

1-1-2007

The relationship between project characteristics and the expert estimation of software development and maintenance

John Farrish
University of Nevada, Las Vegas

Follow this and additional works at: <https://digitalscholarship.unlv.edu/rtds>

Repository Citation

Farrish, John, "The relationship between project characteristics and the expert estimation of software development and maintenance" (2007). *UNLV Retrospective Theses & Dissertations*. 2163.
<http://dx.doi.org/10.25669/nikg-aa0n>

This Thesis is protected by copyright and/or related rights. It has been brought to you by Digital Scholarship@UNLV with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you need to obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/or on the work itself.

This Thesis has been accepted for inclusion in UNLV Retrospective Theses & Dissertations by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.

THE RELATIONSHIP BETWEEN PROJECT CHARACTERISTICS
AND THE EXPERT ESTIMATION OF SOFTWARE
DEVELOPMENT AND MAINTENANCE

by

John Farrish

Associate of Applied Science
ITT Technical Institute
2002

Bachelor of Science
University of Phoenix
2004

A thesis submitted in partial fulfillment
of the requirements for the

**Master of Science Degree in Management Information Systems
Department of Management Information Systems
College of Business**

**Graduate College
University of Nevada Las Vegas
August 2007**

UMI Number: 1448395

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI[®]

UMI Microform 1448395

Copyright 2007 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346



Thesis Approval

The Graduate College
University of Nevada, Las Vegas

August 17, 2007

The Thesis prepared by

John Farrish

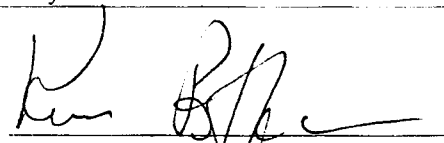
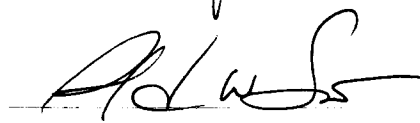
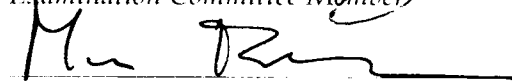
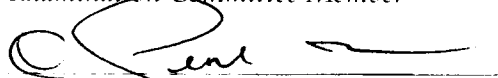
Entitled

The Relationship Between Project Characteristics and the Expert

Estimation of Software Development and Maintenance

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Management Information Systems


Examination Committee Chair
Dean of the Graduate College
Examination Committee Member
Examination Committee Member
Graduate College Faculty Representative

ABSTRACT

The Relationship Between Project Characteristics and the Expert Estimation of Software Development and Maintenance

by

John Farrish

Dr. Ken Peffers, Examination Committee Chair
Professor of Management Information Systems and Department Chair
University of Nevada Las Vegas

Accurately estimating the amount of time and effort required to complete a software development or maintenance project has proven problematic for business. A wealth of literature exists exploring each of the methods for estimating software development, but very little is devoted to understanding how project characteristics relate to estimation accuracy. This research examines expert estimation, the most widely used estimation technique, to determine the relationship between software project characteristics and estimation accuracy. Implications of the findings for research and practice are discussed.

TABLE OF CONTENTS

ABSTRACT.....	iii
ACKNOWLEDGEMENTS.....	vi
LIST OF TABLES.....	vii
CHAPTER 1 INTRODUCTION.....	1
CHAPTER 2 LITERATURE REVIEW.....	4
Algorithmic Models.....	5
Algorithmic Models – Line of Code.....	7
Algorithmic Models – The Putnam SLIM Model.....	8
Algorithmic Models – COCOMO.....	9
Algorithmic Models – Function Point Counting.....	11
Expert Estimation.....	13
Expert Estimation by Analogy or Case-Based Reasoning.....	14
Expert Estimation – Work Breakdown Structures.....	15
Simulation.....	17
Effectiveness of the Various Techniques.....	17
CHAPTER 3 RESEARCH MODEL.....	21
Data.....	25
Kitchenham Data.....	25
Jorgensen Data.....	26
HMO Data.....	28
CHAPTER 4 ANALYSIS AND RESULTS.....	29
CHAPTER 5 DISCUSSION.....	44
CHAPTER 6 IMPLICATIONS, LIMITATIONS, AND SUGGESTIONS.....	49
Implications for Business.....	49
Limitations and Suggestions for Further Research.....	50
APPENDIX.....	52
HMO Data Set.....	52
Jorgensen Data Set.....	53
REFERENCES.....	56

VITA.....	59
-----------	----

ACKNOWLEDGEMENTS

The author would like to thank those without whom this effort would not have been possible:

Garianne Farrish, whose patience, love, and sacrifices have been beyond all expectation.

Dr. Ken Peffers, who has provided help and guidance far above and beyond the call of duty.

Dr. Pearl Brewer for agreeing to be the Graduate College Faculty Representative and for her kind, earnest attention.

Dr. Marcus Rothenberger and Dr. Honghui Deng for agreeing to sit on the examination committee and their work pursuant to that.

All the member of the MIS faculty at UNLV who have provided me with the knowledge necessary to complete this project: Dr. Reza Torkzadeh, Dr. Tae Dong Hahn, Dr. Trevor Moores, Dr. Matt Thatcher, and Dr. Jerry Chang.

Dr. Magne Jørgensen for the contribution of his data set.

Helen Gerth who guided me through bureaucratic waters with consummate skill.

LIST OF TABLES

Table 1	Estimation Techniques: Advantages and Limitations.....	6
Table 2	Mean Actual Project Effort and Duration and Mean Underestimation...	30
Table 3	Estimated Coefficients for Saturation Model (1).....	33
Table 4	Estimated Coefficients for Model (1) – 22 nd Regression.....	34
Table 5	Estimated Coefficients for Model (1) – 29th Regression.....	35
Table 6	Estimated Coefficients for Saturation Model (2).....	37
Table 7	Estimated Coefficients for Model (2) – 22 nd Regression.....	38
Table 8	Estimated Probability Significance for Saturation Model (3).....	41
Table 9	Probability Significance for Reduced Model (3).....	42
Table 10	Comparison of Means for HMO Data Set.....	43
Table 11	Analysis of Support for Hypotheses.....	45

CHAPTER 1

INTRODUCTION

Estimating the cost in time, effort, and money required to develop software has been very problematic for businesses over the years. Early methodologies developed for estimating these costs have not improved much in the last twenty years; yet increasing estimation accuracy can mitigate risk more than any other cost-related parameter (Pfleeger, et. al., 2005). Software development, which involves a great number of related factors that have an effect on project outcome and forecasting accurately, has proven difficult because many of these relationships are still not well understood (Finnie, et. al., 1997).

Much research has been devoted to understanding why this is so. For the most part, this research has centered on the individual estimation methods and whether they produce accurate results. The greater part of current research has not taken into account the type of project being developed; it has looked at estimation techniques, irrespective of the size and scope of the project being estimated. The author can find little, if any, evidence that research has been devoted to determining whether certain project characteristics have an effect on software development or maintenance estimation accuracy.

There are four major reasons why software cost estimates are generally inaccurate (Kemerer, 1991). The first is that developing an accurate estimate is a quite complex task that involves a great deal of effort that most people are not willing to make. The second is that most of the people generating the estimates do not have a great deal of experience at developing these estimates, especially for larger projects. The third is that there is a natural human tendency to underestimate the amount of effort required to complete a task. The fourth problem is that managers will often ask for an estimate when what they really want is a goal and employees know this. When the third and fourth problems are combined, the results can sometimes be disastrous.

This problem is of importance because both underestimating and overestimating the time and effort required to develop software have negative implications for an organization. If development projects are underestimated, they are often released prematurely because a budgetary limit has been exceeded (if indeed they are released at all). These projects tend to be rife with errors and omissions and are rarely tested properly (Kemerer, 1987). Underestimating can also tie organizations to projects that would never have been undertaken had the true costs been known while robbing them of functionality while systems are off line.

Overestimating projects also has negative consequences. Inflating the estimated cost in either time or money may actually cause the project cost to rise as work expands to fill the time or budget allotted to it. Overestimated projects also tend to fall prey to scope creep as developers take the extra time and money given them and add unnecessary bells and whistles (Kemerer, 1991). Most importantly, overestimation may cause

projects possessing a real potential for benefit to be rejected as being too expensive (Vicinanza, et. al., 1991).

The numbers are well known to anyone involved with information systems; the overwhelming majority of software development projects fail to finish on time and within budget. As of twenty years ago nearly fifteen percent of all development projects were abandoned altogether prior to completion due to cost overruns (Jones, 1986); the numbers have not improved significantly since (Briand, 1998 and Jørgensen, 2004).

There are essentially two major types of estimation techniques; model-based and expert-based (Menzies, et. al., 2006), each of which can be broken down into smaller sub-groups. Of interest to this study are the expert-based techniques. Expert estimation was chosen because it enjoys, by far, the most widespread use of any of the major estimation techniques (Briand, et. al., 1998). This study will examine expert estimation of software development and maintenance projects to see which project characteristics have an effect on effort and duration estimation accuracy.

The methodology will involve the examination of one new and two existing data sets. The data will be analyzed using linear and logistic regression techniques as well as by a comparison of means. The results will extend the current literature by showing which project characteristics have a demonstrable effect on effort estimation accuracy, duration estimation accuracy, or both. This information will be value to any organization involved in the maintenance or development of software systems.

CHAPTER 2

LITERATURE REVIEW

The literature review surveys the current state of thought about the major estimation techniques and their effectiveness. There exist essentially two different types of software cost estimation techniques: model-based and expert-based (Menzies, et. al., 2006). These two techniques can be further broken down into seven distinct subgroups (Boehm, 1984). Each of these techniques involves making decisions in conditions of uncertainty and each seeks to mitigate the inherent risk through the use of economic analysis. Some of these analytic techniques concern themselves with making decisions in conditions of complete uncertainty, but are not practical for software engineering problems. There are two other analytic processes that can be of value, however, and it will be useful to examine them before proceeding further.

The first is the expected value technique which estimates the cost of both success and failure and figures the probability of each occurring. The expected value can then be determined mathematically in this fashion:

$$EV = \text{Prob}(\text{success}) * \text{Payoff}(\text{successful OS}) + \text{Prob}(\text{failure}) * \\ \text{Payoff}(\text{unsuccessful OS}) \quad (\text{Boehm, 1984})$$

Expected value techniques are better than estimating in conditions of complete uncertainty, but there is still a great deal of risk involved if the probability of failure is

underestimated as this will consequently lower the cost of failure in the above equation (Boehm, 1984).

The second type of economic analysis involves the buying of information. Prototyping is the most common form of information buying (Sparling, 2003). Prototyping allows a developer to have a greater understanding of high risk elements of a development project before getting too deeply involved. The buying of information does beg the question, however: how much do you invest and at what point do you have so much invested in the buying of information that there is a great deal of pressure generated to proceed with an otherwise untenable project?

Each of the seven cost estimation techniques uses one or more of these economic assessment tools to a greater or lesser extent. What follows is a brief examination of each of these techniques, paying special attention to the expert-based models.

Algorithmic Models

Algorithmic models involve the use of algorithms that generate a cost estimate that is determined by identifying the variables that are seen as the primary (and sometimes secondary) cost drivers (Boehm, 1984). Algorithmic models have the advantage of being objective (for the most part) and repeatable (McConnell, 2006). Their objectivity is limited, however, by the subjectivity of the inputs. Algorithmic models are also efficient and objectively aligned to experience, but experience represents past performance which may not be an indication of future results.

Table 1

Estimation Techniques: Advantages and Limitations

Method	Citation	Advantages	Limitations
Model-based			
Lines of Code	McConnell (2006) Boehm (1984)	<ul style="list-style-type: none"> • Objective • Repeatable 	<ul style="list-style-type: none"> • LOC estimate generated early in process • No consideration of CASE tools, etc.
Putnam SLIM	Putnam (1978, 1982)	<ul style="list-style-type: none"> • Can be calibrated to past projects • Allows for benchmarks, cash flow, etc. 	<ul style="list-style-type: none"> • Asserts trade-off between effort and time • Relies on early estimate of LOC
COCOMO	Boehm (1984)	<ul style="list-style-type: none"> • Fairly detailed estimate with reasonable effort • Accounts for 15 cost drivers 	<ul style="list-style-type: none"> • Relies on early estimate of LOC
Function Point Counting	Albrecht (1979, 1984)	<ul style="list-style-type: none"> • Function points easily determined through requirements elicitation • Easily understood • Independent of technology 	<ul style="list-style-type: none"> • Classification of system components overly simple • May underestimate system complexity
Expert-based			
Expert Estimation	Paynter (1996) Hihn and Habib-Agahi (1991)	<ul style="list-style-type: none"> • Considered to be as accurate as more expensive techniques • Less time consuming 	<ul style="list-style-type: none"> • Based on intuition rather than fact • Relies on availability of expert
Analogy	Pfleeger, et. al. (2006) Vicinanza, et. al. (1991)	<ul style="list-style-type: none"> • Compares estimated project to known quantity • Easy to account for dissimilarities between analogues 	<ul style="list-style-type: none"> • Requires detailed case knowledge and expert estimator • Compares present-day project to past development
Work Breakdown Structures	Jørgensen (2004) NASA (2002)	<ul style="list-style-type: none"> • Creates small parts easily estimated • Easily adaptable 	<ul style="list-style-type: none"> • Resource intensive • Requires detailed specifications

Algorithmic Models – Lines of Code (LOC)

One of the oldest methods for estimating the amount of effort required to develop a piece of software involves determining the number of lines of code required to write the software. This is essentially a two step process (Heiat and Heiat, 1997). The first step involves estimating the number of lines of code required for the information system. The second step requires the calculation of total effort using a formula based on historical data from previous projects. This formula is of the form:

$$EFH = c(LOC)^k$$

where EFH is the estimated effort in person-hours, weeks, or months, c and k are constants, and LOC is the estimated number of lines of code in the proposed application (Boehm, 1984). The limitations of this approach seem obvious.

The first is that the entire process depends on an estimate of the total lines of code required that is generated very early in the development process. This estimate is almost always based on the experience of expert estimators. A second problem is that the LOC model does not consider certain resources available to the development team like CASE tools and the experience of the team itself (Jones, 1986). Finally, and most importantly, the LOC method does not necessarily provide accurate estimates even if the design requirements are specified in detail. Conte, et. al. (1986) provided several experienced project managers with detailed design specifications for sixteen already completed projects. Each manager was asked to estimate the system size in lines of code; each consistently underestimated the actual system size.

Algorithmic Models - The Putnam SLIM Model

The Putnam SLIM model is actually a software application available commercially from Quantitative Software Management, Inc. It is based on L. H. Putnam's analysis of the software development life cycle and focuses on the number of project personnel and the time allotted to them. The basic mathematical model used in SLIM is

$$S_s = C_k K^{1/3} t_d^{4/3}$$

where

S_s = number of delivered source instructions

K = life-cycle effort in person-years

t_d = development time in years

C_k = a "technology constant" (Putnam, 1978)

Values for C_k can vary widely, ranging from about 600 to over 57,000, and it can be either calibrated to past projects or estimated as a function of modern programming practice use, to reflect hardware constraints and/or personnel experience, and certain other factors. The SLIM model also contains extensions that allow the estimator to determine projections for manpower distribution, cash flow, benchmarks, and documentation costs (Putnam, 1982).

SLIM is not without its critics, however. The most contentious assertion SLIM makes is that there exists a trade-off relationship between development effort and development time (i.e., K is directly related to t_d). The primary SLIM equation (from above) tells us that

$$K = \text{constant} / t_d^4.$$

According to this equation it would be possible to cut the cost of a software development project in half by increasing its development time by 19 percent (Boehm, 1984). This notion seems counterintuitive. It would seem apparent that the cost of any project would increase, not decrease with time. Taken to its logical extreme, this would mean that a project of infinite duration would have zero cost.

Algorithmic Models – COCOMO

Like LOC and SLIM, the constructive cost model (COCOMO) represents one of the earlier attempts to create an algorithmic method for estimating development times, and it has enjoyed fairly widespread popularity. Developed by Barry Boehm for TRW, COCOMO's primary motive is to draw a clear connection between management decisions in the commissioning and development of software and the consequences of those decisions.

COCOMO is actually three different models, each of which generates an increasingly detailed estimate and the first and third of which are rarely used. The first, which provides a single macro-estimation, does not provide quite enough information to be of a great deal of use as it provides an overly generalized picture of the project (Boehm, 1984). The third model provides a micro-estimation that includes a three-level work breakdown structure and an array of what Boehm calls "cost driver attributes," each of which takes on a different value depending on the phase of the development process. This is a very detailed process that requires a great deal of study and expert analysis. Because of its unwieldy nature, very few estimators utilize this third approach.

The second of the three models, however, provides a reasonably detailed estimate without requiring too much detailed effort. This model consists of four steps. The first step involves estimating a nominal development effort based on the project's size as measured in thousands of delivered source instructions (KDSI). In other words, it begins by estimating lines of code.

The second step involves the creation of a set of "effort multipliers" that are based on the project's ratings on Boehm's set of fifteen cost driver attributes. These attributes include product attributes, computer attributes, personnel attributes, and project attributes. The next step then generates the estimated development effort by multiplying the nominal effort estimate by the project's effort multipliers. Finally, additional factors are used to determine costs, development schedules, labor distributions, maintenance costs, and other such sundry estimates.

Each of these algorithmic models shares one serious limitation: they all rely on an estimation of lines of code as a starting point for the estimation process. This reliance has two drawbacks. First, the estimation of lines of code can only be, at best, an educated guess. Like expert estimation, the quality of this guess is dependent upon the abilities of the person making the guess (Conte, et. al., 1986). Therefore, an inexperienced project manager might very easily make a flawed judgment at the very beginning of the estimation process. Second, the idea that lines of code is the best predictor of project effort requirements is outdated (Hihn and Habib-Agahi, 1991). There exists an entire array of tools available to developers to mitigate the necessity for writing code; not one of these algorithmic methods really takes this into account.

Algorithmic Models - Function Point Counting

In order to address some of the concerns A. J. Albrecht developed an alternate methodology for determining the amount of work effort in a given project. Albrecht's approach is to "list and count the number of external user inputs, inquiries, outputs, and master files to be delivered by the development project (Albrecht, 1979)." These factors represent all the functions of any given application. Each of these functions is then given a numerical weight that corresponds to its value to the user. The weighted sum of these inputs and outputs is known as function points.

One of the advantages of this approach is that function points are easily determined as a result of the requirements elicitation process. Hence, they are uncovered at an early stage of development and they are far more easily understood by end users (Albrecht and Gaffney, 1983). Each of the different types of function points is ranked as simple, average, or complex and is weighted according to a mathematical formula. These "unadjusted function points" are then multiplied by a "technical complexity factor" to arrive at a final number of function points. This final number has no dimensions; it is simply a measure of system size that can be related to other systems that have also been measured in terms of function points.

Function points are effective as a measurement of system size because first, the measure is based on an external view of the system and is independent of technology. Second, the measure is determined early in the SDLC which allows for the use of function points in the estimation process. Finally, function points are easily understood, even by non-technical users (Albrecht, 1984).

Still, there are drawbacks to the function point approach. The classification of system component types as simple, average, or complex seems overly simple (Symons, 1988). A system component of many hundreds of data elements has, at most, only twice the weighted complexity of a simple component. Second, it is possible that the function point counting method underestimates the complexity of systems which are complex internally and have larger numbers of data elements (Kemerer, 1993).

Despite these drawbacks, function point analysis would seem to have significant advantages over methods previously discussed. First, there is very little subjectivity involved; very few value judgments have to be made. Inputs and outputs are just that; lines of code become almost irrelevant. Hence, the necessity for basing an estimate on an educated guess is eliminated. Also, the ease with which function points are understood not only brings non-technical people into the development process, it also makes it possible for less experienced project managers and developers to make vital contributions. It has also been determined that reassessing the function point counts at critical junctures in the development process can yield extremely valuable information for evaluating design and implementation efficiency (Orr and Reeves, 1999).

It is also critical to note that function points can be used to estimate the complexity of a given system. It stands to reason that the greater the number of inputs, outputs, etc. that a system has to process the more complex that system will be. The fact that function points have no units or dimensions also makes them fairly simple to use. Closely related to function point counting is widget counting which, instead of identifying input, outputs, etc., it identifies repeated characteristics of system development (“widgets”) and assigns a complexity factor to each (Kitchenham, 2002).

Experience is used as a guide to determining how much effort will be required to produce each widget and all widget estimates are summed. Estimates of effort for supporting tasks are then added to the widget sum, and an overall estimate is arrived at.

Expert Estimation

Expert estimation (or judgment) involves an experienced developer making an educated guess about the length of time and/or amount of effort necessary for a particular development or maintenance project. A significant portion of the estimate must be based on intuition. While this might not seem to be the best way of estimating development, the overwhelming majority of development projects use this technique. Many studies back this up; one study done at the Jet Propulsion Laboratory found that 83% of estimators used “informal analogy” as their primary estimation technique (Hihn and Habib-Agahi, 1991). An investigation into software development practices in the Netherlands found that 62% of organizations that produced development estimates did so based on “intuition and experience” (Heemstra and Kusters, 1991). Paynter (1996) determined that fully 86% of software development organizations in New Zealand based their estimates on expert estimation.

Expert estimation is used quite a bit because the preponderance of opinion is that the estimates generated are as good as those generated by other, more expensive and time-consuming models. The literature seems to bear this out. Vicinanza, et. al. (1991) compared the estimation accuracy of five software professionals with that of estimation models using both function points and the constructive cost model (COCOMO) and

found that the expert estimators had both the most and least accurate estimates but were, on average, more accurate than the models.

Expert Estimation by Analogy or Case-Based Reasoning

The next estimation method to be discussed involves analogical or case-based estimation. Simply put, analogical problem solving involves examining the current problem (or target) and relating it to some similar, previously solved problem (the source). Both analogy and the next method to be examined, work breakdown structures, are really subsets of expert estimation. Expert estimation involves an expert's comparison of a current project with a similar project in his or her own past. Analogy simply formalizes this process. The analogy is formalized when the similarities between the target and the source are both real and demonstrable within the same problem context. The problem solver must retrieve, either from memory or through research, several cases similar to the target and analyze them for similarities. The most appropriate one is then selected as the source (Pfleeger, et. al., 2005).

The formalized analogical process involves breaking the analysis into five distinct parts (Vicinanza, et. al., 1991). The first step requires the acquisition of knowledge and the appropriate representation of same. The second step requires the selection of candidate analogous cases for examination. The third step is known as source-target mapping. During this process both the similarities and differences between the source and target are carefully cataloged. Fourth, the solution is transferred based on the chosen source. Finally, the solution is adjusted based on the dissimilarities between the source and target as noted in the third step.

Vicinanza, et. al. (1991) also identified three distinct classes of knowledge that are necessary to develop effective analogical estimates. Case knowledge, which represents the estimator's episodic memory of previously encountered development projects, is the first of these. Second, case selection knowledge is the type of knowledge which allows the estimator to choose the appropriate source. Finally adjustment knowledge allows the estimator to make the proper adjustments to the estimate as noted in step five. These three knowledge types point out one of the great limitations of analogical estimation, namely, it has to be done by a highly experienced developer for it to have any validity. These people are often difficult to locate. Even if they are found, they need to have experience in the same type of environment as the target project.

Also, analogical estimation compares a present day project to something that has happened in the past and the development environment can change dramatically in a short time. Analogies look only at the end product, not the outlying factors involved. Advances in programming such as better CASE tools and increases in component usage can significantly shorten development time (Sparling, 2000). This could lead to a significant overestimation of development time.

Expert Estimation - Work Breakdown Structures

There are two major types of work breakdown models for software development estimation: bottom-up and top-down. The bottom-up method involves the decomposition of a project into smaller constituent parts and then estimating each of those parts separately (Jørgensen, 2004). Each of these smaller parts may be estimated in any of the ways mentioned previously, but the usual practice is to create parts small

enough to be estimated easily by an expert. The individual estimates generated are then combined to create one overall estimate for the entire project.

The bottom-up process is essentially a two-step process involving first decomposition and then integration or reconstitution. One of the advantages to this approach is that it may make explicit many system-level tasks that are often ignored in other methods (NASA, 2002). These tasks include integration, documentation, project control, and configuration management. Another advantage to this approach is that it can be used in conjunction with other engineering tasks, especially those that are hardware-related (Briand, 1998). Because of the flexibility allowed by this method, individual elements can be estimated independently. If, for instance, a portion of the project is closely related to another project recently developed, it can be estimated by analogy while other more problematic portions of the project can be estimated in some other fashion.

This process, however, is very resource-intensive and demands a very detailed specification of requirements (Pfleeger, et. al., 2005). Because requirements often evolve throughout the development process quite a bit of this effort may be wasted. Also, by isolating individual portions of the project, those doing the estimates lose sight of how their individual portions relate to the project as a whole. This can cause estimators to ignore considerations that might be apparent if the project were to be estimated as a whole.

Closely related to the bottom-up technique is the top-down estimation method. This technique also involves the decomposition of the project into its constituent parts, but the effort estimation for each part is generated based on overall project traits, instead

of detailed functional characteristics (Jørgensen, 2004). Under this method the individual portions of the project are generally estimated expertly or by analogy and are therefore subject to the limitations inherent in those methods (Briand, 1998). The top-down method benefits from the same advantages as the bottom-up method as it also involves decomposition and integration, but because the estimates are done at a very high level, the lack of detail makes the estimate nigh on impossible to document or verify effectively (NASA, 2002).

Simulation

Simulation involves the use of software that performs sophisticated statistical simulations that predict the scope and outcome of the work to be performed. Estimation software of this nature usually accounts for a number of different sources of variability, including variations in productivity, program size, and rates of staff expertise and turnover (McConnell, 2006). The software will generate a probability matrix that allows the estimator to determine to a fairly high level of certainty whether the project will meet goals for cost and schedule. Because this technique generates only a set of probabilities, it is almost always used to make a go/no-go decision and not for detailed estimations.

Effectiveness of the Various Techniques

We have seen that there are a number of varied techniques for estimating the resources necessary for completing a software development project. Each has its own limitations. With the exception of function point counting, each relies to a large extent on a certain amount of guesswork. The general rule of thumb seems to be that, the more

detailed the specifications, the easier it is to forecast the necessary resources. While intuition would suggest that the more scientific methods, like function points, would produce better results, the evidence suggests that the most intuitive processes generate, if not the best outcomes, certainly results that are equal to other more scientific methods.

There have been a multitude of studies performed to determine whether each of these techniques is effective. Many of these studies have compared estimation techniques to one another to in an attempt to determine whether one yields more accurate results. The results of these studies have often conflicted with one another. For instance, in 1997 Jørgensen found that function point estimates were more accurate than expert estimates based on a review of 47 industrial projects. Heemstra and Kusters, however, found in 1991 in a survey of 597 Dutch companies that expert estimation yielded far better results. Similar studies undertaken more recently produced similar conflicting results.

Jørgensen and Sjøberg, for instance found in 2002 that regression (algorithmic) techniques produced better results than expert estimations while Kitchenham et. al. found in the same year that there was no difference in results generated by experts and algorithms.

Lederer and Prasad (2000) surveyed 112 different software organizations and found that the algorithmic development methods did not lead to higher accuracy compared with “intuition, guessing, and personal memory.” Atkinson and Shepperd (1994) studied 21 different software projects and compared expert estimates to a number of different techniques including analogy and function points. The expert estimates were not as accurate as the analogy-based estimate, but were more accurate than the estimates

based on function points. Finally, Pengelly (1995) conducted an in-depth study of a single development project and found that the expert estimates were more accurate than all other types of estimation models, including COCOMO, function points, and SLIM.

Taken individually, none of these studies presents compelling evidence to suggest that expert estimation is the best estimation method, but as a body of evidence the trend seems to be obvious. Expert estimation provides projections that are, on average, at least equivalent to, if not better than, the estimates provided by algorithmic models and analogy. In large part this is due to the fact that most of the model-based estimation techniques are based on data from past projects; they simply model formally the things expert estimators are ostensibly doing (Laird, 2006). When we speak of *ex ante* estimates we know that at some point a human has made a guess as to some quantity or other on which a model-based estimation must rely.

Still, as a technique, expert estimation remains problematic. If an organization does not have access to experts, it can not use the technique. Further, expert estimation is not independent of project type. Someone who is proficient at estimating development time in one environment might be completely at a loss in another. Jørgensen (2004) compared expert estimation in top-down and bottom-up cases and found that different techniques were necessary to generate an accurate projection. For instance, estimators following the top-down strategy may be able to provide good estimates at low cost even without any sort of technical expertise. To do so, they need only apply high level knowledge gained from the completion of other projects. To complete a bottom-up estimate, however, an estimator must have both a great deal of expertise and copious

amounts of time. Obviously, an expert estimator well suited for one task may not be competent for the other.

Unfortunately, because so much of expert estimation involves intuition, there is not seem a standard set of best practices that tells a developer how to best go about coming up with an estimate. Experience in a particular development environment seems to be the best predictor of success. For instance, Jørgensen also found that “the applicability of the bottom-up estimation strategy is restricted to situations where there are estimators with sufficient knowledge about how to construct the software (2004).” If there exists a best practice for expert estimation, it would seem to be to select the best expert.

It is unreasonable to assume that the results of all of these studies are flawed. At the same time, however, there must be some reasonable explanation as to why they have produced results that do not allow us to draw any firm conclusions. The most obvious explanation is that there is something in the projects themselves that makes them more amenable to certain types of estimation techniques. Therefore, the next step will be to determine, within the limited scope of this study, whether this is so. It is quite likely that certain estimation techniques will work better on certain types of development projects. The rest of this paper will concern itself with how this effort will be undertaken.

CHAPTER 3

RESEARCH MODEL

Estimations are done at many different times during the development life cycle. For the purposes of this research, however, we will limit ourselves only to *ex ante* estimates of time, effort, and expenditures. Because of this decision, it is necessary to make certain assumptions of the projects estimated as software development and maintenance projects that have not begun share certain characteristics. First, they will, in all likelihood, have had requirements that are incomplete because requirements evolve during the development process (Conte, et. al., 1986). This, in turn, will have required that software entities, characteristics, and relationships be translated into their likely – not definite – size attributes.

The second of these assumptions is that the sizing models used matched the actual conditions under which the project was undertaken. The third is that the estimation model used was used properly and in the prescribed manner. Finally, it is to be assumed that the estimation technique used was tailored to the needs of the organization performing the estimate. These final three assumptions are made because to do otherwise would be illogical and impractical. Each of these assumptions represents a standard best

practice and is made because the results and conclusions of this study will be generalizable only if the estimation technique is being used properly.

The next step is to determine how, exactly, estimation effectiveness is to be measured. There are essentially three considerations that are of concern to those considering the development of anything, software or otherwise. The first is how much money it will cost, the second is how long will it take, and finally, how much effort it will require. Only by knowing all three of these things can managers make effective decisions about the pursuit of individual projects. Hence, this study will concern itself with initial estimates of person hours required and the projected duration of the project. The amount of money a project costs will only be considered as a measure of the size of the project as decision makers can derive monetary costs from measures of effort and duration. These estimates will be compared to the actual amounts of each used upon completion of the project. Since it is necessary to compare estimates to actuals, only completed projects will be considered as unfinished projects do not yet have “actual” values.

There are a number of other factors that may have a bearing on the effectiveness of the estimation technique being used. Generally speaking, software projects can be broken down into two types, development and maintenance. Development projects involve the creation of an entirely new piece of software, one whose requirements, functionality, and size all need to be specified. Maintenance projects involve making adjustments to an existing piece of software. Maintenance projects are of many types and primarily include: preventive maintenance, designed to anticipate problems in systems and correct them before they manifest; perfective, designed to increase the performance

of functioning systems; and corrective, designed to correct problems in systems that generate bad output.

Since this research will be concerned with expert estimation, it will also be important to know some things about the individuals creating the estimates. Expert estimation may be predicated upon the availability of experts, but organizations will still quite often generate “expert” estimates even if there is no expert available. Therefore, there should be a measure of estimator experience not only as an estimator, but as an estimator on a given system as well.

The research model, then, will include measures of project effort, project duration, estimation method, estimator experience, project type (development or maintenance), and project complexity. The data gathered will then undergo statistical analysis to determine whether a relationship exists between or among these data and the ability of expert estimation to provide accurate assessments of the amount of time and effort required to complete a software project.

The first hypothesis to be tested will look for a link between project complexity and estimation accuracy; in this case, specifically effort estimations generated using expert estimation. Complexity was chosen as a dependent variable for each of the first three hypotheses because intuition would suggest that highly complex projects would be much more difficult to estimate accurately than simpler ones. Each of the first three hypotheses was framed so that if an association exists between complexity and estimation accuracy, either positive or negative, acceptance or rejection of the hypothesis would yield worthwhile results.

H(1): Project complexity is associated positively with the accuracy of project effort expert estimation.

The second hypothesis is essentially the same as the first; however this one will test for accuracy of duration estimations.

H(2): Project complexity is associated positively with the accuracy of project duration expert estimation.

Since duration and effort estimations are two elements of a total estimate, we will also test to see if complexity is associated with overall project estimation accuracy.

H(3): Project complexity is associated positively with the overall accuracy of project expert estimation.

The notion that experienced estimators are better able to provide accurate estimates will also be tested. If expert estimation is to have any inherent value, then it would seem logical to test the notion that more estimation experience would lead to better estimates.

H(4): Estimator experience is associated positively with overall project expert estimation accuracy.

Because development projects and maintenance projects can differ substantially it will be instructive to explore whether one or the other is more apt to be estimated accurately. First we will examine project type and effort estimation. As with the first four hypotheses, these last two hypotheses were framed in such a way so that rejection would also yield worthwhile results.

H(5): Project type (development or maintenance) is associated positively with accuracy of project effort expert estimation.

Finally, we will examine whether the same holds true for duration estimation.

H(6): Project type is associated positively with accuracy of project duration expert estimation.

Data

We used three different data sets to perform the analysis for this study. The first data set was provided by Kitchenham, Pfleeger, McColl, and Eagan (Kitchenham, et. al., 2002). It will be referred to as “Kitchenham data.” The second data set was provided by Jørgensen and Sjöberg (2002). The authors provided data that were not included with the published paper (Jørgensen and Sjöberg, 2002). This data set will be referred to as “Jørgensen data.” The final data set was provided by a health maintenance organization located in Las Vegas, Nevada. It will be referred to as “HMO data.”

Kitchenham Data

The Kitchenham data set (Kitchenham, et. al., 2002), available in the referenced paper, was generated from observations of 145 maintenance and development projects managed by a single outsourcing company. The company’s standard estimation practice is to estimate each project’s effort and duration using at least two different estimation techniques. The techniques included expert estimation, CA-Estimacs (a commercial software package), Delphi (a group decision based on the averages of multiple estimates arrived at independently), widget counting, and averaging (the average of two or more estimates arrived at by the same estimator(s)). The two (or more) resultant estimates

would then be presented to a client who would choose the favored one. Only the final choice was made available to Kitchenham.

The data to be analyzed as part of this study include:

- Project type (development or maintenance) (dichotomous)
- Actual start and completion dates as a measure of duration (continuous)
- Estimated and actual effort (in person-hours) as a measure of effort (continuous)
- Adjusted function points as a measure of complexity (continuous)
- Estimation method (dichotomous)

The estimation technique variable was broken down into two distinct categories; expert estimation and all others. This was done because the focus of this research is expert estimation and the other techniques matter only insofar as they are not expert estimation. Further, while the Kitchenham data set differentiated among all the different maintenance types (preventive, corrective, etc.), maintenance projects are essentially all the same insofar as they involve the adjustment of existing systems. Estimating the amount of time and effort required to make these adjustments will be a similar process regardless of the underlying reason for the changes.

The fact that this data set included estimates and actuals of both effort and duration made it possible to assess estimation accuracy for both characteristics. This differentiates the Kitchenham data from the Jørgensen in that the Jørgensen data had essentially a dichotomous accurate/inaccurate measurement of accuracy

Jørgensen Data

The Jørgensen data were gathered in an empirical study of 54 software maintainers in the software maintenance department of a Norwegian company in 2001. The data set referred to maintenance projects only. The data included measures of other essential elements: project effort, project duration, estimation method (all projects utilized expert estimation), and estimator experience.

The data to be analyzed as part of this study include:

- the number of years of experience the estimator had both as an estimator and on the particular application being maintained (continuous)
- the estimator's assessment of the complexity of the project (rated low, medium, or high) (categorical)
- the occurrence of unexpected problems (dichotomous)
- complexity as measured by the total number of lines of code added, deleted, or changed (continuous)
- the size of the application being maintained (continuous)
- a measure of the accuracy of the estimate (categorical)
- the amount of effort required to complete the project measured in person-days (continuous)

The measure of estimation accuracy was derived from the measure of estimator confidence and whether unexpected problems arose during the maintenance effort. Jørgensen rated as "too optimistic" any estimate in which the estimator had confidence, but then went on to experience unexpected problems. These optimistic estimates are considered to be underestimates for the purposes of this research. If an estimator was not

confident but did not encounter unexpected problems, the estimate is labeled “too pessimistic,” or overestimated. If an estimator was not confident and encountered problems or was confident and did not encounter problems, the estimate is considered accurate. All other data in the set were measured directly.

It also bears mentioning that Jørgensen originally asked his respondents to rate their confidence on a “yes,” “maybe,” or “no” scale. Respondents were told to answer “yes” only if they felt there was a very low risk of unexpected problems. Upon examination of the interview results Jørgensen determined there was only a very small difference between the “maybe” and “no” confidence levels. He therefore used only two confidence classes as his confidence measure; Y (“yes”) and N (“maybe” and “no”). For the purposes of this study the same procedure was followed.

HMO Data

The HMO data set consists of 18 distinct observations of software development and maintenance projects. This data set contains measures of size (in budgeted and actual dollars) and duration (in days). All the estimates were generated by expert estimation. The projects themselves involved both maintenance and development. The data will be examined for an indication of project duration estimation accuracy.

CHAPTER 4

ANALYSIS AND RESULTS

Hypotheses one, two, five, and six concern themselves with measures of effort and duration estimation accuracy and could, consequently, be explored effectively by delving into the Kitchenham data set. Therefore the first step was to explore the basic ways in which that data set examines project characteristics relative to effort and duration estimation accuracy. A simple comparison of means (t-test) was the best way to begin. Therefore, the Kitchenham data set was sorted by each of the five independent variables and broken into categories. Each of the continuous variables was broken into high, medium, and low ranges and the dichotomous variables were broken into two distinct sets; expert and others for estimation technique, and development and maintenance for project type. The mean for each category was calculated and the mean underestimation for each category as well. The results appear in Table 2.

Table 2

Mean Actual Project Effort and Duration and Mean Underestimation

By Project Characteristic

	Effort¹			Duration²		
	n	Mean effort	Mean Underestimate	n	Mean effort	Mean Underestimate
Complexity³						
Low	48	964.60	-16.78%	47	164.06	0.71%
Medium	48	1750.67	-6.09%	46	170.91	0.86%
High	48	4315.38	9.52%	48	270.27	21.67%
Difference in means not significant				Difference in means significant at $p < 0.05$ for L - M and L - H		
Size⁴						
Low	48	623.06	-18.93%	47	144.30	-0.04%
Medium	48	1546.12	-12.74%	46	181.11	8.04%
High	48	4861.46	9.91%	48	279.85	15.62%
Difference in means not significant				Difference in means significant at $p < 0.05$ for L - H		
Project Type						
Development	51	2802.84	1.34%	49	199.37	4.46%
Maintenance	93	2091.68	0.69%	92	204.10	12.02%
Difference in means not significant				Difference in means not significant		
Duration⁵						
Low	50	1064.86	-13.83%	49	96.98	-7.62%
Medium	46	2163.96	-4.66%	44	172.14	7.68%
High	48	3847.62	9.90%	48	337.92	16.39%
Difference in means not significant				Difference in means significant at $p < 0.05$ for L - M and L - H		
Estimation Technique						
Expert Estimation	104	2334.23	0.98	101	211.25	7.93%
Other	40	2367.78	0.92	40	180.25	13.61%
Difference in means significant at $p < 0.1$				Difference in means not significant		

1: Measured in person-hours

2: Measured in days

3: Low = 0 - 166 Adjusted Function Points, Medium = 167 - 403 AFP, High = 404 - 2076 AFP

4: Measured in person-hours of effort: Low < 1,000 hrs, M = 1,000 - 2,200 hrs, H > 2,200 hrs

5: Measured in days: L = 0 - 133, M = 134 - 205,
H > 205

The low and medium order projects all overestimated the amount of effort required and estimated fairly accurately the duration required. However, the high order projects dramatically underestimated both the effort and duration required for completion. These observations were consistent throughout the continuous data elements.

A test for difference in means revealed that, for the most part, the differences in means were not statistically significant when looking at effort estimation. The tests for differences in means of estimated duration showed that the mean differences were statistically significantly different from zero.

Next, we endeavored to determine if the five independent variables interacted in such a way as to have a demonstrable impact on effort estimation accuracy. The way to accomplish this was to perform a linear regression analysis. The regression in this case is being used to analyze two distinct dependent variables; effort estimation accuracy and duration estimation accuracy. The dependent variables, however, are the same for both. Therefore the regression equation is:

$$\begin{aligned}
 Y_1, Y_2 = & \beta_0 + \beta_1 E_i + \beta_2 D_i + \beta_3 C_i + \beta_4 T_i + \beta_5 M_i + \\
 & \beta_6 E_i D_i + \beta_7 E_i C_i + \beta_8 E_i T_i + \beta_9 E_i M_i + \beta_{10} D_i C_i + \beta_{11} D_i T_i + \beta_{12} D_i M_i + \beta_{13} C_i T_i + \\
 & \beta_{14} C_i M_i + \beta_{15} T_i M_i \\
 & \beta_{16} E_i D_i C_i + \beta_{17} E_i D_i T_i + \beta_{18} E_i D_i M_i + \beta_{19} E_i C_i T_i + \beta_{20} E_i C_i M_i + \beta_{21} E_i T_i M_i + \\
 & \beta_{22} D_i C_i T_i + \beta_{23} D_i C_i M_i + \beta_{24} D_i T_i M_i + \beta_{25} C_i T_i M_i + \\
 & \beta_{26} E_i D_i C_i T_i + \beta_{27} E_i D_i C_i M_i + \beta_{28} E_i D_i T_i M_i + \beta_{29} E_i C_i T_i M_i + \beta_{30} D_i C_i T_i M_i + \varepsilon
 \end{aligned}$$

where:

$$Y_1 = X_i - E_i / E_i$$

$$Y_2 = Z_i - D_i / D_i$$

X_i = project i was estimated to require X_i hours to complete

Z_i = project i was estimated to require Z_i hours to complete

E_i = project i required E_i hours to complete

D_i = project i required D_i days to complete

C_i = project i contained C_i adjusted function points

T_i = project i was of type T_i (1 = development, 2 = maintenance)

M_i = project i was of type M_i (1 = expert estimation, 0 = other)

The regression equation included 30 different combinations of independent variables. The saturated model yielded the following estimated coefficients:

Table 3

Estimated Coefficients for Saturation Model (1),
Dependent Variable Effort Estimation Accuracy

	β	t	Significance
B₀	0.580		
Single Variables:			
E	-0.869	-0.392	0.696
C	1.548	0.817	0.416
D	-0.036	-0.060	0.952
T	-1.761	-1.544	0.125
M	-0.054	-0.206	0.837
Two-way interactions			
C * E	-3.031	-0.475	0.636
C * D	-2.452	-0.747	0.457
C * M	-1.669	-0.848	0.398
C * T	0.031	0.016	0.987
E * D	0.044	0.014	0.989
E * M	-0.928	-0.405	0.686
E * T	3.177	1.057	0.293
D * M	-0.305	-0.453	0.652
D * T	2.536	1.214	0.227
M * T	1.497	1.384	0.169
Three-way interactions			
C * E * D	5.638	0.700	0.485
C * E * M	4.704	0.790	0.431
C * E * T	0.235	0.058	0.954
C * D * M	2.533	0.803	0.424
C * D * T	-0.874	-0.290	0.772
C * M * T	-0.425	-0.234	0.816
E * D * M	2.552	0.803	0.424
E * D * T	-6.976	-1.132	0.260
E * M * T	-1.165	-0.412	0.681
D * M * T	-2.447	-1.253	0.213
Four-way interactions			
C * E * D * M	-7.956	-1.065	0.289
C * E * D * T	2.262	0.660	0.510
C * E * T * M	-2.055	-0.600	0.549
C * D * T * M	1.618	0.577	0.565
E * D * T * M	4.207	0.750	0.455
R square = 0.118, F = 0.499			

The saturated model did not yield any statistically significant estimated coefficients. The regression analysis therefore continued in stepwise fashion eliminating the least significant variable at each step. After 22 regressions, the model appeared thus:

Table 4
Estimated Coefficients for Model (1) – 22nd Regression
Dependent Variable Effort Estimation Accuracy

	β	t	Significance
β_0	0.384		
Two-way interactions			
E * M	-0.908	-1.540	0.126
D * M	-0.210	-1.522	0.130
Three-way interactions			
C * E * D	0.176	0.618	0.537
C * E * M	0.768	0.940	0.349
C * M * T	-0.263	-0.951	0.344
E * D * M	1.387	1.580	0.116
E * D * T	-0.350	-1.330	0.186
Four-way interactions			
C * E * D * M	-1.263	-1.179	0.241
C * D * M * T	0.523	1.271	0.206
R square = 0.062, F = 0.985			

The 22nd regression did not yield any statistically significant estimated coefficients, either. All the significance factors were still well above the acceptance threshold of 0.1. Therefore the regression had to continue.

The regression continued through 29 steps and finally yielded the following equation with its associated table:

$$Y_1 = \beta_0 + \beta_9 E_i M_i + \beta_{18} E_i M_i D_i$$

Table 5

Estimated Coefficients for Model (1) – 29th Regression

Dependent Variable Effort Estimation Accuracy

	β	t	Significance
β_0	0.299		
E * M	-0.423	-2.071	0.040
E * M * D	0.381	1.869	0.064
R square = 0.30, F = 2.147			

With this final regression, the interaction of size and method was found to be significant as well as the interaction among size, method, and duration. Please note that the measures of size and duration as well as the estimation method are not significant in and of themselves, only as they interact with one another. Therefore, we see that expert-estimated large projects generally relate to underestimation, the opposite holds true for the same project with increased duration.

The regression of the effort estimation data revealed that, all other things being equal, a large project being estimated using expert estimation demonstrated a negative association with effort estimation accuracy. However, when duration was added to this same mix the association became a positive one. This change in the sign from negative to positive means that as projects take longer to complete the ability of expert estimators to predict the effort required increases, assuming that the projects are sufficiently large.

The next step in this process was to test this same data set for duration accuracy. Following the same pattern as with effort estimation, a backwards linear regression was run with the same five independent variables, but the dependent variable in this case was duration estimation accuracy (Y_2).

Again, the regression included 30 different combinations of independent variables and the saturated model appears below. This saturation model did not yield any statistically significant estimated coefficients. The results of this model estimation are shown in Table 6.

Table 6

Estimated Coefficients for Saturation Model (2)

Dependent Variable Duration Estimation Accuracy

	β	t	Significance
B₀	0.580		
Single Variables:			
E	-0.432	-0.195	0.846
C	-0.632	-0.334	0.739
D	-0.618	-1.026	0.307
T	0.184	0.164	0.870
M	-0.188	-0.711	0.479
Two-way interactions	1.802	0.282	0.778
C * E	0.849	0.259	0.796
C * D	0.543	0.276	0.783
C * M	1.083	0.558	0.578
C * T	1.052	0.332	0.741
E * D	-0.869	-0.378	0.706
E * M	-0.152	-0.050	0.960
E * T	0.486	0.720	0.473
D * M	-2.110	-1.017	0.311
D * T	-0.142	-0.133	0.894
M * T	1.802	0.282	0.778
Three-way interactions			
C * E * D	-2.513	-0.312	0.756
C * E * M	-0.102	-0.017	0.986
C * E * T	-8.439	-2.076	0.040
C * D * M	-0.758	-0.240	0.811
C * D * T	1.438	0.477	0.634
C * M * T	-1.032	-0.566	0.572
E * D * M	0.469	0.147	0.883
E * D * T	5.842	0.947	0.346
E * M * T	1.209	0.427	0.670
D * M * T	1.967	1.017	0.311
Four-way interactions			
C * E * D * M	0.595	0.080	0.937
C * E * D * T	1.501	0.436	0.664
C * E * T * M	6.862	2.002	0.048
C * D * T * M	-1.298	-0.462	0.645
E * D * T * M	-6.740	-1.200	0.233
R square = 0.139, F = 0.588			

This data set underwent 22 regressions before yielding the following equation and the attendant Table 7.

$$Y_2 = \beta_0 + \beta_2 D_i + \beta_{11} D_i T_i + \beta_{19} C_i E_i T_i + \beta_{22} C_i D_i T_i + \\ \beta_{17} E_i D_i T_i + \beta_{24} D_i M_i T_i + \beta_{29} C_i E_i M_i T_i + \\ \beta_{30} C_i D_i M_i T_i + \beta_{28} E_i D_i M_i T_i$$

Table 7

Estimated Coefficients for Model (2) – 22nd Regression

Dependent Variable Duration Estimation Accuracy

	β	t	Significance
B₀	.021		
Single variables			
D	-0.171	-1.891	0.061
Two-way interactions			
D * T	-1.217	-2.588	0.011
Three-way interactions			
C * E * T	-5.512	-3.257	0.001
C * D * T	1.850	2.683	0.008
E * D * T	4.019	2.742	0.007
D * M * T	1.280	2.793	0.006
Four-way interactions			
C * E * M * T	5.372	3.272	0.001
C * D * M * T	-1.842	-2.607	0.010
E * D * M * T	-3.805	-2.693	0.008
R square = 0.108, F = 1.748			

The results suggest that project duration has a small negative impact on duration estimation accuracy. That is, for higher values of actual project duration, accuracy is worse. The two-way interaction suggests that this problem is exacerbated if the project is a development project.

Of the seven three- and four-way interactions four include expert estimation (which is part of all the four-way interactions), complexity, duration, and size, and all seven involve development projects. Since the development variable is dichotomous, a relationship can also be drawn with maintenance projects simply by switching “maintenance” for “development” and changing the β value for the interaction from positive to negative, or vice versa.

By looking at the β value, it can also be seen that a large, complex development project is very strongly correlated with an inaccurate estimation of duration, but if an expert estimator is added to the same interaction, the β value is equally strong in exactly the opposite direction. Interestingly, the exact opposite is the case if the same large development project is not complex, but takes a long time. A large long-term development project demonstrates a strong relation to an accurate duration estimate, but if an expert estimator is added to the mix (rather than using some other estimation technique), the likelihood of an accurate duration estimate is very low. The exact opposite will hold true if the project is a maintenance project rather than a development project.

To test the third and fourth hypotheses the Jørgensen data set was used because the dependent variable (overall estimation accuracy) is to be found there. Estimation accuracy, as measured by Jørgensen, consists of overestimations (which Jørgensen terms “too pessimistic”), underestimations (which Jørgensen terms “too optimistic”) and accurate estimations. Because the dependent variable is categorical and not continuous, logistic regression was used. The univariate logistic regression requires a dichotomous dependent variable, but in this case the dependent variable has three categories.

Therefore three different sets of dichotomous variables were created. The first consisted of overestimated and non-overestimated projects, the second consisted of underestimated and non-underestimated projects, and the third consisted of accurate and non-accurate estimations. It is important to note that the regression equation calculates the log of a probability. Unlike a linear regression, logistic regression calculates the likelihood of a certain event taking place.

The regression equation for all three was the same and took the form of:

$$\text{Log} (P_{\theta} / 1 - P_{\theta}) = \beta_1 U_i + \beta_2 L_i + \beta_3 M_i + \beta_4 H_i + \beta_5 A_i + \beta_6 B_i + \beta_7 C_i + \beta_8 S_i + \beta_9 E_i$$

where:

P = probability

Θ = underestimated, overestimated or accurate where:

if estimator confidence = Y and unexpected problems = Y

then project was underestimated

else if confidence = N and unexpected problems = N

then project was overestimated

else project was estimated accurately

U_i = project i experienced unexpected problems (1 = yes, 0 = no)

L_i = project i had a low order of complexity (1 = yes, 0 = no)

M_i = project i had a moderate level of complexity (1 = yes, 0 = no)

H_i = project i had a high order of complexity (1 = yes, 0 = no)

A_i = project i had a low number of LOC changed (1 = yes, 0 = no)

B_i = project i had a moderate number of LOC changed (1 = yes, 0 = no)

C_i = project i had a high number of LOC changed (1 = yes, 0 = no)

S_i = project i contained S lines of code (including comments)

E_i = the estimator of project i had E years of experience

It is important to note that the variables L , M , and H are dichotomous and only one of the three may have a value of one at any one time. The same holds true for the variables A , B , and C .

The first logistic regression generated the estimated saturation model shown in Table 8.

Table 8
Estimated Probability Significance for Saturation Model (3)
Dependent Variable Development Underestimation

Variable	β	Significance
U	22.371	0.996
L		0.026
M	-3.906	0.008
H	-1.962	0.142
A		0.534
B	1.218	0.310
C	1.368	0.352
S	0.0	1.000
E	0.109	0.856

From here we conducted a backwards stepwise regression, eliminating at each step the independent variable with the least significance from the model (assuming that the variable was not significant at the 0.1 level). Estimation of the final regression yielded the results shown in Table 9.

Table 9

Probability Significance for Reduced Model (3)

Dependent Variable Development Underestimation

Variable	β	Significance
L		0.005
M	-1.326	0.209
H	-1.917	0.002

The regression showed that both low and high complexity projects were consistently underestimated. The saturation model correctly predicted 77.1% of the actual outcomes while the final regression accurately predicted 92.6% of the actual outcomes.

It should also be noted that Jørgensen data set was tested to include both measures of estimator experience, both as single independent variables, and by including the additional measure of estimator experience (experience on the particular application being maintained) in the saturation model. In no case did estimator experience show a significant association with estimation accuracy.

The HMO data set was analyzed to see if it could shed any light on hypothesis number two. Therefore, the HMO set was broken down into two subsets based on budgeted dollars and duration estimation accuracy was examined. The results are shown in Table 10.

Table 10

Comparison of Means for HMO Data Set

	n	Mean Estimated Budget	Mean Actual Budget	% Difference	Mean Estimated Duration	Mean Actual Duration	% Difference
Low Budget	9	\$200,963	\$205,230	-2.08%	250.5	249.3	0.48%
High Budget	9	\$1,637,939	\$1,514,190	8.17%	480.6	587.1	18.14%
Difference in means not significant							

The higher budget projects were overestimated to a fairly large degree whereas the lower budget projects were fairly accurately estimated.

CHAPTER 5

DISCUSSION

The discussion will first focus on the hypotheses to see which are supported or if their antitheses are supported. Following that, the discussion will turn to those items suggested by the data analysis but not hypothesized. The first hypothesis attempted to make a connection between effort estimation accuracy and project complexity. The comparison of means test found no significance, nor did the linear regression analysis of the Kitchenham data set find any connection between project complexity and effort estimation accuracy, either positive or negative. The first hypothesis, therefore, must be rejected along with its antithesis.

The second hypothesis poses the same question, but with respect to duration estimation accuracy. The comparison of means test does demonstrate an association of complexity with underestimation, and the association appears fairly strong. In looking at the regression results from the Kitchenham data set, however, a few questions are raised,

When complexity appears as a factor it appears only in conjunction with project type. Because project type is a dichotomous variable, however, if the project type is changed, so is its attendant β value. It is therefore impossible to draw a conclusion about

complexity, expert estimation, and duration estimation accuracy without taking project type into account.

Table 11
Analysis of Support for Hypotheses

Hypothesis	Supported?	Antithesis Supported?	Comments
H (1)	No	No	<ul style="list-style-type: none"> • difference in means not significant • no statistically significant estimated coefficient associates effort estimation accuracy with complexity
H (2)	No	Qualified Yes	<ul style="list-style-type: none"> • difference in means significant • large, complex development projects strongly associated with underestimation
H (3)	No	Yes	<ul style="list-style-type: none"> • both low and high complexity projects strongly associated with underestimation • moderately complex projects not associated with underestimation
H (4)	No	Yes	<ul style="list-style-type: none"> • experience as estimator not associated with estimation accuracy • experience on application not associated with estimation accuracy
H (5)	No	No	<ul style="list-style-type: none"> • difference in means not significant • no statistically significant estimated coefficient associates effort estimation accuracy with project type
H (6)	No	No	<ul style="list-style-type: none"> • difference in means not significant • project type associated with both accuracy and underestimation

Large, complex development projects, for instance, are very strongly associated with inaccurate duration estimates. The same project using expert estimation, however, is just as strongly associated with an accurate estimate. This would indicate that expert

estimation is associated with accurate estimates. But that is only the case with development projects. Maintenance projects of this type are just as strongly associated with inaccurate expert estimates. Therefore, we can only support the antithesis of H(2) insofar as the difference of means test indicates that more complex projects are likely to be underestimated.

In testing the third hypothesis it is to be supposed that, since the Kitchenham data will not support the notion that complexity correlates with either effort estimation or duration estimation accuracy, it will not correlate with overall estimation accuracy, either. What we find, however, is that complexity is an indicator of inaccuracy in both low and high complexity projects, but not in moderate complexity projects. Therefore the third hypothesis must be rejected but its antithesis is supported. The Jørgensen data demonstrate that as project complexity moves toward either extreme, estimation accuracy suffers.

An examination of the data surrounding H(4) produces a very counterintuitive result, but nowhere does the Jørgensen data suggest that estimator experience has any kind of effect, positive or negative, on estimation accuracy. Again, the Jørgensen data only investigate maintenance projects. There is no conclusion drawn about the effect of estimator experience on the estimation of development projects. Still, H(4) and its antithesis must be rejected.

Likewise, the fifth hypothesis can be rejected as the Kitchenham data show no correlation between project type and effort estimation accuracy. There is, however, a correlation between project type and duration estimation accuracy as posited in H(6), but the correlation needs to be examined closely.

The only single variable to be correlated with duration estimation accuracy is duration itself, and its attendant β value is negative. This means that the longer a project takes to be completed the less likely any duration estimate is to be accurate, regardless of how it is arrived at; and although the correlation is significant, the β value is not very strong. While the β value increases if a high duration project is a development project, the results still do not draw any conclusions about expert estimation at this point. That same high duration development project, however, becomes positively associated with duration estimation accuracy if it is estimated using expert estimation. That would seem to indicate that expert estimation leads to better duration estimates.

Looking at the three- and four-way interactions, we can see that project type is certainly an indicator of the accuracy of duration estimation, but it is how the other independent variables interact that is of greater interest. For instance, an expertly estimated development project of substantial duration shows a positive correlation with estimation accuracy, but as that project gets larger it is highly likely that the estimation will be inaccurate.

The situation becomes even more multifaceted when we remember that whatever is said of the interactions involving development projects, the converse is true of maintenance projects. Therefore H(6) and its converse must ultimately be rejected because there can be no definitive statement made regarding project type and duration estimation accuracy.

What, then, can be said about expert estimation's ability to estimate the amount of time and effort necessary to complete a software project? One thing we can say is that as maintenance projects tend toward the extreme in complexity expert estimation is unlikely

to provide a good estimate of effort or duration. The HMO data set provides corroboration. All projects were estimated fairly well as far as their budgets were concerned, but the larger projects were underestimated dramatically as far as duration went.

One other thing that can be said is that, development projects of substantial duration can be well estimated for duration via expert estimation as long as they do not combine more than one of the other independent variables with the duration. With an increase in size or complexity, expert estimation loses its ability to predict duration outcomes. But if this is so, why would an expert estimated development project of great size and high complexity show such a strong correlation with duration estimation accuracy? One has to think about a project that is large and complex but does not take a long time to complete. This describes a project of high importance. If we recall what was mentioned earlier, that oftentimes management will ask for an estimate when what they want is a goal, then the answer seems obvious. A highly important project that needs to be done within a certain time frame regardless of its size or complexity will be done within a certain time frame. If we note that this combination of factors applies only to duration estimation and not effort estimation the answer seems clear. This would also explain why projects of greater duration (and ostensibly lesser importance) would tend not to be estimated correctly.

CHAPTER 6

IMPLICATIONS, LIMITATIONS, AND SUGGESTIONS

Implications for Business

The implications of this research suggest some interesting guidelines for business. Since we have demonstrated that estimator experience is not an indication of estimation accuracy, it would tend to lay to rest the notion that expert estimation requires the presence of an expert. In any event, it may very well be necessary to change our idea of what constitutes an expert. Certainly businesses should think twice about utilizing expert estimation if they are interested in gaining a proper perspective on how much effort – and consequently how much money – a project will require, especially as projects grow in size and scope.

It is outside the purview of this research to make judgments about the efficacy of expert estimation compared to other estimation techniques. Certainly expert estimation is less costly than algorithmic estimation methods and therein may lie its value. Time and money lost because of a poor estimation may be made up for by the low cost of the estimation itself. This brings to mind Boehm's conundrum regarding the buying of information. It must be remembered that the estimates studied herein are *ex ante* and

therefore may very well suffer from incomplete requirements and all the other problems attendant to projects created *ex nihilo*.

Certainly it will be important for businesses to differentiate between development and maintenance projects, which have exactly the opposite effect on duration estimation for many types of projects, especially large, complex projects with their attendant high cost. Keeping two teams of estimators, one for each type of project, may very well increase accuracy significantly.

Limitations and Suggestions for Further Research

This research was limited, first and foremost, by the data sets it was required to use, each of which had its limitations. The Jørgensen data set was limited by the categorical nature of its dependent variable and the fact that this categorical variable was the only measure of estimation accuracy in the entire data set. It was further hampered by the fact that all the observations were only of maintenance projects. The Kitchenham data, on the other hand, was actually fairly rich, but less than 30% of the observations were of estimation techniques other than expert estimation, and some of those other techniques involved the use of expert estimation to a greater or lesser extent. This means that the information gleaned about expert estimation exists in a bit of a vacuum as there is no information about how expert estimation compares to other techniques.

This raises the possibility of further research. First of all, it would be helpful to have similar data regarding the abilities of other estimation techniques to provide accurate results based on project characteristics. It may or may not be possible to ever define a best practice for software development estimation, but if it can be said with some

certainty that a particular technique will work better in a given situation much will have been accomplished. Generating the same sort of data as contained in this research for algorithmic estimation techniques would be a good start.

It may also prove fruitful to pursue the notion that projects of greater importance and urgency achieve higher rates of estimation accuracy. It would certainly be of great interest to examine whether expert estimated development projects that have high complexity and size exceed their budgets regularly or require large inputs of effort. This research suggests that such a link may exist; it is certainly worthy of exploration.

APPENDIX

HMO Data Set

Project Number	Estimated Budget	Actual Budget	Estimated Duration	Actual Duration
1	\$80,000	\$71,600	116	134
2	\$131,700	\$96,548	217	234
3	\$209,964	\$160,200	109	368
4	\$218,400	\$216,128	159	73
5	\$225,100	\$217,248	159	83
6	\$185,112	\$229,295	991	933
7	\$134,750	\$267,207	75	88
8	\$225,100	\$278,150	119	119
9	\$398,540	\$311,812	310	212
10	\$361,684	\$354,370	218	338
11	\$395,700	\$371,322	1021	1091
12	\$510,000	\$373,622	1021	1096
13	\$626,702	\$607,023	233	199
14	\$972,900	\$776,849	240	255
15	\$972,900	\$882,406	240	335
16	\$1,889,191	\$1,792,777	170	270
17	\$4,006,465	\$3,801,088	464	834
18	\$5,005,912	\$4,668,160	718	866

Jørgensen Data Set

Identifier	Complexity	Experience Overall	Experience on Application	Total LOC Changed	Estimation Accuracy
1.00	M	7.00	6.00	250.00	A
2.00	H	7.00	6.00	250.00	P
3.00	L	7.00	6.00	2.00	A
4.00	L	4.00	3.50	4.00	P
5.00	M	4.00	3.50	550.00	O
6.00	M	3.00	2.00	50.00	P
7.00	M	3.00	2.00	16.00	P
8.00	M	3.00	2.00	20.00	A
9.00	L	17.00	2.00	15.00	A
10.00	L	10.00	3.00	7.00	P
11.00	L	10.00	3.00	6.00	P
12.00	L	7.00	5.00	600.00	P
13.00	L	7.00	5.00	10.00	A
14.00	M	22.00	22.00	250.00	A
15.00	L	22.00	22.00	75.00	P
16.00	H	8.00	.30	1000.00	A
17.00	M	8.00	.30	1200.00	A
18.00	L	18.00	3.00	1.00	A
19.00	M	18.00	3.00	1.00	A
20.00	M	2.50	2.50	200.00	A
21.00	L	2.50	2.50	3.00	A
22.00	L	9.00	.50	200.00	O
23.00	L	9.00	.50	300.00	A
24.00	H	9.00	.50	5.00	P
25.00	H	4.00	1.00	25.00	A
26.00	M	1.00	.50	700.00	A
27.00	M	2.00	.30	20.00	O
28.00	M	3.00	3.00	600.00	O
29.00	L	3.00	3.00	.00	A
30.00	M	25.00	3.50	2100.00	A
31.00	M	25.00	3.50	.00	P
32.00	M	10.00	10.00	29.00	O
33.00	L	10.00	10.00	11.00	A
34.00	L	10.00	10.00	310.00	A
35.00	L	8.00	5.00	500.00	P
36.00	M	8.00	5.00	36.00	O
37.00	L	5.00	4.50	1.00	A
38.00	M	5.00	4.50	5.00	A
39.00	L	5.00	4.50	1.00	A

40.00	M	2.50	2.00	10.00	P
41.00	H	2.50	2.00	6.00	A
42.00	M	3.00	3.00	40.00	P
43.00	L	3.00	3.00	100.00	A
44.00	M	6.00	.50	25.00	A
45.00	M	7.00	3.00	140.00	O
46.00	M	8.00	.30	435.00	O
47.00	M	8.00	.30	300.00	O
48.00	L	8.00	.30	6.00	O
49.00	L	4.00	.50	15.00	A
50.00	L	4.00	.50	525.00	A
51.00	M	4.00	.50	400.00	A
52.00	H	4.00	.50	400.00	A
53.00	H	7.00	4.50	11.00	P
54.00	M	7.00	4.50	5.00	A
55.00	M	6.00	6.00	900.00	A
56.00	M	3.00	3.00	1.00	A
57.00	M	15.00	4.00	5.00	A
58.00	L	5.00	4.50	10.00	A
59.00	H	5.00	4.50	50.00	A
60.00	L	9.00	6.00	15.00	A
61.00	L	13.00	.00	5.00	P
62.00	L	6.00	.50	15.00	A
63.00	L	7.00	7.00	30.00	A
64.00	M	7.00	7.00	1500.00	O
65.00	M	7.00	7.00	100.00	O
66.00	L	7.00	7.00	4.00	A
67.00	L	3.00	.20	170.00	P
68.00	M	3.00	.20	20.00	P
69.00	M	3.00	.20	10.00	P
70.00	H	6.00	.30	1600.00	O
71.00	L	6.00	.30	100.00	A
72.00	M	6.00	.30	50.00	A
73.00	L	6.00	.30	30.00	A
74.00	L	17.00	17.00	100.00	A
75.00	M	4.00	3.50	150.00	O
76.00	L	4.00	3.50	84.00	A
77.00	M	2.00	1.50	.00	O
78.00	L	9.00	7.00	30.00	A
79.00	L	9.00	7.00	20.00	A
80.00	L	10.00	1.50	4.00	A
81.00	L	10.00	1.50	3.00	A
82.00	L	15.00	2.00	5.00	P
83.00	L	15.00	2.00	1.00	A
84.00	M	15.00	2.00	5.00	O

85.00	L	3.00	3.00	1.00	A
86.00	L	16.00	1.00	350.00	A
87.00	L	13.00	4.00	200.00	A
88.00	L	13.00	4.00	400.00	A
89.00	L	13.00	.50	1.00	A
90.00	M	13.00	.50	3.00	P
91.00	M	10.00	.30	118.00	P
92.00	M	10.00	.30	50.00	O
93.00	L	3.00	.30	10.00	A
94.00	H	3.00	.30	.00	A
95.00	L	5.50	5.50	1.00	A
96.00	M	5.50	5.50	1600.00	P
97.00	L	5.00	4.50	5.00	A
98.00	L	4.00	4.00	3.00	A
99.00	M	4.00	3.50	900.00	P
100.00	L	4.00	3.50	75.00	O
101.00	H	4.00	.00	500.00	P
102.00	L	5.00	3.00	50.00	A
103.00	M	14.00	1.00	20.00	A
104.00	M	14.00	5.00	500.00	O
105.00	L	14.00	5.00	400.00	A
106.00	H	1.00	.60	100.00	A
107.00	H	5.00	2.50	3.00	O
108.00	L	5.00	2.50	1.00	P
109.00	M	5.00	2.50	250.00	A

REFERENCES

- Aguilar-Ruiz, Jesus S.; Ramos, Isabel; Riquelme, Jose C.; and Toro, Miguel. *An Evolutionary Approach to Estimating Software Development Projects*. Information and Software Technology 43 (2001) 875 – 882.
- Boehm, Barry. *Software Engineering Economics*. IEEE Transactions on Software Engineering, SE-10, 1, 4 – 21; 1984.
- Briand, Lionel; El Amam, Khaled; Surmann, Dagmar; Wieczorek, Isabella; and Maxwell, Katrina. *An Assessment and Comparison of Common Software Cost Modelling Techniques*. Kaiserslauten, Germany; Fraunhofer Center for Empirical Software Engineering, ISERN technical report, 98-27, 1998.
- Conte, S. D.; Dunsmore, H. E.; and Shen, V. Y. Software Engineering Metrics and Models. 1986. Benjamin/Cummings Publishing Company. Menlo Park, CA.
- Finnie, G. R.; Wittig, G. E.; and Desharnais, J-M. *A Comparison of Software Estimation Techniques: Using Function Points With Neural Networks, Case-Based Reasoning and Regression Models*. Journal of Systems Software. 1997; 39: 281 – 289.
- Heiat, Abbas and Heiat, Nafisseh. *A Model for Estimating Efforts Required for Developing Small-Scale Business Applications*. Journal of Systems Software 1997; 39;7 – 14.
- Hihn, J., and Habib-Agahi, H. *Cost Estimation of Software Intensive Projects: A Survey of Current Practices*. International Conference on Software Engineering. IEEE Computer Society Press. 1991, pp.276 – 287.
- Jain, Hemant; Vitharana, Padmal; and Zahedi, Fatemah. *An Assessment Model for Requirements Identification in Component-Based Software Development*. Database for Advances in Information Systems. New York; Fall, 2003. Vol. 34, Iss.4; pg. 48.
- Jørgensen, Magne. *A Review of Studies on Expert Estimation of Software Development Effort*. The Journal of Systems and Software. 70; 2004. pp 37 – 60.
- Jørgensen, Magne. *Top-Down and Bottom-Up Expert Estimation of Software Development Effort*. Information and Software Technology 46 (2004) 3 – 16.

- Jørgensen, Magne and Sjøberg, Dag. *Impact of Experience on Maintenance Skills*. Journal of Software Maintenance and Evolution: Research and Practice. 2002. Vol. 14, Number 2, pp. 123 – 146.
- Kemerer, Chris F. *An Empirical Validation of Software Cost Estimation Models*. Communications of the ACM, 1987, v. 30 no. 5, pp. 416 – 429.
- Kemerer, Chris F. *Software Cost Estimation Models*. Butterworth – Heinemann Ltd. Copyright 1991.
- Kitchenham, Barbara; Pfleeger, Shari Lawrence; McColl, Beth; and Eagan, Susan. *An empirical study of maintenance and development estimation accuracy*. The Journal of Systems and Software, 64, (2002), pp. 57 – 77.
- Laird, Linda M. *The Limitations of Estimation*. IT Professional Magazine. November / December 2006, pp. 40 – 45.
- Lederer, A. L. and Prasad, J. *Software Management and Cost Estimating Error*. Journal of Systems and Software. 50 (1), 33 – 42.
- McConnell, Steve. Software Estimation, Microsoft Press, 2006. Redmond, Washington.
- Menzies, Tim, Chen, Zhihao, Hihn, Jairus, and Lum, Karen. *Selecting Best Practices for Effort Estimation*. IEEE Transactions on Software Engineering. Vol. 32, No. 11, November, 2006.
- National Aeronautics and Space Administration (NASA), *NASA Cost Estimate Handbook*, 2002. Available at <http://www.jsc.nasa.gov/bu2/NCEH/NASA%20CH%20Final%20Production%20Copy%20April%2002.pdf>
- Orr, George, and Reeves, Thomas E. *Function Point Counting: One Program's Experience*. Journal of Systems and Software 53 (2000) 239 – 244.
- Pengelly, A. *Performance of Effort Estimating Techniques in Current Development Environments*. Software Engineering Journal 10 (5), 162 – 170.
- Pfleeger, Shari Lawrence; Wu, Felicia; and Lewis, Rosalind. Software Cost Estimation Sizing Methods. Rand Corporation Publications, 2005. Pittsburgh, Pennsylvania.
- Putnam, L. H. *A General Empirical Solution to the Macro Software Sizing and Estimating Problem*. IEEE Transactions of Software Engineering. pp. 345 – 361, July, 1978.

- Putnam, L. H. *The Real Economics of Software Development*. In The Economics of Information Processing. R. Goldberg and H. Lorin. New York; Wiley, 1982.
- Sparling, Michael. *Lessons Learned Through Six Years of Component-Based Development*. Association for Computing Machinery. Communications of the ACM; Oct., 2003: 43, 10.
- Vicinanza, Steven; Prietula, Michael J.; and Mukhopadhyay, Tridas. *Case-Based Reasoning in Software Effort Estimation*. Proceedings of the Eleventh International Conference on Information Systems, 2000.

VITA

Graduate College
University of Nevada Las Vegas

John Farrish

Local Address

3010 West Gilmore Avenue
North Las Vegas, Nevada 89032

Degrees:

Associate of Applied Science, Software Applications and Programming, 2002
ITT Technical Institute, Henderson, Nevada

Bachelor of Science, Business Administration
University of Phoenix, Las Vegas, Nevada

Thesis Title: The Relationship Between Project Characteristics and the Expert
Estimation of Software Development and Maintenance

Thesis Examination Committee:

Chairperson, Ken Peffers, Ph.D.
Committee Member, Marcus Rothenberger, Ph.D.
Committee Member, Honghui Deng, Ph.D.
Graduate Faculty Representative, Pearl Brewer, Ph.D.