr>
<tr>
<td>Task 1c</td>
<td>.37</td>
<td>.29</td>
</tr>
<tr>
<td>Task 2a</td>
<td>.0054</td>
<td>.47</td>
</tr>
<tr>
<td>Task 2b</td>
<td>.0042</td>
<td>.42</td>
</tr>
<tr>
<td>Task 2c</td>
<td>.11</td>
<td>.32</td>
</tr>
</tbody>
</table>

Table 8-13 Levene Test for Effort and Schedule

In this experiment Task 2 was designed to test the consistency of estimates. As the structure of the project changed, it would be expected that the manual group and the COCOMO II group would have increasingly difficult problems with estimating. From the Levene Test, Task 2a and Task 2b have significant dispersion among the three treatment
groups. The Standard Deviation for Effort shows that the Manual group has much less variance than the COCOMO II or the PSEstimate group. Therefore, Hypothesis H2a is not supported. There were significant difference, but the opposite occurred than was hypothesized.

The Levene Test for schedule shows no significant dispersion. Therefore, hypothesis H2b is also not supported.

It is important to note that bootstrapping is a very conservative technique. In Task 1b, the manual group had a Standard Deviation of 180 versus 4 for COCOMO II and 5 for PSEstimate. This high standard deviation occurred because of 1 participant. The bootstrapping technique will reduce the impact of this influential data point to where it is not significant.

Another point that shows up in the data is that as the complexity of the structure of the project team changes across Task 2, both the Manual and COCOMO II group increase in variance. Note that the PSEstimate group decreases in variance as the project team increases.

8.7 Confidence

Hypothesis 3 states:

H3: User of a software cost estimation tool for software development projects are more likely to have an appropriate level of confidence in their estimates than estimators without support.

There can be four types of confidences, two are inappropriate and two are appropriate. Overconfidence occurs when a participant has a high confidence rating but is
not accurate. The next type is not confident inappropriately. In this case the participant is accurate but has low confidence. Overconfidence and not confident inappropriately are inappropriate confidence estimates.

The remaining two confidence levels are appropriate level of confidence. The first is confident appropriately. In this case the participant is accurate and has a high level of confidence. The last type is not confident appropriately. In this case the participant is inaccurate and has low confidence.

The analysis on testing H3 is unique. The first step is to rate for each task if the participant was accurate or not accurate. The measure of accuracy used the expert’s best and worst case as the acceptable range. If the participant’s estimate, best case or worst case fell within the acceptable range, the estimate was deemed to be accurate. Otherwise it was deemed inaccurate. Next the confidence was analyzed on each task for each participant. Since the expert rated all tasks with a 50% confidence, this was the limit. To be confident, a participant had to be above the expert’s 50% confidence level; otherwise they were rated not confident. The next step was to position the participant’s estimate into one of the four types of estimates. AP is appropriate confidence, OC is overconfident, NCA is not confident appropriately, and NCI is not confident inappropriately. After each task rated, a pivot table in Excel was created. The results follow:
<table>
<thead>
<tr>
<th>Count of Confidence</th>
<th>Type</th>
<th>AP</th>
<th>NCA</th>
<th>NCI</th>
<th>OC</th>
<th>Grand Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Treatment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manual</td>
<td></td>
<td>14</td>
<td>22</td>
<td>13</td>
<td>23</td>
<td>72</td>
</tr>
<tr>
<td>COCOMO II</td>
<td></td>
<td>19</td>
<td>13</td>
<td>13</td>
<td>21</td>
<td>66</td>
</tr>
<tr>
<td>PSEstimate</td>
<td></td>
<td>24</td>
<td>10</td>
<td>8</td>
<td>24</td>
<td>66</td>
</tr>
<tr>
<td>Grand Total</td>
<td></td>
<td>57</td>
<td>45</td>
<td>34</td>
<td>68</td>
<td>204</td>
</tr>
</tbody>
</table>

Table 8-14 Pivot Table of Confidence Type Results

From the table is clear that the PSEstimate group has more appropriate level of confidence than any other group. For Not Confident Appropriately the manual group had neither a good estimate nor high confidence. For Not Confident Inappropriately it is clear why the manual group is so high. Even though they had good historical data, this was not enough for many participants to create a high level of confidence in their estimates. For overconfidence, the groups were about even.

<table>
<thead>
<tr>
<th>Count of Confidence</th>
<th>Type</th>
<th>AP</th>
<th>NCA</th>
<th>NCI</th>
<th>OC</th>
<th>Grand Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Treatment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Tool</td>
<td></td>
<td>14</td>
<td>22</td>
<td>13</td>
<td>23</td>
<td>72</td>
</tr>
<tr>
<td>Tool</td>
<td></td>
<td>43</td>
<td>23</td>
<td>21</td>
<td>45</td>
<td>132</td>
</tr>
<tr>
<td>Grand Total</td>
<td></td>
<td>57</td>
<td>45</td>
<td>34</td>
<td>68</td>
<td>204</td>
</tr>
</tbody>
</table>

Table 8-15 Results of Tool vs. No Tool for Confidence

8.8 Satisfaction and Perceived Usefulness

Satisfaction and Perceived Usefulness make up Hypothesis 4 and 5.

H4: Use of a software cost estimation tool for software development projects increases the users’ satisfaction with the estimation technique.

H4a: The PSEstimate group will have higher satisfaction than COCOMO II group and the COCOMO II group will have higher satisfaction than the Manual group.
H4b: The COCOMO II and PSEstimate group together will have higher satisfaction than the Manual group.

H5: Use of a software cost estimation tool for software development projects increases the user’s perceived usefulness with the estimation technique.

H5a: The PSEstimate group will have higher perceived usefulness than COCOMO II group and the COCOMO II group will have higher perceived usefulness than the Manual group.

H5b: The COCOMO II and PSEstimate group together will have higher perceived usefulness with the estimation technique than the Manual group.

The constructs for satisfaction and perceived usefulness will be analyzed with identical analysis techniques. The first step in checking for differences among the three treatment group for satisfaction and perceived usefulness is to conduct two psychometric tests on the items that measure these two constructs.

An item-total along with Cronbach’s alpha is a standard technique to test reliability. The results of these tests are shown in Table 8-16 and Table 8-17:
<table>
<thead>
<tr>
<th>Item Wording</th>
<th>Scale</th>
<th>Item-Total</th>
<th>Cronbach’s Alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Dissatisfied---Very Satisfied</td>
<td>1-7</td>
<td>.81</td>
<td></td>
</tr>
<tr>
<td>Very displeased---Very Pleased</td>
<td>1-7</td>
<td>.92</td>
<td>.93</td>
</tr>
<tr>
<td>Very frustrated---Very contented</td>
<td>1-7</td>
<td>.85</td>
<td></td>
</tr>
<tr>
<td>Absolutely terrible—Absolutely delighted</td>
<td>1-7</td>
<td>.75</td>
<td></td>
</tr>
</tbody>
</table>

Table 8-16 Item-Total for Satisfaction

<table>
<thead>
<tr>
<th>Item Wording</th>
<th>Scale</th>
<th>Item-Total</th>
<th>Cronbach’s Alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the software estimation technique in this experiment improves my performance in conducting software cost estimation.</td>
<td>1-7</td>
<td>.94</td>
<td></td>
</tr>
<tr>
<td>Using the software estimation technique in this experiment improves my productivity in conducting software cost estimation.</td>
<td>1-7</td>
<td>.94</td>
<td>.97</td>
</tr>
<tr>
<td>Using the software estimation technique in this experiment improves my effectiveness in conducting software cost estimation.</td>
<td>1-7</td>
<td>.95</td>
<td></td>
</tr>
<tr>
<td>Overall, the software technique used in this experiment was useful in conducting software cost estimation.</td>
<td>1-7</td>
<td>.88</td>
<td></td>
</tr>
</tbody>
</table>

Table 8-17 Item-Total and Cronbach’s Alpha for Perceived Usefulness

The results show strong item-total correlation for the items with the constructs.

Cronbach’s alpha is excellent for both constructs.

Having very reliable measures, a new measure called TotalSat was created that is a summation of the four satisfaction items. Also another measure TotalUse was created that was also a summation of the four perceived usefulness items. These variables are used as dependent variables in the analysis.
A bootstrapping technique was used to test the two hypotheses with the newly created dependent variables TotalSat and TotalUse. The results follow:

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Satisfaction (TotalSat)</th>
<th>Perceived Usefulness (TotalUse)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td>Mean = 12.5</td>
<td>Std. Dev = 4.14</td>
</tr>
<tr>
<td></td>
<td>Mean = 12</td>
<td>Std. Dev = 6.6</td>
</tr>
<tr>
<td>COCOMOII</td>
<td>Mean = 17.7</td>
<td>Std. Dev = 4.1</td>
</tr>
<tr>
<td></td>
<td>Mean = 20</td>
<td>Std. Dev = 5.2</td>
</tr>
<tr>
<td>PSEstimate</td>
<td>Mean = 16.5</td>
<td>Std. Dev = 4.3</td>
</tr>
<tr>
<td></td>
<td>Mean = 16</td>
<td>Std. Dev = 5.0</td>
</tr>
</tbody>
</table>

Table 8-18 Satisfaction and Treatment Means

Based on Table 8-18 hypothesis H4a and H5a are not supported. COCOMO II had a higher satisfaction and perceived usefulness than both the PSEstimate and manual group. By combining the COCOMO II and PSEstimate group together, a tool versus no tool analysis can be conducted. This test can be used to test hypothesis H4b and H5b.

<table>
<thead>
<tr>
<th>Contrast</th>
<th>Satisfaction</th>
<th>Perceived Usefulness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool vs. No Tool</td>
<td>.014</td>
<td>.012</td>
</tr>
</tbody>
</table>

Table 8-19 Bootstrap p-vals for Satisfaction and Perceived Usefulness

From Table 8-19 there is support for Hypothesis H4b and H5b. The satisfaction and perceived usefulness of using an estimation tool was significantly better than not using an estimating tool or conducting the estimation manually.

The results tend to show two different underpinnings when analyzing satisfaction and perceived usefulness. The first thing is individuals do not like to do estimation manually. It is rather frustrating for some and many people do not think it is an effective use of their time. This can result in a low rating for satisfaction and perceived usefulness. This result was prevalent during the pilot test after debriefing participants. The effect
carried over to the main experiment. The second result is people really liked using COCOMO II to do estimating. Maybe because it is the state-of-the-practice tool, but whatever the reason, people report high satisfaction and perceived usefulness with COCOMO II. The PSEstimate group was inconclusive. It was not significantly different from the manual group or the COCOMO II group. The PSEstimate group was almost significantly different from the Manual group. The raw p-values were .03 and .07 when comparing the Manual group versus the PSEstimate group. The bootstrapping reduced the p-values to non-significant results. Some additional work needs to be conducted to see what is causing satisfaction and perceived usefulness to lag slightly below COCOMO II.

From the results is it clear that PSEstimate needs more development, most likely in the interface. PSEstimate is addressing a much more complex task in the explict modeling of team structure versus the COCOMO II version. A better way of inputting team information can affect the perceived usefulness scores.
CHAPTER 9 CONCLUSIONS AND CONTRIBUTIONS

9.1 Introduction

Software cost estimation remains an important unsolved challenge. Project managers need to have tools that help them successfully manage their projects. By better understanding software development, better software cost estimation models can be created that will help project managers meet their goals. By introducing the software handoff, a different approach to software cost estimation is undertaken. A new software cost estimation tool is created to help support decision making by project managers.

9.2 Contributions to Research

This dissertation contributes to research in many ways. First, a theoretical framework for software cost estimation is presented. By using the concept of communication overhead, a cost estimation model that includes communication is created. The theoretical framework provides a measure that is important for cost estimation but is not always measured.

The second contribution to research is the use of secondary data to perform validation of the theoretical framework presented in this dissertation. By showing rigor, the effect that the findings are only spurious correlations is minimized. In the research performed so far on software cost estimation, many researchers ignore the assumptions of
the particular statistical method used. By properly performing the analysis on the secondary data, a documented method of analysis is presented for others to understand.

The third contribution to research comes from the experimental validation of the software cost estimation model. Designing a software cost estimation validation experiment is not a method commonly performed in the field. A novel approach and methodology is presented in this dissertation. With this information, future software cost estimation experiments can be performed.

The fourth contribution to research comes in the form of the optimization formula presented. Software cost estimation has yet to model the trade-off between effort and schedule. This initial attempt at developing an optimization formula will give future researchers a starting point when trying to understand the tradeoff project managers make when respect to effort and schedule.

The fifth contribution is from the empirical study itself. An experiment was thought out and conducted that clearly provides useful results to the software cost estimation research community. Accuracy, Consistency, Satisfaction, Confidence, and Perceived Usefulness are presented and measured in an experimental setting.

From the experimental results, there was mixed support for the hypothesized relationships. An interesting finding was that even though effort was significantly different among the teams, the estimates for schedule were not. This finding will have to be further investigated in another study.

In general PSEstimate was positive for estimators. People that estimate software
believe there should be tools to help estimation. The group that did estimation manually thought the process was archaic and many stated there has to be a better way to estimate.

Through the design science paradigm, the artifact was created, which was the new software cost estimation model. The model was instantiated through PSEstimate and tested in the field. A finding the PSEstimate needs to be improved slightly in from the perceived usefulness ratings is another type of contribution.

9.3 Contributions to Practice

Currently the COCOMO II schedule reduction multiplier is the most common method of estimating the impacts of reducing the delivery date of software projects. Many times a project manager is in charge of a project that has a critical time-to-market delivery date. Based on the work presented in this dissertation, the COCOMO II Schedule Reduction Multiplier is ineffective at helping a project manager make changes to the project to deliver a project with the desired schedule reduction. By using COCOMO II on different sized projects to estimate effort, schedule and staff and then using the schedule reduction multiplier to recalculate effort, schedule, and staff, a clear pattern emerges. For any sized project, according to COCOMO II to reduce the schedule to 75% of the original, staffing needs to be increased by about 91%. Using the schedule-reduction-multiplier methodology ignores any effects of communication overhead.
Table 9-1 COCOMO II Schedule Reduction Multiplier

<table>
<thead>
<tr>
<th>Lines of Code</th>
<th>Original Estimate</th>
<th>75% of Original</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total Effort</td>
<td>Schedule Staff</td>
</tr>
<tr>
<td>2000</td>
<td>6.7</td>
<td>6.7</td>
</tr>
<tr>
<td>8000</td>
<td>31</td>
<td>10.9</td>
</tr>
<tr>
<td>16000</td>
<td>66.4</td>
<td>13.9</td>
</tr>
<tr>
<td>32000</td>
<td>142.2</td>
<td>17.8</td>
</tr>
<tr>
<td>64000</td>
<td>304.8</td>
<td>22.6</td>
</tr>
<tr>
<td>128000</td>
<td>653.2</td>
<td>28</td>
</tr>
<tr>
<td>512000</td>
<td>3000</td>
<td>46.8</td>
</tr>
</tbody>
</table>

This dissertation provides a replacement to the COCOMO II schedule reduction multiplier so needed by project managers. By including the effects of communication overhead, a better formulated estimate of what occurs when the schedule is reduced is explained.

Another practical contribution to project management from this research is in giving project managers the ability to experiment with different team sizes. With software cost estimation models such as COCOMO II, team size was not directly changeable. There is poor linkage between the staffing estimate given by the software cost estimation models and staff members assigned to a software project. Many software cost estimation tools give a staffing estimate, but no support occurs if the staffing needed is not available. Project managers now have a tool to help with different staffing situations.

The ability to understand that not every software development project needs the same type of structure adds to the practical contributions of this dissertation. By showing
three different process structures with estimates for effort, schedule, and staffing allows project managers to explore different structures to develop software.

Finally delivering a decision support tool that the project manager can easily run on a personal computer is a major contribution to practice. With project managers five years behind the state-of-the-art in tools and techniques for project management, getting relevant knowledge to project managers is a challenge. By developing an easy to use tool that helps support decision making, the knowledge gap can be addressed.

9.4 Limitations and Key Assumptions

There are two key assumption and limitations of this work. First, COCOMO II is used as the basis for effort calculations and any errors in COCOMO II are inherited in the estimates in this dissertation. Being an extension rather than a replacement to COCOMO II, criticisms and limitations of COCOMO II are also assumed in this dissertation. A limitation in COCOMO II occurs when estimating very small project sizes less than eight KLOC. The cost estimation model presented in this dissertation will not be able to provide reasonable estimates of very small projects.

The second limitation occurs with the empirical data on communication overhead with very large teams. Little empirical data explains the impact of communication overhead in teams over 30 people. At 30 people, the communication overhead is 54%. However, as the group increases to 50 people little is known. This dissertation assumes that for all teams above 30 people, the communication overhead remains at 54%. Based on the exponential shape of the communication overhead chart, communication overhead is expected to continually increase to a point where the communication overhead exceeds
100%, meaning adding an additional person will cause more effort to be expended in communication than work on the project. By placing more than 30 people in a team, the model will underestimate the effort needed.

9.5 Future Work

The work presented in this dissertation is a solid contribution to software cost estimation. Future work is possible based on this dissertation. First, only three different process structures are presented in this dissertation. In reality, having a customizable process structure allows the greatest flexibility to a project manager. By ensuring the process structure matches what is used in the project managers’ organization in addition to other potential process structures that might be used, will allow the software cost estimation to provide the best contribution to practice.

Second, experience is an important cost driver in COCOMO II. However, COCOMO II provides experience at a group level rather than an individual level. There is a cost driver for analyst experience and programmer experience in COCOMO II, but the impacts of experience are not isolated. Consider the experience level of programmer is medium. An additional expert replaces a medium programmer thereby increasing the experience level of the group. With COCOMO II, the experience level cost driver for the programmers lowers the effort multiplier, which lowers the effort estimate. With the lower effort estimate, less staff will be needed, depending on which people are not needed, the experience level changes again, causing a different staffing level needed. This circular process never stops, therefore only rough estimates of experience are modeled. By modeling each individual experience will allow this cost estimation model
to better explain the effect of having people with different experiences. In addition, experience can become a factor in different process structures.

Third, taking the lessons learned in this dissertation and creating an optimization tool, where a project manager can input the team information and get an “optimal” team structure based on the team is possible. A decision support tool can have many more structures available for the manager.


The Standish Group (2003). Press release: Latest Standish Group CHAOS Report shows project Success Rates have improved by 50%. West Yarmouth, MA.


APPENDICES
## APPENDIX A: KEMERER DATASET

<table>
<thead>
<tr>
<th>Project Number</th>
<th>Software</th>
<th>Hardware</th>
<th>Months</th>
<th>Effort</th>
<th>KSLOC</th>
<th>SLOC/MM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cobol</td>
<td>IBM 308X</td>
<td>17</td>
<td>287</td>
<td>253</td>
<td>884</td>
</tr>
<tr>
<td>2</td>
<td>Cobol</td>
<td>IBM 43XX</td>
<td>7</td>
<td>82.5</td>
<td>40.5</td>
<td>491</td>
</tr>
<tr>
<td>3</td>
<td>Cobol</td>
<td>DEC VAX</td>
<td>15</td>
<td>1107.31</td>
<td>450</td>
<td>406</td>
</tr>
<tr>
<td>4</td>
<td>Cobol</td>
<td>IBM 308X</td>
<td>18</td>
<td>86.9</td>
<td>214.4</td>
<td>2467</td>
</tr>
<tr>
<td>5</td>
<td>Cobol</td>
<td>IBM 43XX</td>
<td>13</td>
<td>336.3</td>
<td>449.9</td>
<td>1338</td>
</tr>
<tr>
<td>6</td>
<td>Cobol</td>
<td>DEC 20</td>
<td>5</td>
<td>84</td>
<td>50</td>
<td>595</td>
</tr>
<tr>
<td>7</td>
<td>Cobol</td>
<td>DEC 20</td>
<td>5</td>
<td>23.2</td>
<td>43</td>
<td>1853</td>
</tr>
<tr>
<td>8</td>
<td>Cobol</td>
<td>IBM 43XX</td>
<td>11</td>
<td>130.3</td>
<td>200</td>
<td>1535</td>
</tr>
<tr>
<td>9</td>
<td>Cobol</td>
<td>IBM 308X</td>
<td>14</td>
<td>116</td>
<td>289</td>
<td>2491</td>
</tr>
<tr>
<td>10</td>
<td>Cobol, Natural</td>
<td>IBM 308X</td>
<td>5</td>
<td>72</td>
<td>39</td>
<td>542</td>
</tr>
<tr>
<td>11</td>
<td>Cobol</td>
<td>IBM 308X</td>
<td>13</td>
<td>258.7</td>
<td>254.2</td>
<td>983</td>
</tr>
<tr>
<td>12</td>
<td>Cobol</td>
<td>IBM 43XX, 308X</td>
<td>31</td>
<td>230.7</td>
<td>128.6</td>
<td>557</td>
</tr>
<tr>
<td>13</td>
<td>Cobol</td>
<td>HP 300, 68</td>
<td>20</td>
<td>157</td>
<td>161.4</td>
<td>1028</td>
</tr>
<tr>
<td>14</td>
<td>Cobol</td>
<td>IBM 308X</td>
<td>26</td>
<td>246.9</td>
<td>164.8</td>
<td>667</td>
</tr>
<tr>
<td>15</td>
<td>Natural</td>
<td>IBM 308X</td>
<td>14</td>
<td>69.9</td>
<td>60.2</td>
<td>861</td>
</tr>
</tbody>
</table>

Kemerer (1987) Dataset
APPENDIX B: MERMAID-2 DATASET

<table>
<thead>
<tr>
<th>Project Number</th>
<th>Adjusted FP</th>
<th>Raw FP</th>
<th>Total Effort (hours)</th>
<th>Total Duration (months)</th>
<th>Project Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23</td>
<td>23</td>
<td>238</td>
<td>3.45</td>
<td>E</td>
</tr>
<tr>
<td>2</td>
<td>38</td>
<td>42</td>
<td>490</td>
<td>6.75</td>
<td>E</td>
</tr>
<tr>
<td>3</td>
<td>36</td>
<td>44</td>
<td>616</td>
<td>2.9</td>
<td>E</td>
</tr>
<tr>
<td>4</td>
<td>57</td>
<td>51</td>
<td>910</td>
<td>2.55</td>
<td>E</td>
</tr>
<tr>
<td>5</td>
<td>36</td>
<td>47</td>
<td>1540</td>
<td>10</td>
<td>E</td>
</tr>
<tr>
<td>6</td>
<td>29</td>
<td>38</td>
<td>1680</td>
<td>10</td>
<td>E</td>
</tr>
<tr>
<td>7</td>
<td>23</td>
<td>34</td>
<td>1750</td>
<td>10.5</td>
<td>E</td>
</tr>
<tr>
<td>8</td>
<td>99</td>
<td>115</td>
<td>3234</td>
<td>9</td>
<td>N</td>
</tr>
<tr>
<td>9</td>
<td>605</td>
<td>550</td>
<td>3360</td>
<td>2</td>
<td>N</td>
</tr>
<tr>
<td>10</td>
<td>34</td>
<td>42</td>
<td>3850</td>
<td>5.5</td>
<td>E</td>
</tr>
<tr>
<td>11</td>
<td>338</td>
<td>317</td>
<td>5460</td>
<td>15</td>
<td>N</td>
</tr>
<tr>
<td>12</td>
<td>133</td>
<td>157</td>
<td>5110</td>
<td>16.25</td>
<td>E</td>
</tr>
<tr>
<td>13</td>
<td>118</td>
<td>107</td>
<td>6440</td>
<td>11</td>
<td>E;N</td>
</tr>
<tr>
<td>14</td>
<td>653</td>
<td>643</td>
<td>17920</td>
<td>35</td>
<td>E</td>
</tr>
<tr>
<td>15</td>
<td>502</td>
<td>528</td>
<td>18620</td>
<td>20</td>
<td>n/a</td>
</tr>
<tr>
<td>16</td>
<td>306</td>
<td>268</td>
<td>21280</td>
<td>27</td>
<td>N</td>
</tr>
<tr>
<td>17</td>
<td>170</td>
<td>179</td>
<td>24850</td>
<td>11.6</td>
<td>N</td>
</tr>
<tr>
<td>18</td>
<td>911</td>
<td>884</td>
<td>48230</td>
<td>29.6</td>
<td>E</td>
</tr>
<tr>
<td>19</td>
<td>221</td>
<td>235</td>
<td>3415</td>
<td>7.5</td>
<td>n/a</td>
</tr>
<tr>
<td>20</td>
<td>613</td>
<td>626</td>
<td>11551</td>
<td>7</td>
<td>N</td>
</tr>
<tr>
<td>21</td>
<td>1507</td>
<td>1408</td>
<td>4860</td>
<td>8.5</td>
<td>E</td>
</tr>
<tr>
<td>22</td>
<td>559</td>
<td>n/a</td>
<td>14224</td>
<td>26</td>
<td>E</td>
</tr>
<tr>
<td>23</td>
<td>218</td>
<td>291</td>
<td>9080</td>
<td>9</td>
<td>N</td>
</tr>
<tr>
<td>24</td>
<td>479</td>
<td>499</td>
<td>1635</td>
<td>9</td>
<td>E</td>
</tr>
<tr>
<td>25</td>
<td>26</td>
<td>33</td>
<td>296</td>
<td>4</td>
<td>E</td>
</tr>
<tr>
<td>26</td>
<td>125</td>
<td>1337</td>
<td>3720</td>
<td>5</td>
<td>E</td>
</tr>
<tr>
<td>27</td>
<td>205</td>
<td>n/a</td>
<td>4672</td>
<td>6</td>
<td>E</td>
</tr>
<tr>
<td>28</td>
<td>105</td>
<td>109</td>
<td>2065</td>
<td>8</td>
<td>E</td>
</tr>
<tr>
<td>29</td>
<td>114</td>
<td>107</td>
<td>1690</td>
<td>6</td>
<td>E</td>
</tr>
<tr>
<td>30</td>
<td>36</td>
<td>38</td>
<td>504</td>
<td>4</td>
<td>E</td>
</tr>
</tbody>
</table>

MERMAID-2 Dataset (Kitchenham 2002)
APPENDIX C: LINGO SCRIPT FOR TIER-THREE

MODEL:
SETS:
  GROUPS / R D I IT AT RD DI IT AT RAT DIT O /: n, EM, E, emu;
ENDSETS

E(1) = PR1 + PR7;
E(2) = PR4 + PR5 + PD2;
E(3) = PA3;
E(4) = IT5;
E(5) = 0;
E(6) = PR2 + PR3 + PD1 + PA1 + IT1;
E(7) = PD3 + PA2 + IT2;
E(8) = IT3;
E(9) = IT7;
E(10) = PR8 + PD7;
E(11) = PD4 + PA4 + IT4 + PD5 + PA5 + PA7;
E(12) = PR6 + PD6 + PA6 + IT6 + PD8 + PA8 + IT8;
Design = N(1);
Requirements = N(2);
Implementation = N(3);
Testing = N(4);
Customer = N(5);

@FOR (GROUPS(I) :
  EM(I) = 1 + (.001248269 * n(I) * (n(I) - 1)/2)) ;

@FOR (GROUPS(I) : @GIN(n(I)))
@FOR (GROUPS(I) : n(I) >= 1);
N(6) = N(1) + N(2);
N(7) = N(2) + N(3);
N(8) = N(3) + N(4);
N(9) = N(4) + N(5);
N(10) = N(1) + N(5);
N(11) = N(2) + N(4);
N(12) = N(1) + N(2) + N(3) + N(4) + N(5);

MINSTAFFCALC = @IF(MINSCHECK #EQ# 0, 5, MINSTAFF);
MAXSTAFFCALC = @IF(MAXSCHECK #EQ# 0, 10000, MAXSTAFF);
N(12) >= MINSTAFFCALC;
N(12) <= MAXSTAFFCALC;
COCOMOEFFORT = COCOEFFORT;
@FOR(GROUPS(I): emu(I) = E(I) * EM(I));
effort = @SUM(GROUPS : emu) * COCOMOEFFORT;
time = effort * (PR1 * (1 + PR6))/N(1) +
(PR2 * (1 + PR6))/(N(1) + N(2)) +
(PD1 * (1 + PD6 + PD8))/(N(1) + N(2)) +
(PD2 * (1 + PD6 + PD8))/(N(2)) +
(PD3 * (1 + PD6 + PD8))/(N(2) + N(3)) +
(PA3 * (1 + PA6 + PA8))/(N(3)) +
(IT3 * (1 + IT6 + IT8))/(N(3) + N(4)) +
(IT5 * (1 + IT6 + IT8))/(N(4)) +
(IT7 * (1 + IT6 + IT8))/(N(4) + N(5));
APPENDIX C: LINGO SCRIPT FOR TIER-THREE (continued)

DATA:
@POINTER( 1) = time;
@POINTER( 2) = Design;
@POINTER( 3) = Requirements;
@POINTER( 4) = Implementation;
@POINTER( 5) = Testing;
@POINTER( 6) = Customer;
PR1 = @POINTER( 7);
PR2 = @POINTER( 8);
PR3 = @POINTER( 9);
PR4 = @POINTER(10);
PR5 = @POINTER(11);
PR6 = @POINTER(12);
PR7 = @POINTER(13);
PR8 = @POINTER(14);

PD1 = @POINTER(15);
PD2 = @POINTER(16);
PD3 = @POINTER(17);
PD4 = @POINTER(18);
PD5 = @POINTER(19);
PD6 = @POINTER(20);
PD7 = @POINTER(21);
PD8 = @POINTER(22);

PA1 = @POINTER(23);
PA2 = @POINTER(24);
PA3 = @POINTER(25);
PA4 = @POINTER(26);
PA5 = @POINTER(27);
PA6 = @POINTER(28);
PA7 = @POINTER(29);
PA8 = @POINTER(30);

IT1 = @POINTER(31);
IT2 = @POINTER(32);
IT3 = @POINTER(33);
IT4 = @POINTER(34);
IT5 = @POINTER(35);
IT6 = @POINTER(36);
IT7 = @POINTER(37);
IT8 = @POINTER(38);
MINSTAFF = @POINTER(39);
MAXSTAFF = @POINTER(40);
MINSCHECK = @POINTER(41);
MAXSCHECK = @POINTER(42);
COCOEFFORT = @POINTER(43);

ENDDATA
MIN = time;
END
APPENDIX D: INSTITUTIONAL REVIEW BOARD APPROVAL

August 15, 2005

Michael Douglas and Alan Hevner
Information Systems / Decision Sciences
University of Arkansas at Little Rock
2801 South University Avenue
Little Rock, AR 72204-1099

RE: Approved Application for Continuing Review
IRB#: 101906
Title: The Impacts of the Handoffs on Software Development: A Cost Estimation Model
Start Approval Period: 08/11/2005 to 08/10/2006

Dear Mr. Douglas and Mr. Hevner:

On August 11, 2005, the Institutional Review Board (IRB) reviewed and APPROVED your Application for Continuing Review for the afore noted protocol. It was the determination of the IRB that your study qualified for expedited review based on the federal expedited category number six (6) and number seven (7), including the informed consent form. Approval is granted for the period indicated above.

Please note, if applicable, the enclosed informed consent/assent documents are valid during the period indicated by the official, IRB-Approval stamp located on page one of the form. Valid consent must be documented on a copy of the most recently IRB-approved consent form. Make copies of the enclosed original.

Please reference the above IRB protocol number in all correspondence regarding this protocol with the IRB or the Division of Research Compliance. In addition, we have enclosed an Institutional Review Board (IRB) Quick Reference Guide providing guidelines and resources to assist you in meeting your responsibilities in the conduction of human subjects research. Please read this brochure carefully. It is your responsibility to conduct this study in accordance with IRB policies and procedures and as approved by the IRB.

We appreciate your dedication to the ethical conduct of human subject research at the University of South Florida and your continued commitment to the Human Research Protections Program. If you have any questions regarding this matter, please call 813-974-9343.

Sincerely,

[Signature]

Paul G. Stiles, J.D., Ph.D.
USF Institutional Review Board

Enclosure(s): (If applicable) IRB-Approved, Stamped Informed Consent/Assent Documents(s)
IRB Quick Reference Guide
Cc: Brenda Kuska, USF IRB Professional Staff
Informed Consent
Social and Behavioral Sciences
University of South Florida

Information for People Who Take Part in Research Studies

The following information is being presented to help you decide whether or not you want to take part in a minimal risk research study. Please read this carefully. If you do not understand anything, ask the person in charge of the study.

Title of Study: A Software Cost Estimation Study

Principal Investigator: Michael Douglas

Study Location(s): University of South Florida, College of Business Administration

You are being asked to participate because we are examining the software cost estimation process.

General Information about the Research Study

The purpose of this research study is to analyze participants' process of estimating software development projects. The research study allows different estimation tasks to occur in an experimental setting. By participating in the experiment, more knowledge will be known about individuals that estimate software development projects.

Plan of Study

You will be required to conduct various software cost estimation tasks. You will be given information to estimate tasks and will be required to come up with an estimate of effort and schedule. In addition, some questions will be asked about your experiences with the estimation tasks. You will use a computer to help you estimate the software development costs. The whole study should take no more than 90 minutes.

Payment for Participation

You will not be paid for your participant in this study.

Benefits of Being a Part of this Research Study

You may get experience in conducting software cost estimation. By participating in this study, you may realize problems that occur in everyday software estimation and experience solutions to these problems.

Risks of Being a Part of this Research Study

There are no expected risks.

Confidentiality of Your Records

Your privacy and research records will be kept confidential to the extent of the law. Authorized research personnel, employees of the Department of Health and Human Services, and the USF Institutional Review Board and its staff, and other individuals, acting on behalf of USF, may inspect the records from this research project.

IRB Form: ICedub-LR-5Bv17
The results of this study may be published. However, the data obtained from you will be combined with data from others in the publication. The published results will not include your name or any other information that would personally identify you in any way.

Your identity is only required to sign this consent form. A code number will be assigned to each participant. Only the researchers in this study will have access to this data and the data will be kept on file in the office of the principal investigator.

Volunteering to Be Part of this Research Study

Your decision to participate in this research study is completely voluntary. You are free to participate in this research study or to withdraw at any time. There will be no penalty or loss of benefits you are entitled to receive, if you stop taking part in the study. The decision to participate or not to participate will in no way affect your student status or your grade.

Questions and Contacts

- If you have any questions about this research study, contact Michael Douglas at 501-663-7139
- If you have questions about your rights as a person who is taking part in a research study, you may contact the Division of Research Compliance of the University of South Florida at (813) 974-5638.
APPENDIX E: EXPERIMENTAL MATERIALS

A Software Cost Estimation Study

Thank You

Thank you for agreeing to participate in this study. Your participation in this study will lead to a better understanding of software cost estimation.

Informed Consent

First please read the informed consent document in the folder with this sheet. If you have any questions regarding this study please contact the principal investigator as listed on the informed consent document. After reading the consent document, if you wish to participate in the study please sign the informed consent document. The informed consent must be signed and returned.

This study has been approved by the University of South Florida Institution Review Board as an academic research study.

The Study

This study is designed to take no more than one hour. You are asked to report the starting and stopping time on the information sheet for each task.

This study consists of:
1) An IRB Form
2) A background questionnaire with demographic and software estimation experience.
3) The first estimation task with three subtasks.
4) The second estimation task with three subtasks.
5) An after experiment questionnaire that asks about the experiment.
6)
Informed Consent
Social and Behavioral Sciences
University of South Florida

Information for People Who Take Part in Research Studies

The following information is being presented to help you decide whether or not you want to take part in a minimal risk research study. Please read this carefully. If you do not understand anything, ask the person in charge of the study.

Title of Study: A Software Cost Estimation Study
Principal Investigator: Michael Douglas
Study Location(s): University of South Florida, College of Business Administration

You are being asked to participate because we are examining the software cost estimation process.

General Information about the Research Study
The purpose of this research study is to analyze participants’ process of estimating software development projects. The research study allows different estimation tasks to occur in an experimental setting. By participating in the experiment, more knowledge will be known about individuals that estimate software development projects.

Plan of Study
You will be required to conduct various software cost estimation tasks. You will be given information to estimate tasks and will be required to come up with an estimate of effort and schedule. In addition, some questions will be asked about your experiences with the estimation tasks. You will use a computer to help you estimate the software development costs.
The whole study should take no more than 90 minutes.

Payment for Participation
You will not be paid for your participation in this study.

Benefits of Being a Part of this Research Study

You may get experience in conducting software cost estimation. By participating in this study, you may realize problems that occur in everyday software estimation and experience solutions to these problems.

Risks of Being a Part of this Research Study
There are no expected risks.

Confidentiality of Your Records
Your privacy and research records will be kept confidential to the extent of the law. Authorized research personnel, employees of the Department of Health and Human Services, and the USF Institutional Review Board and its staff, and other individuals, acting on behalf of USF, may inspect the records from this research project.
The results of this study may be published. However, the data obtained from you will be combined with data from others in the publication. The published results will not include your name or any other information that would personally identify you in any way.

Your identity is only required to sign this consent form. A code number will be assigned to each participant. Only the researchers in this study will have access to this data and the data will be kept on file in the office of the principal investigator.

Volunteering to Be Part of this Research Study
Your decision to participate in this research study is completely voluntary. You are free to participate in this research study or to withdraw at any time. There will be no penalty or loss of benefits you are entitled to receive. If you stop taking part in the study, The decision to participate or not to participate will in no way affect your student status or your grade.

Questions and Contacts
- If you have any questions about this research study, contact Michael Douglas at 501-683-7139.
- If you have questions about your rights as a person who is taking part in a research study, you may contact the Division of Research Compliance of the University of South Florida at (813) 974-5638.
Consent to Take Part in This Research Study
By signing this form I agree that:

- I have fully read or have had read and explained to me this informed consent form describing this research project.
- I have had the opportunity to question one of the persons in charge of this research and have received satisfactory answers.
- I understand that I am being asked to participate in research. I understand the risks and benefits, and I freely give my consent to participate in the research project outlined in this form, under the conditions indicated in it.
- I have been given a signed copy of this informed consent form, which is mine to keep.

Signature of Participant ___________________________ Printed Name of Participant _________________ Date _______________

Investigator Statement
I have carefully explained to the subject the nature of the above research study. I hereby certify that to the best of my knowledge the subject signing this consent form understands the nature, demands, risks, and benefits involved in participating in this study.

Signature of Investigator ___________________________ Printed Name of Investigator _________________ Date _______________

Or authorized research investigator designated by the Principal Investigator
APPENDIX E: EXPERIMENTAL MATERIALS (continued)

Please answer the following background questions:

1. Age: _________________

2. Gender (circle one): Male Female

3. Current Work Status (circle one): Full-Time Part-Time Unemployed

4. What is your total full-time work experience in years? ___________________________

5. What is your total length of full-time IT experience in years? ______________________

6. What is your current role in your organization? ________________________________

7. How long have you been in your current role? _________________________________

8. What organizational level describes your position? [ ] Executive [ ] Middle Management [ ] Professional [ ] First Line Management [ ] Technical/Clerical [ ] Other

9. If you have estimated a project before, how many projects have you estimated? _____________________

10. What techniques have you used for estimation? (circle all that apply):

   | Ad Hoc | Informal analogy (rules-of-thumb) | Formal analogy (Example: A database of previous projects) | Formal model (Example: COCOMO) | Other (specify): ________________________ |
   | PRICE-S | COCOMO | COCOMO II | SLIM | 

11. If a formal model was used, which model(s) were used for estimation? (circle all that apply)

   | PRICE-S | COCOMO | COCOMO II | SLIM | Other (specify): ________________________ |

12. For what proportion of projects did you use an estimation technique other than Ad Hoc? (circle one)

   | None | 1-25% | 26-50% | 51-75% | > 75% |

13. Please list your educational background:

   | Bachelors: Degree Major |
   | Masters: Degree Major |
   | Doctorate Degree Major |

14. Please circle any professional certificates you have:
   a. PMI Project Management Professional Certificate (PMP)
   b. PMI Certified Associate in Project Management Certificate (CAPM)
   c. Working on certificate ________________________
   d. Other (please explain) ________________________

15. My use of estimation techniques is (circle only one)
   a. I have not used estimation techniques.
   b. I have used estimation in an initial project only.
   c. I have used in mostly small projects, but not in large projects.
   d. I have used in a mixture of small and large projects.
   e. I have used in mostly large projects, but not in small projects.
   f. My use of estimation is completely routine (in all my projects).

16. Typically, in which phase do you make your first estimate of software costs (e.g., budget, effort)?
   a. Requirements specification
   b. Software analysis
   c. Software design
APPENDIX E: EXPERIMENTAL MATERIALS (continued)

d. Implementation
e. Testing
f. Maintenance

17 Typically, in which phases, if any, do you revise your initial software cost estimate? (circle all that apply):
   a. Requirements specification
   b. Software analysis
   c. Software design
d. Implementation
e. Testing
f. Maintenance

The following questions measure your feelings about conducting software cost estimation. Please indicate the extent to which you agree or disagree with these statements by circling a number between 1 and 7 for each statement where:

1 = Strongly Disagree
2 = Somewhat Disagree
3 = Slightly Disagree
4 = Neutral
5 = Slightly Agree
6 = Somewhat Agree
7 = Strongly Agree

<table>
<thead>
<tr>
<th></th>
<th>1 = Strongly Disagree</th>
<th>2 = Somewhat Disagree</th>
<th>3 = Slightly Disagree</th>
<th>4 = Neutral</th>
<th>5 = Slightly Agree</th>
<th>6 = Somewhat Agree</th>
<th>7 = Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>I am capable of dealing with most estimation problems that come up at work.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>If I can’t estimate a project the first time, I keep trying until I can.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>When I set important goals for myself, I rarely achieve them.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>If estimation looks too complicated, I avoid it.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>When trying to estimate a new project, I soon give up if I am not initially successful.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>If a new estimation project seems especially difficult, I become more determined to master it.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Initial failure in estimation just makes me try harder.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>I feel confident about my ability to estimate projects.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>I am a self-reliant person in software cost estimation.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>I can come up with good estimates for straightforward projects.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Obstacles in estimating will not frustrate me.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>I can come up with estimates under any circumstances.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>I can come up with good estimates if I had a tool to help me.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>I can come up with good estimates if I see someone else estimating a project before I try it.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>I can come up with good estimates for projects similar to projects I previously estimated.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Task 1

Time Started: ____________

Time Stopped: ____________
APPENDIX E: EXPERIMENTAL MATERIALS (continued)

Estimation Task 1a:

In this task you are to estimate the amount of effort (in people-months) and schedule (in months) needed to develop the following software development project.

Assume the following conversions:

Effort:
1 people-month = 19 days.
1 working-day = 8 hours.

Schedule:
1 month = 30 days.

Estimation Details:

You are a project manager for a small software development company. Your organization consists of 12 total employees. All the employees work in a single development team throughout system development.

Project Information:

A database application is expected to be around 30 KDSI.

This development project is commonly conducted in this organization. The project is not complex; in fact, it will be a simple development project. The project will need to be reused. All the people on the team will be highly skilled team members and everyone works well together. The development platform the system will be developed on is commonly used throughout the organization.

Typically for this kind of project, the following historical data is available.

Historical Data:

<table>
<thead>
<tr>
<th>Estimated Effort (man-months)</th>
<th>Estimated Schedule (months)</th>
<th>Estimated Team Size (KDSI)</th>
<th>Actual Effort (man-months)</th>
<th>Actual Schedule (weeks)</th>
<th>Actual Schedule (months)</th>
<th>Actual Average Team Size</th>
<th>Project ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>42.25</td>
<td>9</td>
<td>5</td>
<td>74.25</td>
<td>54.5</td>
<td>35</td>
<td>8.75</td>
<td>6</td>
</tr>
<tr>
<td>169</td>
<td>13</td>
<td>13</td>
<td>183.5</td>
<td>252.5</td>
<td>89</td>
<td>22.25</td>
<td>11</td>
</tr>
<tr>
<td>26.25</td>
<td>6</td>
<td>4</td>
<td>16</td>
<td>19.5</td>
<td>20</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>38</td>
<td>12.99</td>
<td>3</td>
<td>15</td>
<td>38</td>
<td>58.43</td>
<td>14.6075</td>
<td>3</td>
</tr>
</tbody>
</table>

Estimated and Actual Size is in KDSI (1 KDSI = 1000 Lines of Code)
APPENDIX E: EXPERIMENTAL MATERIALS (continued)

Please answer the following questions regarding the project:

Effort:

1. Please estimate (in people-months) how much effort will be required to conduct the project _____________.

2. Please state how confident (from 0% to 100%) you are about your effort estimate _____.

3. Please give a worst case estimate of effort_______________________________.

4. Please give a best case estimate of effort _________________________________.

5. Please describe your rationale for your estimate of effort.

Schedule:

6. Please estimate how long it will take to conduct the project ___________________.

7. Please state how confident (from 0% to 100%) you are about your effort estimate _____.

8. Please give a worst case estimate of schedule _____________________________.

9. Please give a best case estimate of schedule _______________________________.

10. Please describe your rationale for your estimate of effort.
Estimation Task 1b:

In this task you are to estimate the amount of *effort* (in people-months) needed to develop the following software development project.

Task 1b is the same as Task 1a except for the following:

The amount of staff of the project is doubled to 24 employees.

Please answer the following questions regarding the project:

**Effort:**

11. Please estimate (in people-months) how much effort will be required to conduct the project _____________.

12. Please state how *confident* (from 0% to 100%) you are about your effort estimate _____.

13. Please give a *worst case* estimate of effort ________________________________.

14. Please give a *best case* estimate of effort ________________________________.

15. Please describe you rationale for your estimate of effort.

**Schedule:**

16. Please estimate how long it will take to conduct the project ____________________.

17. Please state how *confident* (from 0% to 100%) you are about your schedule estimate ___.

18. Please give a worst case estimate of schedule ____________________________________.

19. Please give a best case estimate of schedule ____________________________________.

20. Please describe you rationale for your estimate of effort.
APPENDIX E: EXPERIMENTAL MATERIALS (continued)

Estimation Task 1c:

In this task you are to estimate the amount of **effort** (in people-months) and **schedule** (in months) needed to develop the following software development project.

Assume the following conversions:

**Effort:**
- 1 people-month = 19 days.
- 1 working-day = 8 hours.

**Schedule:**
- 1 month = 30 days.

Estimation Details:

You are a project manager for a small software development company. Your organization consists of 13 total employees. The employees all work in a single development team throughout system development.

Project Information:

A database application is expected to be around 180 KDSI.

This development project is commonly conducted in this organization. The project is not complex; in fact, it will be a simple development project. The project will need to be reused. All the people on the team will be highly skilled team members and everyone works well together. The development platform the system will be developed on is commonly throughout the organization.

Typically for this kind of project, the following historical data is available.

Historical Data:

<table>
<thead>
<tr>
<th>Estimated Effort (man-months)</th>
<th>Estimated Schedule (months)</th>
<th>Estimated Team Size (KDSI)</th>
<th>Actual Effort (man-months)</th>
<th>Actual Schedule (weeks)</th>
<th>Actual Schedule (months)</th>
<th>Actual Average Team Size</th>
<th>Project ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>42.25</td>
<td>9</td>
<td>5</td>
<td>54.5</td>
<td>35</td>
<td>8.75</td>
<td>6</td>
<td>3004</td>
</tr>
<tr>
<td>169</td>
<td>13</td>
<td>13</td>
<td>183.5</td>
<td>252.5</td>
<td>89</td>
<td>22.25</td>
<td>11</td>
</tr>
<tr>
<td>26.25</td>
<td>6</td>
<td>4</td>
<td>16</td>
<td>19.5</td>
<td>20</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>38</td>
<td>12.99</td>
<td>3</td>
<td>15</td>
<td>38</td>
<td>58.43</td>
<td>14.6075</td>
<td>3</td>
</tr>
</tbody>
</table>

Estimated and Actual Size is in KDSI (1 KDSI = 1000 Lines of Code)

174
APPENDIX E: EXPERIMENTAL MATERIALS (continued)
Please answer the following questions regarding the project:

Effort:

21. Please estimate (in people-months) how much effort will be required to conduct the project _____________.

22. Please state how confident (from 0% to 100%) you are about your effort estimate _____.

23. Please give a worst case estimate of effort ________________________________.

24. Please give a best case estimate of effort ________________________________.

25. Please describe your rationale for your estimate of effort.

Schedule:

26. Please estimate how long it will take to conduct the project _________________.

27. Please state how confident (from 0% to 100%) you are about your schedule estimate ___.

28. Please give a worst case estimate of schedule ________________________________.

29. Please give a best case estimate of schedule ________________________________.

30. Please describe your rationale for your estimate of effort.
Task 2

Time Started: ____________

Time Stopped: ____________
APPENDIX E: EXPERIMENTAL MATERIALS (continued)

Estimation Task 2a:

In this task you are to estimate the amount of **effort** (in people-months) and **schedule** (in months) needed to develop the following software development project.

Assume the following conversions:

**Effort:**
1 people-month = 19 days.
1 working-day = 8 hours.

**Schedule:**
1 month = 30 days.

Estimation Details:

You are a project manager for a medium sized software development company. For this particular project you are to manage 30 staff. All the employees work in a single development team also known as an integrated project team throughout system development.

Project Information:

An ecommerce web application is expected to be around 80 KDSI.

The web application is a business-to-business e-commerce project. Important stock transaction data will be routed through this application allowing mutual fund companies to trade stocks directly to other mutual funds.

The following historical data is available for past projects, but the historical data is not expected to be helpful since this project will have team sizes more than double past projects.

**Historical Data:**

<table>
<thead>
<tr>
<th>Estimated Effort (man-months)</th>
<th>Estimated Schedule (months)</th>
<th>Estimated Team Size (KDSI)</th>
<th>Actual Effort (man-months)</th>
<th>Actual Schedule (weeks)</th>
<th>Actual Schedule (months)</th>
<th>Actual Average Team Size</th>
<th>Project ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>42.25</td>
<td>9</td>
<td>5</td>
<td>74.25</td>
<td>54.5</td>
<td>35</td>
<td>8.75</td>
<td>6</td>
</tr>
<tr>
<td>169</td>
<td>13</td>
<td>13</td>
<td>183.5</td>
<td>252.5</td>
<td>89</td>
<td>22.25</td>
<td>11</td>
</tr>
<tr>
<td>26.25</td>
<td>6</td>
<td>4</td>
<td>16</td>
<td>19.5</td>
<td>20</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>38</td>
<td>12.99</td>
<td>3</td>
<td>38</td>
<td>58.43</td>
<td>14.6075</td>
<td>3</td>
<td>8965</td>
</tr>
</tbody>
</table>
APPENDIX E: EXPERIMENTAL MATERIALS (continued)
Estimated and Actual Size is in KDSI (1 KDSI = 1000 Lines of Code)
Please answer the following questions regarding the project:

Effort:

31. Please estimate (in people-months) how much effort will be required to conduct the project _____________.

32. Please state how confident (from 0% to 100%) you are about your effort estimate _____.

33. Please give a worst case estimate of effort ____________________________________.

34. Please give a best case estimate of effort ____________________________________.

35. Please describe your rationale for your estimate of effort.

Schedule:

36. Please estimate how long it will take to conduct the project _____________________.

37. Please state how confident (from 0% to 100%) you are about your schedule estimate ____.

38. Please give a worst case estimate of schedule ____________________________________.

39. Please give a best case estimate of schedule ____________________________________.

40. Please describe your rationale for your estimate of effort.
APPENDIX E: EXPERIMENTAL MATERIALS (continued)

Estimation Task 2b:

In this task you are to estimate the amount of **effort** (in people-months) and **schedule** (in months) needed to develop the following software development project.

Task 2b is the same as Task 2a except for the following:

Instead of developing the system in one large integrated project team, the project will be broken into three different teams.

The first team will be the requirements and design team. This team will consist of 9 people. The second team will be the implementation team and will consist of 13 people. The third team is the testing team and will consist of 8 people.

Notice, there is still a total of 30 people working on the system development.

Please answer the following questions regarding the project:

**Effort:**

41. Please estimate (in people-months) how much effort will be required to conduct the project ____________.

42. Please state how **confident** (from 0% to 100%) you are about your effort estimate _____.

43. Please give a **worst case** estimate of effort ____________________________________.

44. Please give a **best case** estimate of effort ____________________________________.

45. Please describe you rationale for your estimate of effort.

**Schedule:**

46. Please estimate how long it will take to conduct the project ________________.
47. Please state how confident (from 0% to 100%) you are about your schedule estimate ___.

48. Please give a worst case estimate of schedule ________________________________.

49. Please give a best case estimate of schedule ________________________________.

50. Please describe your rationale for your estimate of effort.
APPENDIX E: EXPERIMENTAL MATERIALS (continued)

Estimation Task 2c:

In this task you are to estimate the amount of effort (in people-months) and schedule (in months) needed to develop the following software development project.

Task 2c is the same as Task 2a except for the following:

Instead of developing the system in one large integrated project team, the project will be broken into five different teams.

The first team will be the requirements team. The requirements team will consist of 4 people. The second team will be the design team. Design team will consist of 6 people. The third team will be the implementation team and will consist of 12 people. The fourth team is the testing team and will consist of 7 people. The fifth team is the customer acceptance team. One person will perform all the customer acceptance activities.

Notice, there is still a total of 30 people working on the system development.

Please answer the following questions regarding the project:

Effort:

51. Please estimate (in people-months) how much effort will be required to conduct the project ____________.

52. Please state how confident (from 0% to 100%) you are about your effort estimate ________.

53. Please give a worst case estimate of effort ____________________.

54. Please give a best case estimate of effort ____________________.

55. Please describe you rationale for your estimate of effort.
Schedule:

56. Please estimate how long it will take to conduct the project ____________________.

57. Please state how confident (from 0% to 100%) you are about your schedule estimate ___.

58. Please give a worst case estimate of schedule ____________________________.

59. Please give a best case estimate of schedule ____________________________.

60. Please describe your rationale for your estimate of effort.
AFTER TASKS QUESTIONNAIRE
APPENDIX E: EXPERIMENTAL MATERIALS (continued)

The following questions measure your feelings based on your experience in estimating in the experiment. Please indicate the extent to which you agree or disagree with these statements by circling a number between 1 and 7 for each statement where:

1 = Strongly Disagree
2 = Somewhat Disagree
3 = Slightly Disagree
4 = Neutral
5 = Slightly Agree
6 = Somewhat Agree
7 = Strongly Agree

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>Using the software estimation technique in this experiment improves my performance in conducting software estimation.</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>34</td>
<td>Using the software estimation technique in this experiment improves my productivity in conducting software estimation.</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>35</td>
<td>Using the software estimation technique in this experiment improves my effectiveness in conducting software estimation.</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>36</td>
<td>Overall, the software technique used in this experiment was useful in conducting software cost estimation.</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

How would you rate your overall experience using the software estimation technique in this experiment (4=Neutral):

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>Very Dissatisfied</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>38</td>
<td>Very displeased</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>39</td>
<td>Very frustrated</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>40</td>
<td>Absolutely terrible</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

In making my software cost estimates, the technique I mainly used:

a. A Calculator
b. Spreadsheet
c. Historical Data
d. Historical Data along with COCOMO II
e. Historical Data and PSEstimate
f. Other (please specify) __________________________________

During the study, circle all of the following techniques that you used to make software cost estimates:

a. A Calculator
b. Spreadsheet
c. Historical Data
d. COCOMO II
e. PSEstimate
f. Other (please specify) __________________________________

My preferred method to estimate software cost is to use ________________ to come up with my estimates.

When conducting Task 2, how do you think the difference in structures changed the communication that occurred as the same thirty people moved smaller group
ABOUT THE AUTHOR

Michael Douglas holds a B.S. in Computer Engineering from Kansas State University and a M.B.A. from Fontbonne University in St. Louis, MO. Michael is starting his career at the University of Arkansas at Little Rock. Michael is happily married to Kelly. Michael and Kelly have a poodle named Cody.