ESTIMATING SOFTWARE EFFORT HOURS FOR
MAJOR DEFENSE ACQUISITION PROGRAMS

by

Corinne C. Wallshein
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
the Requirements for the Degree
of
Doctor of Philosophy in
Information Technology

Committee:

Andrew G. Loerch, Dissertation Director

Alexander H. Levis

Andrew P. Sage

Peggy S. Brouse

Daniel A. Menasce, Senior Associate Dean

Lloyd J. Griffiths, Dean, The Volgenau School of Information Technology and Engineering

Date: May 24, 2010

Summer Semester, 2010
George Mason University
Fairfax, Virginia

Estimating Software Effort Hours for Major Defense Acquisition Programs

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctorate of Philosophy at George Mason University

By

Corinne C. Wallshein
M.S., Operations Research
George Mason University, 1997

Director: Dr. Andrew G. Loerch, Associate Professor
Department of Systems Engineering and Operations Research

Summer Semester 2010
George Mason University
Fairfax, Virginia

DEDICATION

This dissertation is dedicated to my family – past, present, and future.

ACKNOWLEDGEMENTS

Dr. Stephen Nash planted a seed when he asked my Operations Research capstone class in May 1997 if anyone was thinking of coming back to George Mason University to get a doctorate. John Riordan, as my coach in a federal leadership program, convinced me to try. Dr. Charles Tichenor, briefing "The Management Science of Function Point Analysis" in April 2003, suggested software cost estimation topics. Dr. Gerald (Jerry) Diaz and Dr. Jacqueline Henningsen, as my supervisors and mentors, nominated me to participate in a leadership development program. The Air Force Scientist and Engineer Career Field Team, my employers, and my family funded my graduate studies. Dr. Henningsen's steady support kept me moving forward. My gratitude is profound for this encouragement and assistance.

I am indebted to the Air Force Cost Analysis Agency and to the Department of Defense's Cost Analysis Improvement Group. They provided supervision, guidance, and access to data and cost estimators. Justin Moul, Dr. Wilson Rosa, Walt Cooper, Xiang-Zhen (David) Lin, John McCrillis, Joe Dean, Mike Popp, Steve Miller, and Krysty Kolesar helped me understand software cost estimating. I very much appreciate Dr. Andrew Loerch, Dr. Daniel Carr, Dr. Stephen Book, Dr. John Tomick, Dr. David Alberts, Dr. Richard Hayes, and Dr. Clifton Sutton for their advice, instruction, and documentation.

My committee members are role models. My dissertation advisor and director, Dr. Andrew Loerch, expertly guided me through the process. Dr. Peggy Brouse focused my thinking and my writing. Dr. Andrew Sage provided key systems engineering resources. Dr. Alexander Levis, as the Air Force Chief Scientist, motivated me to start and, as a key committee member, helped me to finish.

I thank my family, friends and colleagues. The Meeting Street and beach friends were a wellspring of support. I owe Mary Bonnet and Dr. James Harris special thanks.

My husband, Wayne, likened my progress to sports. He coached me to move on to the next shot when I fell short of expectations or to move the yard marker when I met criteria. Our firstborn, Nathaniel, read my drafts. As his writing skills surpass mine, I welcomed his feedback. What our youngest son, Scott, answered when people asked him about his George Mason University sweatshirt continues to charm me. I am very grateful to have such awesome personal, professional, and academic support.

TABLE OF CONTENTS

TABLE OF ACRONYMS

ACRONYM	NAME
ACAT	Acquisition Category
ActHrsK	Actual (Final) Effort Hours in Thousands
ACWP	Actual Cost of Work Performed
AF	Air Force
AFP	Adjusted Function Points
APB	Acquisition Program Baseline
AR	Absolute Residual
ARA	Acquisition Resources and Analysis
ARE	Average Relative Error
AT&L	Acquisition, Technology & Logistics
BAC	Budget at Completion
BCWP	Budgeted Cost of Work Performed
BCWS	Budgeted Cost of Work Scheduled
BRE	Balanced Relative Error
CAE	Component Decision Authority
CAIG	Cost Analysis Improvement Group
CAPE	Cost Assessment and Program Evaluation
CCDR	Contractor Cost Data Report
CER	Cost Estimating Relationship
CFSR	Contract Funds Status Report
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integrated
COCOMO	Constructive Cost Model
COCOTS	Constructive COTS Model
COSMIC	Common Software Measurement International Consortium
COTS	Commercial Off-the-Shelf
CPI	Cost Performance Index
CPLX	COCOMO input variable for Product Complexity
CPR	Contract Performance Report
CSDR	Cost and Software Data Reporting
CV	Cost Variance
CWBS	Contract Work Breakdown Structure
DACIMS	Defense Acquisition Automated Cost Information System
DAES	Defense Acquisition Executive Summary
DAMIR	Defense Acquisition Management Information Retrieval

DCARC ..... Defense Cost and Resource Center
DID ..... Data Item Description
DISA ..... Defense Information Systems Agency
DLA ..... Defense Logistics Agency
DoD ..... Department of Defense
EAC ..... Estimate at Completion
EA/SD ..... Evolutionary Acquisition with Spiral Development
EstCOTS ..... Estimated COTS
EstExtReq ..... Estimated Number of External Requirements
EstInitModCodeK ..... Estimated Thousands of Lines of Modified Code
EstInitNewCodeK ..... Estimated Thousands of Lines of New Code
EstInitUnmodCodeK ..... Estimated Thousands of Lines of Unmodified Code
EstHiExp ..... Estimated Percentage of Highly Experienced Staff
EstHrsK ..... Estimated (Initial) Effort Hours in Thousands
EstNoExp ..... Estimated Percentage of Entry-Level Staff
EstNomExp ..... Estimated Percentage of Nominal (Average) Experienced Staff
EstPstaff ..... Estimated Peak Staff
EstSWreq ..... Estimated Number of Software Requirements
ETR ..... SEER-SEM's Effective Technology Rating
EVMS ..... Earned Value Management System
FLEX ..... COCOMO input variable for Development Flexibility
FP ..... Function Point
GAO ..... U.S. Government Accountability Office
GFS ..... Government-Furnished-Software
GOTS ..... Government Off-the-Shelf
HTML ..... Hyper Text Mark-up Language
IBRE ..... Balanced Relative Error
IEC ..... International Electrotechnical Commission
IEEE ..... Institute of Electrical and Electronics Engineers
IFPUG ..... International Function Point Users Group
ISO ..... International Standards Organization
IT/MIS ..... Information Technology/Management Information System
LOC ..... Lines of Code
LOC ..... Lines of Code
KNEW ..... New Source Lines of Code in Thousands
KSLOC ..... Source Lines of Code in Thousands
LS ..... Logical Statements
MAIS ..... Major Automated Information System
MAR ..... MAIS Acquisition Reports
MBI ..... SLIM's Manpower Build-up Index
MDA ..... Milestone Decision Authority
MDAP ..... Major Defense Acquisition Program
ME ..... Magnitude of Error
MER ..... Magnitude of Error Relative to the Estimate

MIL-HDBK ..... Military Handbook
Mk II ..... Mark II
MMRE ..... Mean Magnitude of Relative Error
MQR ..... MAIS Quarterly Reports
MRE ..... Magnitude of Relative Error
MSE ..... Mean Squared Error
NASA ..... National Aeronautics and Space Administration
NESMA ..... Netherlands Software Metrics Association
OSD ..... Office of the Secretary of Defense
OSS ..... Open Source Software
OTS ..... Off-the-Shelf
PI ..... SLIM's Productivity Index
PMAT ..... COCOMO Variable for Process Maturity
PREC ..... COCOMO input variable for Precedentedness
PRED ..... Prediction Accuracy
PROBE ..... Proxy-based estimation
PSP ..... Personal Software Process
QQ Plot ..... Quantile - Quantile Plot
QSM ..... Quantitative System Management
R&D ..... Research and Development
RAD ..... Rapid Application Development
RDT&E ..... Research, Development, Test and Evaluation
RE ..... Relative Error
RESL ..... COCOMO input variable for Architecture / Risk Resolution
RFP ..... Request for Proposal
RUP ..... Rational Unified Process
RMS ..... Root Mean Squared Error
RRMS ..... Relative Root Mean Squared Error
SAR ..... Selected Acquisition Reports
SCAMPI ..... Standard CMMI Appraisal Method for Process Improvement
SEI ..... Software Engineering Institute
SLIM ..... Software Lifecycle Model
SLOC ..... Source Lines of Code
SOS ..... System of Systems
SPI ..... Schedule Performance Index
SQL ..... Structured Query Language
SRDR ..... Software Resources Data Report
SV ..... Schedule Variance
SW-CMM ..... Software Capability Maturity Model
TEAM ..... COCOMO Variable for Team Cohesion
UFP ..... Unadjusted Function Points
VAF ..... Value Adjustment Factor (for Function Points)
WBS ..... Work Breakdown Structure

TABLE OF TABLES

TABLE OF FIGURES

ABSTRACT

ESTIMATING SOFTWARE EFFORT HOURS FOR MAJOR DEFENSE ACQUISITION PROGRAMS

Corinne C. Wallshein, M.S.

George Mason University, 1997

Dissertation Director: Dr. Andrew G. Loerch

Software Cost Estimation (SCE) uses labor hours or effort required to conceptualize, develop, integrate, test, field, or maintain program components. Department of Defense (DoD) SCE can use initial software data parameters to project effort hours for large, software-intensive programs for contractors reporting the top levels of process maturity, assuming these levels produce acceptable quality. Statistical analysis using ordinary least squares (OLS) proved initial parameters, such as estimated hours, initial peak staff, or estimated software size, could accurately predict actual effort hours. DoD cost estimating relationship (CER) equations differed by process maturity levels and differed for application area subsets by process maturity level. Grouping by application area subsets or adding Earned Value Management System's metrics (such as Schedule Performance Index and Cost Performance Index) did not consistently improve CER accuracy for the top two process maturity levels.

1. Introduction

In DoD, software cost estimates provide a foundation for budgeting and funding. Cost estimates are required, use of a Work Breakdown Structure (WBS) is required, but estimating techniques and WBS sub-levels vary across DoD. Top WBS levels, suggested in the DoD Military Handbook (MIL-HDBK) 881A for common application categories, are not mandated. Software cost estimation has taken on greater importance as investments in information systems, networks transferring information, and software-intensive program development have increased. Demand for reasonable and accurate estimates of software development costs and maintenance costs places a premium on the development and use of analytic tools. DoD's growing collection of programmatic and software data, available to government cost analysts, is intended for the creation of software cost estimating relationships (CERs) and cost analysis tools. A key data source is the Software Resources Data Reports (SRDRs), storing initially estimated software and process metrics with the contractor's delivered, actual software and process metrics.

Motivation and Background

Although "it is difficult to get a balanced view on the software industry's estimation performance without unbiased information from a representative set of projects and organizations", software cost overruns ranged from 33% to 89%. [Molokken and Jorgensen

2003] The Government Accountability Office (GAO) studied high visibility, large DoD programs' baseline Research, Development, Test, and Evaluation (RDT&E) cost estimates and found systemic cost growth, averaging forty percent. Underestimation of software activities may have resulted in cost overruns. [GAO 2008] Estimating software is complicated by the "unique aspects of software engineering: no physical properties; lack of product visibility; very few product metrics; multiple development strategies; changing/evolving requirements; apparent ease of change; propagation of change; little use of pre-existing components; and white-collar craftsmen and women." [Nidiffer 2006] DoD's historical software resources data reports (SRDRs) provide the opportunity to experiment with deriving evidence-based cost estimating relationship (CER) equations and comparing accuracy measures.

Estimating techniques rely on historical data or human judgment; one author refers to this dichotomy as many-data or sparse-data techniques. [Myrveit et al. 2005] Reported DoD documentation varies from many-data to sparse-data; access is generally restricted to DoD personnel. Since DoD officials promoted the use of Commercial-Off-The-Shelf (COTS) software and non-developmental item "products and services when refining, reengineering or redesigning functional processes", many programs incorporate them. [DoDD 8000.01 2002] An alternative to building applications with custom software, under current regulations, is to build these applications using pre-existing software. Pre-existing software comes from software libraries, other developers or other projects (commonly referred to as Government Off-The-Shelf [GOTS]), the Internet (commonly

referred to as Open Source Software [OSS]), and vendors of COTS software components. In DoD, COTS, GOTS, and OSS packages coexist in many programs. At least 70% of new corporate software applications used COTS in 2003, according to the Gartner Group, and more than 90% of future corporate software applications will use COTS. [Ayala 2008] Although many metrics are collected, quality metrics are absent, save the Software Engineering Institute (SEI) Capability Maturity Model Integrated (CMMI) level rating attained by the contractor.

DoD acquisition system for large, software-intensive programs is documentation-intensive, due to myriad statutory and regulatory requirements, including ones to establish an Earned Value Management System (EVMS). In the SRDR reports, the contractor describes the most recently achieved process maturity level. A key assumption in this dissertation is that developers with the two highest level of process maturity produce software of high quality and record high quality data. High levels of process maturity reflect a continuous focus to reduce defects, achieve stakeholder satisfaction, and enhance software management, maintenance and improvements. [http://www.sei.cmu.edu/cmmi/ 2009] High levels of process maturity should translate to reduced defects in software products (including documentation), improved customer satisfaction, and improved follow-on software maintenance and upgrade activities. Studies have shown higher CMMI levels correspond to lower defect levels. [http://www.sei.cmu.edu/library/assets/2004-CMMI-006.pdf 2009] Among the many possible accuracy measures for the proposed effort-estimating relationships, two metrics

in the literature, Mean Magnitude of Relative Error (MMRE) less than 25% and for 75% of predicted values to be within 25% of actual values (PRED), are considered "industry-standard". [Conte, Dunsmore and Shen 1986; Subramanian 1991; Wieczorek and Ruhe 2002]

Need

Although many software cost models exist, including the suite of Constructive Cost Models (COCOMO) with a setting for process maturity, none of the models or studies have studied CMMI Level 5 and Level 4 organizations together. One article in the literature studied 37 projects from four CMMI Level 5 organizations and contrasted their study to the only other one they found on productivity and conformance within a single CMMI Level 5 organization. The authors found "a steep reduction in variance" in effort and cycle time and a reduced significance of software modeling factors such as personnel capability and requirements specifications as "a potential benefit of achieving high process maturity" leading them to speculate they could port their models across CMMI Level 5 organizations. [Agarwal and Chari 2007] These authors, like many before them, declared software size as the most significant variable to predict development effort and cycle time.

Problem Statement

Many DoD cost analysis professionals use the current contract or historical contract performance as a basis for estimating future project costs, regardless of CMMI level and regardless of factors other than the original bid. This dissertation analyzed 30 projects from ten CMMI Level 5 organizations and 34 projects from five CMMI Level 4 organizations to develop and determine the accuracy of parametric cost estimation models by CMMI level in lieu of accessing all historical contractor data. In addition, the choice of the top two CMMI levels was a conscious choice to be surrogates for quality software. DoD does not make a catalog software defect rates available in their online acquisition repositories. This dissertation extracted initial estimates of software parameters and analyzed them (not actually produced software parameters which are a primary source for other software cost models) to determine whether they could predict final, actual effort. Earned Value Management System (EVMS) data, from the start of the code and test phase, added a set of modeling parameters to test whether an index for schedule performance or cost performance impact estimation accuracy. Data by CMMI level went into non-overlapping application areas created by the author and tested to determine whether the subset's parametric cost models had greater accuracy than the parametric cost models by CMMI level. The models were analyzed using visualization, goodness of fit measures, and industry-standard accuracy measures for software cost and effort models. The accuracy measures relied upon the magnitude of relative error measures, comparing the predicted effort using the model to the actually reported final effort hours.

Dissertation Organization

The dissertation has five main sections. The first main section titled 'Literature Search' discusses software cost estimation, software methods, software data sets, software cost models, software process maturity levels, software and project metrics (including EVMS and CMMI), DoD acquisition categories and DoD databases. Appendices supplement the literature search. The second main section discusses the research methodology. The third main section details the research results. The second-to-the-last main section reports dissertation conclusions. The last section provides suggestions on future research.

2. Literature Review

Figure 1 shows the organization of my literature review, conducted on "widely scattered" references. [Brooks 1995; McConnell 2006] While literature on predicting costs for software programs is growing, published, public domain, studies on costing major DoD software-intensive acquisition programs are scarce due to data sensitivity and relative newness of the data. Records analyzed in this dissertation dated from 2003 to 2008.

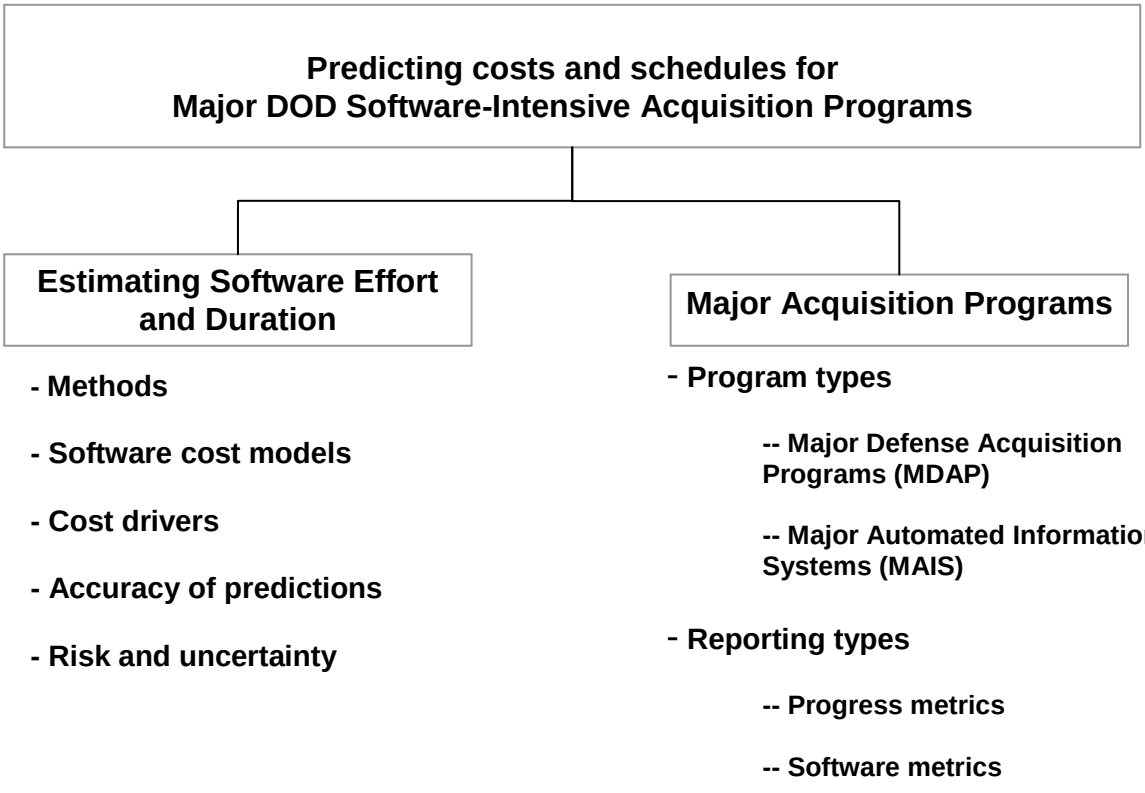

Figure 1: Literature Map (Top-level)

Estimating Software Effort

Software effort estimates provide a basis for funding and budgeting decisions by all levels of management. When estimates err optimistically, overruns generally result; when they err pessimistically, waste generally results. Optimism is an oft-reported common characteristic of software professionals. [Brooks 1995; Nidiffer 2006] Since software has no physical form, its creation can compare to designing and manufacturing widgets, writing short stories or novels, or solving a mathematical problem or proof. [Putnam 2007; Lewis 2001] While white-collar software professionals realize fostering software takes money and time, they traditionally underestimate both. [Nidiffer 2006] Of the three managerial factors to balance in software efforts, i.e., cost, schedule, and delivered functionality (including required quality), past research has focused on estimating software effort hours as a proxy for software cost. [Gilb 1986] This dissertation continues the tradition of past research.

Software Estimating Methods

In software, "…progress depends on groups of humans doing highly interrelated, creative or thinking work in a systems context…" [Putnam 1980] To estimate the future, cost estimates use historical metrics, some of which are listed in the succeeding graphic. [https://www.softwaretechnews.com/stn_view.php?stn_id=47 2008]

Figure 2: Effort = *function* (size, productivity)
Cover page from Data and Analysis Center (DACS) Software Tech News, October 2008

To calculate resources required, particularly effort, we start with an estimate of software size (in lines of code, function points, object points, or expected amounts of reused code with an uncertainty range on the estimated size) and an estimate of productivity. Function points are "based on a rather old-fashioned underlying concept (or metamodel) in which all systems are seen to consist of two parts: database structures and functions that access those structures." [Moser and Nierstrasz 1996] While the program's desired function or performance forms the basis for counting function points, object-oriented programs lend themselves to counting numbers of objects or object points. Estimated productivity from historical data, measured consistently, is defensible. With increased hardware capacity for software storage, many developers reported increased productivity. [Boehm 1981] System-of-systems attributes, such as interaction complexity, code coupl-

ing management, and architecture, impact productivity. [Yu, Smith and Huang 1991; Cain and McCrindle 2002] Documented productivity increases with highly capable personnel or teams are widely reported. [Boehm 1981; Vosburgh, Curtis, Wolverton, Albert, Malec, Hoben and Liu 1984; Yu, Smith and Huang 1991] Other research on programmer's experience levels "showed non-significant relationships to productivity measures." [Chand and Gowda 1993] Resource constraints (i.e., scheduling constraints, timing constraints, memory utilization constraints, and CPU occupancy constraints) decreased productivity, either singly or in combination. [Vosburgh, Curtis, Wolverton, Albert, Malec, Hoben and Liu 1984; Peters, O'Connor, Pooyan and Quick 1984; Jeffery 1987]

In a small software firm, adopting agile development practices increased productivity. [Maurer and Martel 2002] In a large firm (Lockheed Martin Integrated Systems and Solutions), receiving the highest possible process maturity rating (Software Engineering Institute [SEI] Capability Maturity Model Integrated [CMMI] Level 5) related to improved productivity. [McLoone and Rohde 2007] Studies have shown moving up one CMMI level can increase productivity. [Clark 1997; McConnell 2000]

Software effort estimates are generally in person-hours; DoD data repositories align with this tradition. Software size, typically in source lines of code (SLOC), or function points (FPs), provides a basis for software effort estimating techniques and acts as a primary input to popular, contemporary software cost estimating models. DoD data repositories store SLOC by programming language in mutually exclusive areas: new SLOC, modified

SLOC, and unmodified SLOC. Since informal and multiple conventions for measuring SLOC exist, such as logical SLOC, physical SLOC, non-commented SLOC, developers provide explanations on their reported software measurements.

Creating software involves extensive labor with corresponding effort hours, therefore, hours multiplied by an average cost per hour approximate software cost. Software estimating methods depend on effort drivers. Once actual effort occurs, the accuracy of the estimates can be determined. Software cost models have various inputs and algorithms to estimate effort. No estimate is complete without explicit treatment of risk (i.e., acknowledged hazards that may occur) and uncertainty (i.e., acknowledged probabilities of the hazards and of occurrences affecting the estimate). Cost analysis courses, DoD guidance, and an Air Force handbook describe ways of handling risk and uncertainty in software program acquisition and cost estimates.

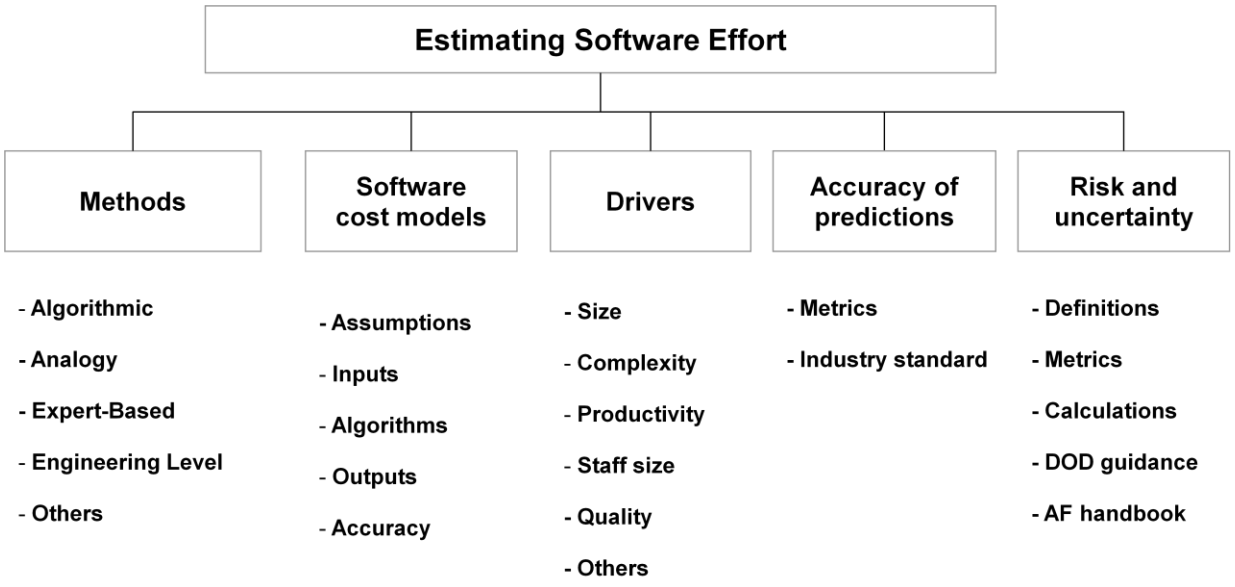

Figure 3: Literature Map (Level 2 - Estimating Software Effort)

There are multiple ways to categorize estimating software effort due to the linkage between forecasting methods, cost drivers, software cost models, prediction accuracy, and risk computations and uncertainty area identification. All forecasting methods and models require objective or subjective inputs, referred to as cost drivers. "Cost drivers are used to capture characteristics of the software development that affect the effort to complete the project." [Boehm, Abts, Brown, Chulani, Clark, Horowitz, Madachy, Reifer and Steece 2000]

The following table describes the different methods listed and gives an overview of their strengths and weaknesses.

Table 2-1: Software Estimating Methods [Boehm 1981; Ormon 2002; Briand, El Eman and Bomarius 1998; Schoedel 2006]

METHOD	DESCRIPTION	STRENGTHS	WEAKNESSES
Algorithmic	A procedure, formula, and/or constraint rules to estimate effort or duration – includes automation and checklists	Objective method able to be repeated and analyzed	Susceptible to incorrect and/or subjective inputs; combining data elements into a single metric can only be as objective as the least objective element
Analogy	Historical precedent or representative experience with a comparable or related item/project/system/program	History tends to repeat itself, with nuances	The historical precedent or representative experience may not apply
Expert-based	Consultation with one or more experts, making use of elicitation techniques such as questionnaires, panels, the Delphi method, or an expert knowledge base	Proficiency basis to assess representativeness, interactions, and special circumstances	Susceptible to biases, subjectivity, incomplete recall, group dynamics, and expert's ability to project experience into new situations
Engineering Level	Individual items estimated separately, then added for subtotals and grand totals; entails use of a WBS. Top-down focuses on system or program level; Bottom-up details use a disaggregated approach to calculate discrete cost elements or items.	Ideal method for known, stable systems with multi-level WBS	Top down has little detail and justification with fewer lower WBS levels; bottom up overlooks costs not detailed or justified. Both require much effort to classify costs in defined WBS levels
Others	Price-to-win, Parkinson, Hybrid, Personal Software Process (PSP) and PROxy Based Estimation (PROBE)	Price-to-win strategy obtains contracts; Parkinson's law correlates with some experience; Hybrid combines methods: PSP estimates size and effort at software design inception; PROBE estimates software size and effort – and can be used for Structured Query Language (SQL)	Price-to-win generally overruns costs and schedules; Parkinson fosters poor practices; Hybrid is as good as the least stable method used; PSP and PROBE require specialized collection of historical software development data

Detail regarding the software estimating methods listed in this table is in Appendix A.

Software Cost Models

Industry software cost estimation models referenced by sources in my literature review are:

1. University of Southern California (USC) Center for Systems and Software Engineering (CSSE) COCOMO model suite
2. Quantitative Systems Management (QSM) Incorporated's SLIM model
3. PRICE Systems TruePlanning model (formerly PRICE-Software and TRUE-Software)
4. Galorath Incorporated's SEER-SEM model
5. Software Engineering Incorporated (SEI) Sage model, and
6. Software Productivity Research (SPR) Incorporated's KnowledgePLAN (formerly CHECKPOINT) model.

Details beyond that in the table below are in Appendix B.

Table 2-2: Software Cost Models' Parameters: Inputs and Outputs [http://csse.usc.edu/tools/COCOMOII.php 2009; S3DB 2005; http://cost.jsc.nasa.gov/pcehhtml/pceh225.htm 2009; http://fast.faa.gov/pricing/c1919-19D.htm 2009; http://seisage.net/sage.htm 2009;Nidiffer 2006]

Software Cost Model	Inputs	Outputs
COCOMO II	**Size** (*New, Reused, Modified SLOC, along with % Design Modified, % Code Modified, % Integration Required, Assessment and Assimilation, Software Understanding, and Unfamiliarity*), 5 *Scale Drivers* (*Precedentedness (PREC), Development Flexibility (FLEX), Architecture / Risk Resolution (RESL), Team Cohesion (TEAM), Process Maturity (PMAT) related to SEI Capability Maturity Model*), 17 **Cost Drivers** ((*5 for Product: Required Software Reliability; Data Base Size; Product Complexity; Required Reusability; Documentation to Match Lifecycle Needs),(6 for Personnel: Analyst Capability; Programmer Capability; Personnel Continuity; Application Experience; Platform Experience; Language and Toolset Experience),(3 for Platform: Execution Time Constraint; Main Storage Constraint; Platform Volatility), and (3 for Project: Use of Software Tools; Multi-Site Development; Required Development Schedule*), **Labor Rates**, and **Staffing Percentage by Phase** (*inception, elaboration, construction, and transition*)	Effort hours Schedule in calendar months Costs in U.S. dollars
SLIM	Languages (*Choice or mix*), System or Application Type (*9 available*), Environmental Information (*Tools, methods, practices, database usage, availability/use of standards*), Process Productivity Parameter (*scale ranges from 0 to 40*), Management Constraints (*Planned schedule, cost, staff size, and software reliability*), Accounting (*such as labor rates and inflation rates*), Flexibility (*Milestones, Phase Definitions, and fraction of time and effort applied*)	Development Time Cost Effort Reliability Expected with risk profiles Comparisons with similar projects
SEER-SEM	Size (*Minimum, Maximum, and Most Likely SLOC, traditional FP, Galorath FP – New, Preexisting and designed for reuse, or Pre-existing and not designed for reuse*), Knowledge-base Inputs (*Platform, Application, Acquisition Method, Development Method, Development Standard, Class*), Complexity, Personnel capability and experience, Development Support Environment, Product Development Requirements, Reusability Requirements, Development Environment Complexity, and Target Environment, Schedule Constraints, Labor Rates, Integration Requirements, Personnel Costs, Metrics, and Software Support	Size Effort Schedule Effective Technology Rating (ETR) during the estimation process

Software Cost Model	Inputs	Outputs
PRICE-S → TRUE-S → TruePlanning	Application Type (*7 basic functional categories*), Productivity Factor, Complexity, Platform, Utilization (of processor capability), Level of New Design and Code, Internal and External Integration Effort, Schedule Start Date, Schedule End Date, Programming Language(s), Economic Factors	Effort estimate in person-months Schedule estimates by milestones Staffing Profile with available sensitivity and schedule effect analysis with summaries for project management
CHECKPOINT → KnowledgePLAN®	Size (*Converts SLOC inputs to FP; uses FP sizing or analogy*), Project description information, Project Nature (*New program , Enhancement, or Conversion, Re-engineering, Maintenance, etc.*), Project Scope (*Stand-alone program, System-of-system, Prototype, etc.*), Project Class (*Single site, multi-site, or network for contract, commercial, government, IT/MIS, etc.*), Project Type, Software Products (*for sizing by analogy*), Personnel attributes (*for project management, development experience, user personnel experience, and quality experience*), Technology attributes, Process attributes, Environment attributes, and Product Factors	Schedule Staffing Effort estimates in dollars or person-months using tabular or graphical Gantt charts
Sage	Size (*SLOC, New source code, Modified source code, Reused source code, executable statements, data declarations, compiler directives, and format statements*), Personnel attributes (*for analyst and programmer capability, application experience, development experience, programming language experience, practices/methods experience, target systems experience. continuity*), Support attributes (*development system complexity, development system volatility, modern practices use, process improvement, practices/methods volatility, reusability level required, required schedule, automated tool support*), Management attributes (*multiple classification levels, multiple development organizations, multiple development sites, resource/support location access*), Product attributes (*staffing complexity, special display requirements, development rehosting, target system memory constraints, required software reliability, real-time operations requirements, system requirements volatility, system security requirements, system CPU timing constraint, target system volatility*)	Most likely and worst case cost and schedule predictions Cost and schedule risk estimates Resource and staff profiles for development Size growth predictions Comparison of estimates with historical data

Cost Drivers

Potential cost drivers or estimating parameters for software cost and schedule, used in some form in the software cost models and described in detail later are software size; staff size; COTS; EVMS; quality; and others. There should be a logical, proven link between the measure and its purpose. The reason for the metric should be clear to the measures' providers and users. "Unfortunately, metrics tend to describe properties and conditions for which it is easy to gather data rather than those that are useful for characterizing software content, complexity, and form." [Tucker 1996]

Software cost estimating drivers depend on data collected as this provides the finite set of possible variables. Over time, with growing data collection, this set has expanded. Traditional software cost estimate drivers derive from the collective history of this craft. "Cost drivers are used to capture characteristics of the software development that affect the effort to complete the project." [Boehm, Abts, Brown, Chulani, Clark, Horowitz, Madachy, Reifer and Steece 2000] Size, complexity, productivity, planned schedule, personnel experience, personnel numbers, requirements volatility, and computing constraints are the backbone of software effort and duration equations in the reviewed software cost models. Additional parameters (such as application type, domain, development method, programming languages, and COTS integration) joined the set of hypothesized effort and duration drivers. "From a conceptual standpoint the derivation of estimating relation-

ships may perhaps best be viewed as a process involving the testing of hypotheses. This implies that the cost analyst should start out by developing a theory about the possible generators of cost for the particular activities, equipments, or facilities under consideration. Then certain hypothesis can be formulated and tested in light of the available data base." [Fisher 1970] Other variables emerge as our metrics set expands.

"Once we understand engineering as an economic-cooperative game, the difficulty of accurately predicting the trajectory of an engineering project becomes understandable…we need different terms: methodology size, ceremony, and weight; problem size; project size; system criticality; precision; accuracy; relevance; tolerance; visibility; scale; and stability." [Cockburn 2006] The advice to "Look for something to count that is a meaningful measure of the scope of work in your environment" merges with Cockburn's statements. [McConnell 2006] So far, however, we may only study software with the variables we collect, test, and use in our experiments, models, and work places.

Software Size

Source lines of code (SLOC) or (LOC) is a long-standing and familiar software size metric. Many authors use SLOC or LOC in software cost studies. [Najberg 1988; Long and Lucas 1996; Vijayakumar 1997; Gaylek, Long, Bell, Hsu and Larson 2004; Misra 2005; Martin, Pasquier, Yanez and Tornes 2005; Mohagheghi, Anda and Conradi 2005; Twala, Cartwright and Shepperd 2005] Others use function points (FP). [Heemstra and Kusters 1991; Briand, El Emam, Surmann, Wieczorek and Maxwell 1999; Angelis, Stamelos and

Morisio 2001; Wieczorek and Ruhe 2002; Foss, Stensrud, Kitchenham and Myrtveit 2003; Liu and Mintram 2005; Tan and Mookerjee 2005; Ahmed, Bouktif, Serhani and Khalil 2008; Gupta, Kaushal and Sadiq 2008; Lan 2008] A raw, unadjusted function point count (UFP) can be determined from the data function type (Internal Logical File and External Interface File) and the transactional function type (External Input, External Output, and External Inquiry). These data function types and transactional function types are five major components to be "classified, ranked, and tallied." [Agarwal, Kuman, Yogesh, Mallick, Bharadwaj and Anantwar 2001]

Calculating a Value Adjustment Factor (VAF) from the degree of influence of fourteen general system characteristics is the International Function Point Users Group (IFPUG) approach. Adjusted function points (AFP) equal the product of UFP times VAF. Some authors use adjusted function points. [Andreou, Papatheocharaous and Skouroumounis 2007; Deng, Purvis and Purvis 2007; Keung and Kitchenham 2007; Andreou and Papatheocharaous 2008]

Barry Boehm's original COCOMO model counted delivered source instructions. "It turns out that the most significant input to the COCOMO II model is Size." [Boehm, Abts, Brown, Chulani, Clark, Horowitz, Madachy, Reifer and Steece 2000] However, what one expects to code may or may not be what one codes. The size measure, from the first to the last effective estimate, remains a best guess; derived in the varied ways as effort and schedule estimates. "The sizing of software programs in terms of source lines of

code has long been a subjective art at best." [Kalb 1988] Alongside this subjectivity, sizing metrics vary widely; SLOC is one metric among many choices for sizing. "The problem is that there is no agreement among professionals as to the right units for measuring software size or the right way to measure within selected units." [Minkiewicz 2008] Lack of agreement on sizing metrics began early. Compounded by programming languages' use of delimiters between statements, such as semi-colons, one line in a program can house many logical statements. [Conte, Dunsmore and Shen 1986]

The International Standards Organization (ISO), along with the International Electrotechnical Commission (IEC), declaration of "Let the market decide" created multiple function point counting standards in use today. [http://www.cosmicon.com/historycs.asp 2009] Namely, there is ISO/IEC 20926 for the International Function Point Users Group (IFPUG) method's functional size component, ISO/IEC 24570 for the Netherlands Software Metrics Association (NESMA) functional sizing measurement, ISO/IEC 19761 for Common Software Measurement International Consortium (COSMIC) functional size metric, and ISO/IEC 20968 for the Mark II (Mk II) Function Point Analysis method.

Only at the conclusion of the effort, can size be determined with any confidence within the choice of measurement scale. Developing working code requires dual understanding. The first understanding is how-to-code in the designated development. The second is what-to-code. Understanding requires learning, and learning classically demonstrates non-linear behavior. Typical learning curves have formulas of "$L(y) = Ay^b$, where $L(y)$ =

the number of hours needed to produce the *y*th unit, *A* = the number of hours needed to produce the first unit, *y* = the cumulative unit number, and *b* = the learning index, the learning-curve parameter, or the learning-curve slope parameter." [Loerch 2001]

Although what constitutes a line of code has been argued over, one book stated this definition as a common one for researchers: "A line of code is any line of program text that is not a comment or a blank line, regardless of the number of statements or fragments of statements on the line. This specifically includes all lines containing program headers, declarations, and executable and non-executable statements." [Conte, Dunsmore and Shen 1986] SLOC measurement scales include logical statements, physical LOC, and non-commented source lines. "SLOC was selected early as a metric by researchers, no doubt due to its quantifiability and seeming objectivity. Since then, an entire subarea of research has developed to determine the best method of counting SLOC." [Kemerer 1987] Logical and physical are the two primary methods of describing SLOC, although pre-1992 references generally do not specify SLOC type. Logical SLOC counts only logical statements in the software; Physical SLOC counts all but comments and blank lines. [Parks 1992] Despite its longevity, SLOC is challenged by other size measures: Function Points (FP), Object Points, Use Cases Points, Feature Points, Web Object Points, etc. [Jones 2007] Actual counts can change depending on the invocation of an automated code counter for identical source programs. [Reifer 2009] Automated code counters work for only specific languages; other languages require manual counting.

Programming languages, by generation or type, can relate to software size. "Languages impact both productivity and the amount of code that will be generated." [Boehm, Abts, Brown, Chulani, Clark, Horowitz, Madachy, Reifer and Steece 2000] Low- level languages take many more source code statements to perform operations than high-level languages. [Gao and Lo 1994] There are currently five generations of languages from low level (1st and 2nd generation) to high level (3rd generation and beyond). FORTRAN and COBOL are example 3rd generation languages (3GLs). SAS and Cold Fusion are example 4th generation languages (4GLs). The 5th generation languages (5GL) build specified constraints via logic into the software development environment to allow and disallow certain user-machine interactions. [http://e-words.us/w/5GL.html 2009] There is debate as to whether 5GL exist. [http://www.it-director.com/content.php?cid=9096 2009] In addition to generational languages for general-purpose use, there are domain languages for specific-purpose use. For instance, the domain language of Cold Fusion Mark-up Language is akin to Hyper Text Mark-up Language (HTML), the difference being specialized server tags.

A rough conversion between two methods of counting source lines of code, logical statements and physical lines of code is in the following table, based on NASA's Jet Propulsion Laboratory historical databases:

Table 2-3: Converting Software Size Estimates [http://software.gsfc.nasa.gov/docs/QSM-class/Day%201-Cost/04a-Size.ppt 2009]

Language	**To Derive Logical SLOC**
Assembly and Fortran	Assume physical SLOC = Logical SLOC
Third-Generation Languages (such as C, Cobol, Pascal, Ada 83)	Reduce physical SLOC by 25%
Fourth-Generation Languages (such as SQL, Perl, Oracle)	Reduce physical SLOC by 40%
Object-oriented Languages (such as Ada 95, C++, Java, Python)	Reduce physical SLOC by 30%

The next table illustrates the order of magnitude between the programming language's generation and the average source lines of code per functionality represented by raw, un-adjusted function points, a standardized count of data function type and the transactional function type. This dissertation incorporated the preceding table and recent research on converting non-commented source lines of code and physical lines of code to logical statements to normalize the software size metric.

Table 2-4: Functionality versus Size by Programming Language Generation [Jones 2007]

PROGRAMMING LANGUAGE	***AVERAGE SLOC PER UFP***
5GL default	5
4GL default	20
3GL default	80
2nd Generation default	107
1st Generation default	320
Machine language	640

Staff Size

Peter Norden depicted numbers of workers visually in a series of graphs representing the Research and Development (R&D) phase manpower build-up and ramp-down. [Putnam 1980] The patterns in the graphs led him to experiment with Rayleigh probability distribution parameters. After experimenting, he found parameters to fit the graphs. Putnam's SLIM model and the COCOMO models use the Rayleigh distribution to estimate typical labor build-ups and ramp-downs, while the more adaptive Gamma distribution may improve predicted labor curves for less typical labor patterns and performed well for quicker labor build-ups. [Pillia and Nair 1997] Norden reports, "The purposes change throughout the life of a project, and these changes characterize the effort cycles …The cycles do not depend on the nature of work content of the project but seem to be a function of the way groups of engineers and scientists tackle complex technological development problems. Each cycle can be described by a comparatively simple equation: $y' = 2Kate^{-at\wedge 2}$ where y' = manpower utilized each time period, K = total cumulative manpower utilized by the end of the project, a = shape parameter (governing time to peak manpower), t = elapsed time from start of cycle." [Putnam 1980]

Mr. Putnam defined software state variables as follows: state of technology C_n or C_k; applied effort K; development time t_d; and independent variable time t. "The software equation relates the product to the state variables: … $S_s = C_k K^{1/3} t_d^{4/3}$. The tradeoff law $K = C/t_d^4$ demonstrates the cost of trading development time for people." [Putnam 1980] In-

dividual and team efforts vary according to the number of workers and the ability of the team to partition the workload.

Pair Programming has become a best practice along with Mike Fagan's code inspections [Nidiffer, 2006]. "Since software construction is inherently a systems effort – an exercise in complex interrelationships – communication effort is great, and it quickly dominates the decrease in individual task time brought about by partitioning. Adding more men then lengthens, not shortens, the schedule." [Brooks 1995] For group intercommunication channels, with n = number of communicating people, c = the number of lines of communication required, can be calculated by the formula, c =$\frac{n(n-1)}{2}$. [Brooks 1995]

Large teams have a "very substantial impact on project productivity, thereby confirming that compressing cycle time…comes at a substantial additional cost." [Briand, El Emam and Wieczorek 1999] This additional cost is likely due to the increase in group's communications, the transmission of clear direction and instructions, and the understanding of the technical and organizational processes needed to perform the intellectual work to produce the software. In tandem with the increased group interaction channels, larger teams appear to create more defects, resulting in substantial rework to fix them. [Armel 2006]

TruePlanning, from PRICE Systems, bounds staff size, as a user input, from the minimum expected to the maximum. SLIM, from QSM Incorporated, requests peak staff as an input. The staff's experience and continuity, rather than the expected number of staff, are input variables in COCOMO II, Sage, and SEER-SEM. Staffing profiles are outputs, with corresponding peak staffs, for the following software cost models: SLIM, TruePlanning, Sage, KnowledgePLAN, and SEER-SEM.

COTS

Like any shared resource, using COTS requires planning and coordination to work well. COTS products target a commercial market segment, and within that segment, offer options. That COTS components can fulfill a program's purpose is a coincidence. [Albert and Brownsword 2002] "COTS components introduce 'hard points' into the system architecture before the system has been fully optimized and matured, resulting in the need for non-COTS components to conform to COTS established interfaces." [Sage 2005] COTS components manifest vendor architecture design and paradigm assumptions as well as end-user process assumptions. [Albert and Brownsword 2002] The following terms describe COTS: white box, grey box, and black box software. A white box is transparent as source code is visible and changeable. Open Source Software (OSS) is white box. A grey box has the capability to interface with another component via its own extension language or application programming interface (API). [Sage 2005] A black box is opaque as source code is neither visible nor changeable. A black box is "where only a binary executable form of the component is available and there is no extension

language or API." [Sage 2005] Proposed variables for judging matches between software architecture and component integration opportunities are: (1) packaging, (2) control, (3) information flow, (4) synchronization, and (5) binding. [Yakimovich, Bieman and Basili 1999] 'Packaging' refers to the use of independent programs, overlays, dynamic link libraries, or class libraries. 'Control' refers to the type of automatic control mechanism: allowing simultaneous multiple processes, centralizing control, or decentralizing control. 'Information flow' refers back 'control' as processes or data may trigger the use of specific control mechanisms. 'Synchronization' refers to concurrency: processes are either asynchronous or synchronous. 'Binding' can be static or dynamic. [Yakimovich, Bieman and Basili 1999]

"Our research indicates that the number of unique interfaces and the number of different component systems are the two best factors for determining the size of the SOS [System of Systems] effort." [Minkiewicz 2006] In the planning for COCOSIMO, which stands for the Cost of the System of Systems Model, proposed independent input variables relating to the cost of COTS are the number of unique component systems, the number of interface protocols for the system to track and adhere to, and the number of independent component system organizations. [Lane 2007] COCOTS uses the number of different classes of COTS products being tailored as an input variable multiplied by the mean tailoring effort for all the classes of COTS products along with the tailoring complexity qualifier to compute the total COTS product tailoring effort. [Abts 2004]

Earned Value Management System (EVMS)

The Earned Value Management System (EVMS) has lent itself to software project management in DoD. [Lipke 2002] EVMS is an integrated management tracking system of the actual work scope against planned cost and schedule using three standardized metrics: Actual Cost of Work Performed (ACWP), Budgeted Cost of Work Performed (BCWP), and Budgeted Cost of Work Scheduled (BCWS). The actual work scope uses a Work Breakdown Structure (WBS) for contractual work planning and control, where the contractor can adjust the program's WBS into a contract-appropriate WBS called the Contract WBS (CWBS), to crosswalk requirements from successively higher or lower WBS or CWBS levels to higher or lower level design or reporting documents. [http://guidebook.dcma.mil/79/evhelp/wbs.htm 2010]

A Cost Performance Index (CPI) is ——— , the dollar value of work performed divided by the work billed for a given time period. If the number is less than one, the project is over budget by the percentage: ——— . The percent spent is ——— where BAC stands for Budget at Completion representing the total amount negotiated for the contracted individual delivery order. A Schedule Performance Index (SPI) is ———, the dollar value of work performed divided by the work scheduled for a given time period. If this is less than one, the project is behind schedule. If the fractional computation came to 0.80, the

project would be at 80% of the planned schedule or 20% behind. In this dissertation, I use SPI and CPI, to represent cost and schedule performance in similar measurement units.

Percent complete is —— which represents the dollar value of the work performed divided by the total budgeted amount. The Cost Variance (CV) equals so a negative CV means more money went towards the effort than was planned. [DCMA 2006] The Schedule Variance (SV) equals so a negative SV means less work accomplished than planned, but it does not necessarily follow that the project is behind. [DCMA 2006] A zero SV can mean the project is waiting to start or the project proceeded to the plan.

Software Quality Metrics

Quality metrics for software development focus on three areas: the process, the people, and the product. [Reifer 2002] The most common process model for software in DoD is CMMI, though there are others, such as ISO-9000, a management quality standard in the commercial marketplace. In software cost estimating, the proxy for personnel quality is people's experience levels; people with higher levels of experience follow higher quality processes and produce higher quality software. Product quality is an output in software cost estimating models such as SLIM, KnowledgePLAN, and TruePlanning.

Quality could be determined by examining development products such as requirements, plans, and expenditures-to-date. These could be documented requirements, requirements traceability, code quality, documented test results (number of defects, priority of defect, defect density, defect discovery rate, mean time between failures, time to fix defects, re-work from defects, and escapes [defect fixes ahead of schedule]), quality assurance plans, configuration management plans, and resources expended. Establishing a baseline software project or set of projects allows qualitative and quantitative comparisons to the development project. Desired product quality could be an input to the software cost estimate, if such data is collected.

In an empirical study, where four contracts were let, there was a difference in effort expended, based on the input quality goals specified; where higher quality goals resulted in larger effort. [Anda, Benestad and Hove 2005] Buggy software has been associated with larger code sizes as well as larger staff sizes. [Armel 2006] Large software projects may have "requirements errors, design errors, coding errors, user documentation errors, and bad fixes...a bad fix is a failed attempt to repair a prior bug that accidently contains a new bug." [Jones 2005]

The Carnegie Mellon University's Software Engineering Institute (SEI) produced the CMMI to rate a software engineer's process maturity level from the lowest score of zero to the highest score of five. To judge the quality of the software development process, a

model titled Capability Maturity Model for Software (CMM or SW-CMM) was absorbed into CMMI. Ratings for CMM expired as of January 1, 2008. [http://www.sei.cmu.edu/cmmi/faq/comp-faq.html 2009] CMMI provides a process model for improvement of software and systems engineering development practices. For a CMMI rating, organizations begin with the appraisal reference model, follow a formal and collaborative appraisal process, such as the Standard CMMI Appraisal Method for Process Improvement (SCAMPI), involve management, focus on business objectives, maintain confidentiality and non-attribution, and produce an appraisal the organization can act upon to continue to improve in the area certified: Development, Service, or Acquisition. [http://www.sei.cmu.edu/cmmi/adoption/pdf/cmmi-overview07.pdf 2009] There are two structurally different CMMI process models: staged and continuous. For the most part, information technology organizations follow the staged CMMI process. [Yahya, Ahmad and Lee 2008]

DoD contracts with software developers and integrators at all CMMI maturity levels; and "many DoD acquisition programs are including requests for CMMI maturity levels in requests for proposals (RFPs) in spite of the fact that DoD has not promulgated policy requiring adherence to a CMMI maturity rating." [Schaeffer, Osiecki, Richter and Baldwin 2007]

Other Software Cost Estimating Inputs

The other inputs for software cost estimating are complexity, productivity, development method, and life cycle phases. Complexity is an input in many software cost estimation models and has been the subject of much research. Complexity may have a direct relationship with size, where greater complexity corresponds to greater code size. [Booch 1998; Zhao, Tan and Zhang 2003; Armel 2006] Productivity appears to have an inverse relationship to quality, where higher productivity may result in lower software product quality. [Davis 1995; Anda, Benestad and Hove 2005] Development methodologies can pair with life cycle phasing methods. Often-used software development methods are waterfall, spiral, evolutionary, and incremental; often-used life cycle phases in DoD are research and development (including concept development, planning, designing, coding, and testing), production and fielding, and operations and support. Subsequent sections of the dissertation describe these inputs.

Complexity

"Because complexity obscures the perception and understanding of information cues, it is believed to significantly degrade task performance." [Banker, Davis and Slaughter 1998] The COCOMO II model has an input variable, Product Complexity or CPLX. This variable is "divided into five areas: control operations, computational operations, device-dependent operations, data management, and user interface management operations."

[Boehm, Abts, Brown, Chulani, Clark, Horowitz, Madachy, Reifer and Steece 2000] The categorical scale for CPLX is very low, low, nominal, high, very high, and extra high. Settings for this variable are subjective in selecting between categories. [Pfleeger, Wu and Lewis 2005] The definition of complexity for software development and maintenance is far from standardized. Complexity metrics to convert unadjusted function points into adjusted function points, however, have remained the same for over twenty-five years. [Ahmed, Bouktif, Serhani and Khalil 2008] In an attempt to clarify complexity, a "complexity factor" in software development is system size itself. [Armel 2006] Joining system complexity is algorithmic complexity, service software complexity, and database software complexity. [Bennatan 2000] "A great deal more research is needed on all forms of software complexity, and particularly on complexity associated with algorithms, visualization, software requirements, specifications, test cases, and data complexity." [Jones 2007] High levels of complexity exist in large, complex, software-intensive DoD programs.

In 2005, at the IEEE International Symposium of Empirical Software Engineering, Bente Anda, Hans Christian Benestad and Siw Elisabeth Hove reported an interesting multiple-case study on software effort estimation. [Anda, Benestad and Hove 2005] When multiple companies had the same functional specifications, used the same programming languages, had similarly talented workforces, but had different non-functional specifications and used different development processes, actual effort variation was large. "The differences in the development process entailed different assumptions on the non-functional

requirements on the system with respect to the quality of the code…a heavier development process with an increased emphasis on the quality of the code led to a large increase in actual effort…supports previous results on the effect of complexity factors in similar estimation methods." [Lokan and Abran 1999]

Productivity

Productivity computations are implicit or explicit, depending on the method used. Average productivity calculations can misinform software estimates. "The best-producing people can be 20 or more times better than the low-end group. A factor of more than 100:1 may separate programmers at the ends of the spectrum." [Gruschke 2005] Productivity rates can decline from approximately 300 lines of code per month to 85, as the project progresses. [Book 2001] The notion of "average-worker-productivity" discounts the wide variance between a competent and incompetent programmer, logged at 28-to-1 for 'Algebra' program debugging hours and 26-to-1 for 'Maze' program debugging hours. [Sackman, Erikson and Grant 1968] Size and number of components has an impact on project productivity; blocks of existent code and multiple COTS packages slow productivity as the team gains knowledge to build out an integrated product. [Abts 2004; Salter 2001] Productivity is difficult to measure. Productivity metrics such as lines of code per month rely upon the interpretation of the numerator and the denominator. A programmer can be very competent in one setting, where he or she knows the programming language, the domain of the application, the required functionality, the computing environment, and the software components to integrate, and less competent in another.

Assuming programmer continuity, competence through learning behaviors may accompany the next incremental change in languages, domains, requirements, environments, or components.

Development Methods and Life Cycle Phases

The waterfall method is a formal development method, using measured, well-documented processes. The waterfall process "forces the problem to fit the development cycle, rather than the other way around." [Miller, Paradis, and Whalen 1991] Different terms describe different development methods. To support documentation-intensive processes, the heavyweight waterfall process surpasses other methods. [Ikoma, Ooshima, Tanida, Oba, and Sakai 2009] For staged contracts, where software deliveries tie into major milestones, a staged approach bests the traditional waterfall approach. [Lott 1997] The spiral method is iterative allowing for incorporation of feedback upon artifact delivery based on partial specifications. According to the 2003 Bob Stump Authorization Act, the Secretary of Defense can conduct major defense acquisition programs using the spiral method. The 2008 version of DoD 5000.02 states programs should practice evolutionary acquisition. This is an offshoot of an approach using evolutionary acquisition with spiral development (EA/SD) adopted by DoD after 2003. [Pagliano and O'Rourke 2004] Evolutionary development blends incremental deliveries into the whole software effort. Extreme Programming, Agile methods, Rapid Application Development (RAD), and incremental methods are other methods with their own vocabulary, instructions, and followers. While different development methods are well documented, "almost any non-trivial project or

organization must combine technique, common sense, and domain experience." [Royce 2005] Detail on development methods and on life cycle phases is in Appendix C.

Software Cost Estimating Accuracy

To determine a cost estimating relationship's accuracy: predicted values should be generated and compared to actual values; a validation technique should be chosen, defended, and used; and information should be provided on the data set. [Mair and Shepperd 2004] The types and amounts of input data dictate the types of possible cost analyses. Measuring the presence or absence of a characteristic provides some information about that characteristic. Measuring the magnitude of the characteristic provides more information.

"The accuracy of COCOMO II allows its users to estimate within 30 percent of actuals, 74 percent of the time. This level of unpredictability in the outcome of a software development process should be truly frightening to any software project investor." [Reifer 2002] Early software cost estimates within plus or minus 30 percent of the actual cost are successful per Bernard Londeix in *Cost Estimating for Software Development*, published by the Addison-Wesley Publishing Company in New York in 1987. [Mertes 1996] In 1981, Robert Thibodeau published a General Research Corporation paper titled "An Evaluation of Software Cost Estimating Models" reporting model calibration to an organ-

ization's historical data can improve software cost model accuracy by a factor of five [Mertes 1996].

To figure out how accurate you are, you need metrics. "Stevens (e.g., 1968) developed the concept of scales of measurement." [Neale and Liebert 1986] Scales are nominal, ordinal, binary, interval, and ratio. Nominal values can be equal or unequal. Preferences are nominal values: FORTRAN can be preferred to Pascal and logical sizing can be preferred to physical. Ordinal values can be equal, unequal, less than, greater than, and ranked from lowest to highest. Binary values equal 1 or 0. Interval values have the same properties as ordinal and distance from one interval to the next is the same for all values so addition and subtraction are possible. Ratio values permit all mathematical operations. A ratio scale has a true zero point, meaning a complete absence of the measured characteristic. While scale is important, "it will usually be necessary or desirable to describe the set of observations numerically." [Neale and Liebert 1986] Categorical input data causes problems in the creation of regression models, as ordinary least squares performs best when the variables' values are numeric and of similar magnitudes. [Angelis, Stamelos and Morisio 2001] The authors suggest binary (on-off) variables or subjective quantification of categorical values for building models for datasets with few numeric fields. They produced "only one possible statistical cost model using a subset of the [International Software Benchmarking Standards Group's project repository, release 6] database." [Angelis, Stamelos and Morisio 2001] A common problem in data sets is input variables

skew toward few fixed values, rather than spreading out across the range. [Angelis, Stamelos and Morisio 2001]

In software cost estimating, Cost Estimating Relationships (CERs) generally follow the form of '$Y_j = f(x_j, \beta)\varepsilon_j$ for j = 1,…n' where:

n is the sample size

Y_j is the observed cost of the jth data point

$f(x_j, \beta)\varepsilon_j$ is the result of the experimental CER

β is the vector of coefficients estimated by the CER

x_j is the vector of cost driver input variables, and

ε_j is a multiplicative error term with a mean of 1 and variance of σ^2 [SAF 2007]

To transform ε_j into a normal distribution, with mean of 0 and variance of σ^2, use the formula: $_j$ ———. "Regardless of what method is used to generate the CER, it is very important that the user of the CER is aware of the CER result meaning and how the error should be modeled." [SAF 2007]

One recommended way to improve accuracy of software cost models is to calibrate the model to the organization's historical data, assuming the organization has it. From past research in published theses from the Air Force Institute of Technology, calibration has

mixed success. "Even after the software cost models are calibrated to DoD databases, most have been shown to be accurate to within only 25 percent of actual cost or schedule about half the time…Without a holdout sample, the predictive accuracy of the model is probably overstated. Since all new projects are outside of the historical database(s), validation is much more meaningful than the more common practice of analyzing within-database performance." [Ferens and Christensen 2000] A table describing accuracy metrics found in the literature is in Appendix G. The table below highlights the accuracy metrics in this dissertation.

Table 2-5: Industry-Standard Accuracy Metrics [Jalali 2008; Conte, Dunsmore and Shen 1986]

Measure	Description	Meaning
Relative Error (RE)	On the numerator, actual value minus predicted value, all divided by actual value (for each predicted value)	RE can be positive or negative. If RE is negative, it can go to negative infinity. If RE is positive, it cannot exceed a value of one. If RE is zero, the predicted value equals the actual value.
Magnitude of the Relative Error (MRE)	The absolute value of RE	MRE can only be positive. The lower the MRE, the better the prediction; higher MRE means a worse prediction. If MRE is zero, the prediction equals the actual value.
Mean Magnitude of the Relative Error (MMRE)	The sum of all the individual MRE values divided by *n*, the number of values calculated	Generally, smaller MMRE positive values represent better overall agreement in predicted and actual values. However, small MMRE values could mask one or more large deltas. An industry standard is MMRE ≤ 0.25 and is "acceptable for effort prediction" [Conte, Dunsmore and Shen 1986]

Measure	Description	Meaning
Prediction at Level *L* or *l* (PRED(*L*) or PRED(*l*))	In a set of *n* projects, a value, *k*, is the number of projects whose MRE ≤ *l*, so PRED(*l*) equals *k* divided by *n*; *l* is number less than 1.0 whereas *L* is a percentage	For PRED(*l*) = *k*/*n*, the *k*/*n* ratio of predicted values are within *l* percentage of actual values. The accepted industry standard is PRED(0.25) ≥ 0.75 [Conte, Dunsmore and Shen 1986]

S. D. Conte, H. E. Dunsmore, and V. Y. Shen suggest PRED and MMRE thresholds for prediction accuracy. [Mertes 1996; Morgan 1997; Wieczorek and Ruhe 2002] Other measures do not enjoy this citing repetition in the literature. Researchers referencing the COCOMO model suite have reported MMRE and PRED levels from 1981 through 2009. [Boehm 1981; Kemerer 1987; Smith 1998; Reifer 2002; Abts 2004; Valerdi 2005; Shen 2008; Fortune 2009]

Risk and Uncertainty

Treating risk and uncertainty explicitly is a standard practice in software cost estimation. Risk is the chance of an outcome being harmful or damaging; uncertainty is the chance of any possible outcome defined probabilistically. This quote from Cornelius Keating is worth repeating: "Risk is the unwanted subset of a set of uncertain outcomes". [http://dissertations.ub.rug.nl/FILES/faculties/rw/2009/x.c.mandri.perrot/06c6.pdf 2010] The risk of over- or under-estimating software effort or duration can harm the project,

perhaps irreparably, causing resources to be misdirected. The uncertainty in the estimate should reflect all the inputs' uncertainties.

The currency and applicability of historical data varies by application and by time. Data collection mores change, interpretations change, and even definitions change. Static and dynamic factors plague comparability of historical data to contemporary programs. Discovering completed, comparable programs to use as the basis of cost analytic methods is complicated by the situation of "the analyst is all too often in the world of extremely small samples." [Fisher 1970] Even with large samples, there can be problems. "We can assemble big pieces of information and little pieces, but we can never get all the pieces together. We never know for sure how good our sample is. That uncertainty is what makes arriving at judgments so difficult and acting on them so risky." [Bernstein 1996] Further discussion on risk and uncertainty is in Appendix D.

By pairing initially estimated parameters with final, actually performed effort hours at the two highest CMMI levels, the risk of waiting until future projects are finished to gather actual parameter to relate to actual effort disappears. Further, the risk of applying a growth factor to the initially estimated software size to estimate final size and then use a software cost estimating relationship is avoided because this presumed growth (or shrinkage as the case may be) is embedded in the relationship between the initially estimated parameters and the final, reported effort.

DoD Major Acquisition Programs

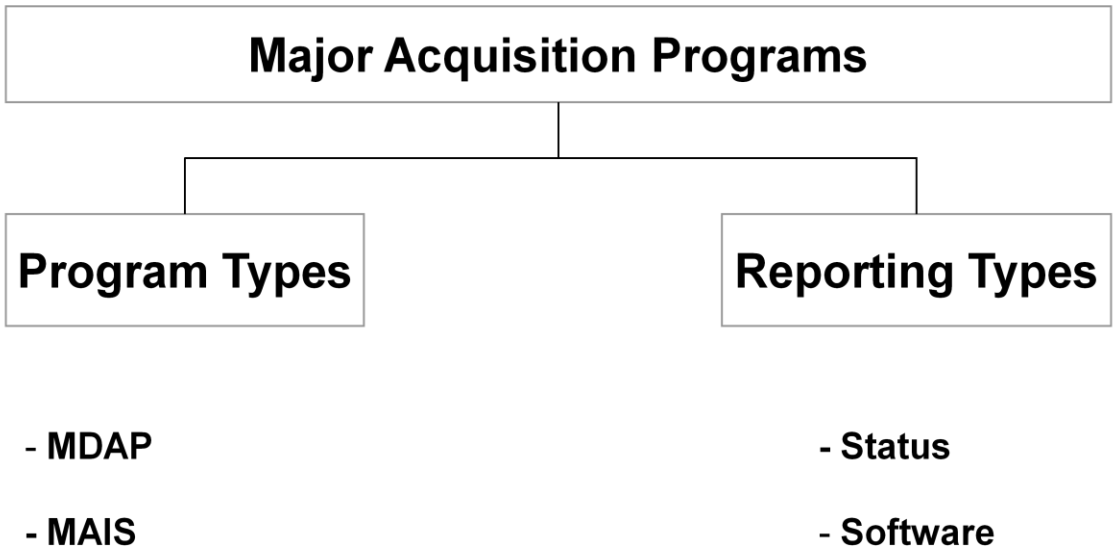

Figure 4: Literature Map (Level 2 - Major Acquisition Programs)

Large, software-intensive programs start out or become Major Defense Acquisition Programs (MDAP), Major Automated Information Systems (MAIS), or both. MDAP and MAIS acquisition plans and program activities meet or exceed dollar thresholds set in Title 10, United States Code. MDAP is depicted as Acquisition Category (ACAT) I, whereas MAIS is depicted as ACAT IA. The designations specify the Milestone Decision Authority (MDA). ACAT ID and ACAT IAM mean DoD is the MDA; ACAT IC

and ACAT IAC mean the Component is the MDA. A component can be a separate Armed Service (e.g., the Air Force, Navy, or Army), or a DoD agency (e.g., Defense Logistics Agency (DLA) or Defense Information Systems Agency (DISA)). The following table provides more detail.

Table 2-6: Description and Decision Authority for Acquisition Category (ACAT) I Programs [DoDI 5000.02, Enclosure 3, Table 1 2008]

Acquisition Category	Reason for ACAT Designation	Decision Authority
ACAT I	• MDAP (section 2430 of Title 10, United States Code) o Dollar value: estimated by the USD(AT&L) to require an eventual total expenditure for research, development, test and evaluation (RDT&E) of more than $365 million in fiscal year (FY) 2000 constant dollars or, for procurement, of more than $2.190 billion in FY 2000 constant dollars o MDA designation • MDA designation as special interest	ACAT ID: USD(AT&L) ACAT IC: Head of the DoD Component or, if delegated, the CAE (not further delegable)
ACAT IA[1, 2]	• MAIS (Chapter 144A of Title 10, United States Code): A DoD acquisition program for an Automated Information System[3] (either as a product or a service) that is either: o Designated by the MDA as a MAIS; or o Estimated to exceed: ▪ $32 million in FY 2000 constant dollars for all expenditures, for all increments, regardless of the appropriation or fund source, directly related to the AIS definition, design, development, and deployment, and incurred in any single fiscal year; or ▪ $126 million in FY 2000 constant dollars for all expenditures, for all increments, regardless of the appropriation or fund source, directly related to the AIS definition, design, development, and deployment, and incurred from the beginning of the Materiel Solution Analysis Phase through deployment at all sites; or ▪ $378 million in FY 2000 constant dollars for all expenditures, for all increments, regardless of the appropriation or fund source, directly related to the AIS definition, design, development, deployment, operations and maintenance, and incurred from the beginning of the Materiel Solution Analysis Phase through sustainment for the estimated useful life of the system. • MDA designation as special interest	ACAT IAM: USD(AT&L) or designee ACAT IAC: Head of the DoD Component or, if delegated, the CAE (not further delegable)

For the most part, MDAP programs result in a tangible product whereas MAIS programs result in an intangible service. [Jones 2009] These different programs are usually interdependent.

DoDI 5000.04-M-1, Cost and Software Data Reporting Policy, dated April 18, 2007, lays out reporting requirements for software-intensive major DoD programs. Cost reports track contractor activities. Software reports collect data on the software activities, components, peak staff, and processes. Although Congress authorized spiral development for MDAP research and development, via Section 803 of Public Law 107-314, in 2003; in December 2008, the newly published instruction, DoDI 5000.02, encouraged evolutionary development. A significant difference between spiral and evolutionary development is the scheduled delivery, under evolutionary development, of fully functional software. Under spiral development, iterative conceptual prototypes aid communications and understandings between developers and users as the system progresses through design to coding and testing. In both development methods, a series of deliveries make up the whole; the difference is the intention in evolutionary development of independent, stand-alone deliveries vice interdependent, iterative deliveries in spiral development.

For MDAP, MAIS, pre-MDAP, and pre-MAIS programs or for ACAT I, ACATI IA, pre-ACAT I, and pre-ACAT IA programs, the Contractor Cost Data Report (CCDR) and the Software Resources Data Report (SRDR) fall under the umbrella of Cost and Software Data Reporting (CSDR). The CCDR dollar threshold is fifty million dollars; the SRDR dollar threshold is twenty million dollars. These thresholds are in then-year or current-year dollars. The table outlines the regulatory contract reporting requirements, as well as the timing of the submissions of these reports, in the DoD manual, DoD 5000.04-M-1.

Table 2-7: DoD Regulatory Contract Reporting Requirements [DoD 5000.04-M-1 2007]

REPORT REQUIRED	WHEN REQUIRED
Contractor Cost Data Report (CCDR)	• All major contracts[1] and subcontracts, regardless of contract type, for ACAT I and IA programs and pre-MDAP and pre-MAIS programs subsequent to Milestone A approval, valued at more than $50[2] million (then-year dollars) • Not required for contracts priced below $20 million (then-year dollars) • The CCDR requirement on high-risk or high-technical-interest contracts priced between $20 and $50 million is left to the discretion of the DoD PM with approval by the Chair, CAIG • Not required under the following conditions provided the DoD Program Manager (PM) requests and obtains approval for a reporting waiver from the Chair, CAIG: procurement of commercial systems or for non-commercial systems bought under competitively awarded, firm fixed-price contracts, as long as competitive conditions continue to exist.
Software Resources Data Report (SRDR)	• All major contracts and subcontracts, regardless of contract type, for contractors developing/producing software elements within ACAT I and IA programs and pre-MDAP and pre-MAIS programs subsequent to Milestone A approval for any software development element with a projected software effort greater than $20M (then-year dollars). • The SRDR requirement on high-risk or high-technical-interest contracts priced below $20 million is left to the discretion of the DoD PM with approval by the Chair, CAIG.
Notes: 1. For CSDR purposes, the term "contract" (or "subcontract") may refer to the entire standalone contract, to a specific task/delivery order, to a series of task/delivery orders, to a contract line item number, or to a series of line item numbers within a contract. The intent is to capture data on contractual efforts necessary for cost estimating purposes irrespective of the particular contract vehicle used. 2. For CSDR purposes, contract value shall represent the estimated price at contract completion (i.e., initial contract award plus all expected authorized contract changes) and be based on the assumption that all contract options shall be exercised.	

All programs meeting these thresholds require a Contractor Cost and Data Report (CCDR) for new, deleted, and changed contracts and subcontracts. A Contract WBS (CWBS) should follow guidelines in MIL-HDBK 881A, with a mapping or translation to the Program Office WBS. Linked to the CWBS, contractors submit a Software Resources Data Report (SRDR), with a companion data dictionary, at the beginning and the end of each contracted activity or delivery order. In the SRDR, contractors list the COTS

products used during the course of their activities. In the data dictionary, contractors describe how they interpreted the SRDR reporting requirements and how they defined their terms.

DoD online data sources available for cost analysts are the Defense Acquisition Management Information Retrieval (DAMIR) and Defense Acquisition Automated Cost Information System (DACIMS) systems. The DAMIR site, at http://www.acq.osd.mil/damir/, is sponsored by OUSD/AT&L and Acquisition Resources and Analysis (ARA). DAMIR holds the following: Selected Acquisition Reports (SARs), MAIS Acquisition Reports (MARs), Defense Acquisition Executive Summary (DAES) reports, MAIS Quarterly Reports (MQRs), Acquisition Program Baselines (APBs), and Earned Value Management System (EVMS) data. As program baselines change, new ABPs are added to DAMIR. The DACIMS site, at https://ders.dcarc.pae.osd.mil/DACIMS/Pages/Index.asp, is sponsored by the Defense Cost and Resource Center (DCARC), formerly known as the Contractor Cost Data Report (CCDR) Project Office (CCDR-PO). DCARC belongs to the Office of the Secretary of Defense (OSD) Cost Assessment and Program Evaluation (CAPE) organization. DACIMS contains CCDR reports, SRDR reports, and companion documentation such as Contractor Work Breakdown Structure (CWBS) descriptions and SRDR data dictionaries. SRDR submissions follow instructions in the Data Item Description (DID) from DI-MGMT-81739 for the Initial Developer Report and in the DID from DI-MGMT-81740

for the Final Developer Report.

[http://dcarc.pae.osd.mil/Policy/CSDR/csdrReporting.aspx 2009]

Status Reporting

DoD guidelines for collecting EVMS data provided the foundation for American National Standards Institute/Electronic Industries Alliance Standard ANSI/EIA-748. Contractor EVMS reports link funds allocated and expended with work performed, based on the CWBS. As the contractor progresses, the government gets periodic reports, explicitly relating BCWP, ACWP, BCWS, and Estimate at Completion (EAC). With these metrics, the contractor and the government can compute the cost and schedule variance along with other performance indices using standard EVMS formulas. [DCMA 2006]

The two main WBS structures relevant to DoD cost analysis are the program's WBS and the CWBS. MIL-STD-881-A as of July 30, 2005, provides guidance for program WBS. In the Air Force Cost Analysis Agency, WBS sub-levels correspond to the three main appropriation categories: Research and Development (R&D), Procurement (PROC), and Operations and Support (O&S). Level 1 for the entire program scope; Level 2 for major subordinate elements such as R&D (including system engineering), PROC, and O&S; and Level 3 for elements below Level 2 such as software. "Within the scope of the WBS, the contractor has the flexibility to use the work breakdown elements to support on-going management activities. These may include EVM, cost estimating, and managing contract funds." [MIL-HDBK-881A 2005] The WBS organizes cost reporting in the Contract

Funds Status Report (CFSR), the Contractor Cost Data Report (CCDR), and the Contract Performance Report (CPR). The WBS and CWBS establish interrelationships between funding requirements, cost reports, and EVM data.

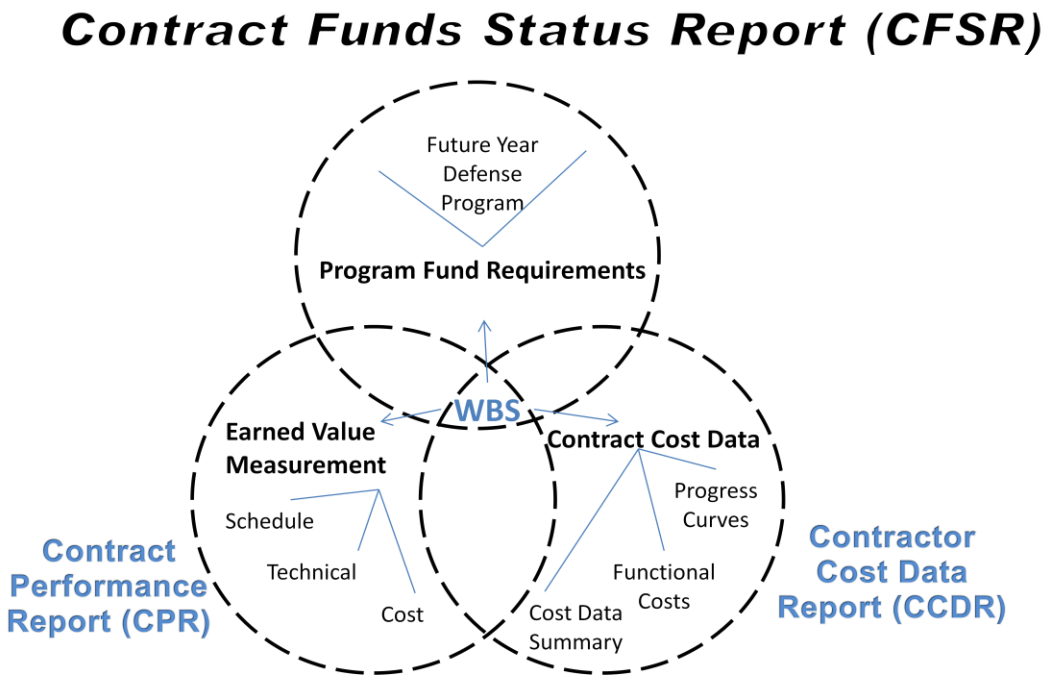

Figure 5: WBS support of CFSR, CPR, and CCDR [Cloos 2006]

Software Resources Data Report (SRDR) records, a component of the CSDR along with Contractor Cost Data Report (CCDR) records, record programming languages used in software development with Off-the-Shelf (OTS), including Commercial-Off-The-Shelf (COTS) and Government-Off-The-Shelf (GOTS), or Government-Furnished-Software (GFS). Designated elements are available in these reports. A table of the data elements to report is in Appendix F.

There have been attempts to normalize the SRDR data for records denoting logical SLOC counts [Jensen and Dupaix 2008], to generalize productivity factors and CER equations in SRDR data by platforms or contractors [Popp 2008], and to evaluate CCDR data for cost and schedule overruns [Selby, Hafen, Mink, Nicol, Flowe, Lile and Gold 2007]. SRDRs are based on historical program execution and provide various means to partition the data sets for analysis. One way to partition is by application type. Application types detailed in the SRDR reporting instructions come to a total of seventeen, with two to seventeen sub-types, adding up to a grand total of 119. In other literature, application types vary from two (recoded as low or high) up to thirty-two, if four operating environments combine with eight application domains. [Angelis, Stamelos and Morisio 2001;Gaylek, Long, Bell, Hsu and Larso 2004; Long and Lucas 1996] Although MIL-HDBK-881A allows for eight common elements for hardware and software combinations of weapon systems, SRDR allows for seventeen. The eight MIL-HDBK-881A elements are: (a) Aircraft System; (b) Electronic / Automated Software System; (c) Missile System; (d) Ordnance System; (e) Sea System; (f) Space System; (g) Surface Vehicle system; and (h) Unmanned Air Vehicle. Most SRDR entries are under (b) Electronic / Automated Software System. The MIL-HDBK-881A and SRDR instructions underscore the lack of standardized application types in software reporting.

"The lack of definitions for application type creates an open interpretation by both the data submitter and the user". [Jensen and Dupaix 2008] This lack of definition and standardization affects the analysis of software performance and of software estimation. "Software development practitioners do not have a chance of operating with socially refined and unified size measures." [Gencel and Demirors 2008] Size metrics, complexity metrics, productivity metrics, and progress metrics, aside from the internationally standardized EVMS metrics, are up for interpretation by the data submitter and the user. Although the SRDR submissions include data dictionaries, the quality and usefulness of these dictionaries rely on the producer's skill and the user's interpretation.

In addition to the lack of commonality in application types, it is common for fields to be missing data. Missing data can be treated by imputation or by removal of the record. Techniques such as listwise deletion [removing an entire column of data having missing values], mean imputation [filling in missing values with the mean for that variable], and hot-deck imputation [filling in missing values from another observation using closest distance or Mean Absolute Deviation] are three common imputation techniques. [Strike, El Eman and Madhavji 2001] Methods should match the data. Recently, the following imputation techniques have been explored: decision tree single imputation, k-Nearest neighbor single imputation, mean or mode single imputation, expectation-maximization single imputation, expectation-maximization multiple imputation, 'fractioning' cases, and surrogate variable splitting. [Twala, Cartwright and Shepperd 2005] In the Software Da-

tabase established in 1983, out of approximately 2600 data points, only 318 data points reported effort and schedule. [Long and Lucas 1996]

For the SRDR data from 2003 to 2008, there were five programs at CMMI Level 4 and eight at CMMI Level 5. All the CMMI Level 4 programs reported EVM metrics. There were two contractors at CMMI Level 5 with missing EVM metrics, omitted from this data analysis, per my dissertation committee's direction.

Data Profile

Of the thirty-four records representing the data set of five programs reporting CMMI Level 4 in the final SRDR, twenty-five belonged to a single program. Of the thirty records representing the data set of eight programs reporting CMMI Level 5 in the final SRDR, eight belonged to a single program.

CMMI Level 4 development processes were reported as follows: twenty-five, waterfall; five, spiral; two, Rapid Application Development (RAD); and one, Incremental. CMMI Level 5 development processes were more varied: ten reported using the waterfall development process; nine, iterative; six, spiral; and the remainder were Modified Rational Unified Process (RUP), Incremental, and Evolutionary.

CMMI Level 4 records had the same or similar primary programming languages: C, C/C++, C++, and Ansi C. CMMI Level 5 records had varied primary programming lan-

guages. Contractors at CMMI Level 5 used C, C++, C#, Ada, Ada-95, and Java as their primary language. Use of other languages in addition to a primary language was common for both CMMI level data sets.

3. Research Methodology

To determine whether the recently collected DoD data fulfills its intended purpose to improve software cost estimating, my research followed a step-by-step process. Data analysis, including statistical visualization of the relationships between initial estimated software parameters and final effort hours, focused on highly correlated initial parameters to final effort hours to derive DoD cost estimating relationships (CERs) using ex post facto analysis of secondary data sets. Software task records, for the two highest maturity rated contractors at CMMI Levels 4 and 5, along with the Earned Value program records contain potential independent variables with final software effort hours as the dependent variable. After CERs were derived using ordinary least squares (OLS) regression, statistical visualization of the residuals ascertained their distribution. For each CER, I computed accuracy metrics considered 'standard' in SCE.

The method of linear and log-linear regression is "a mathematical optimization technique used to find the 'best linear fit' to a set of data" where either the "error is assumed to be a fixed, additive value", normally distributed about the CER linear fit or "a multiplicative error term" for '$Y_j = f(x_j, \beta)\varepsilon_j$ for $j = 1,\ldots n$'. [SAF 2007]

Statistical tests check the validity of the assumptions. These assumptions are the sample represents the population, the predicted dependent variables have normal distributions with identical standard deviations about the regression line, and the predicted dependent variables are independent of each other. [Sanders 1990] In multivariate regression, for each independent variable, the t-test can check whether that variable adds value. The null hypothesis is the variable does not add value or the slope of the variable's coefficient is zero. If the slope is zero, the variable may be redundant. While the t-test measures the effect of single independent variables, the partial F-test measures the effect of sets of independent variables, and the F-test measures all independent variables (i.e., with the null hypothesis that all the slopes equal zero). Since residuals are deviations from the fitted values, the process of identifying data structures, generating a distribution 'fit', and examining residuals is statistical analysis. The residual-fit spread plot compares the "spread of the fitted value with the spread of the residuals…Data that are skewed toward large values occur commonly". [Cleveland 1993] Monotone spread means spread increases with location or higher values have more spread.

Skewed values and monotone spread complicate data analysis and are typically found in software cost data, including the DoD data. After transforming a data set to handle outliers, formal statistical rules allow the user to measure the fit on the both the transformed and original data scales. [Lurie 2006] If a prediction uses a transformed scale, statistical procedures can handle retransformation bias. [Lurie 2006] When the CER has a multiplicative form, such as '$a*Var1^{b}*Var2^{c}*\ldots*\varepsilon$', transforming it using logarithms makes it

linear, and the unit space error term follows a lognormal distribution. "Log-linear models are in a very common and distinct class of non-linear relationships that are rendered linear when transformed into log-space." [SAF 2007]

I created three distinct and non-overlapping application area subsets to categorize the data within each CMMI level: system, support, and user. Descriptions of these application areas follow:

- Signal processing, operating system augmentation, missile computers, flight control, radar control and identification, and space payload represent system applications.

- Mission application, mission planning, and test programs represent support applications.

- Display and control, simulators, external communications, information management, network management, and architecture systems represent user-interaction applications.

By hypothesizing and probing relationships, this ex post facto analysis offers "simulate[d] experimental procedures by matching subjects who have with those who have not received some 'natural manipulation' on factors that might have been relevant before the time of the study…in the comparison…(matched after the fact, rather than at the outset, hence ex post facto)". [Neale and Liebert 1986] Three research hypotheses were:

- H_1: Early prediction of software effort hours is possible using CERs based on data available without knowing contractor in advance
- H_2: Splitting out data by application areas and by process maturity level improves CER accuracy
- H_3: EVMS metrics as independent variables improve CER accurcy

The following figure summarizes possible relationships in the data sets for effort:

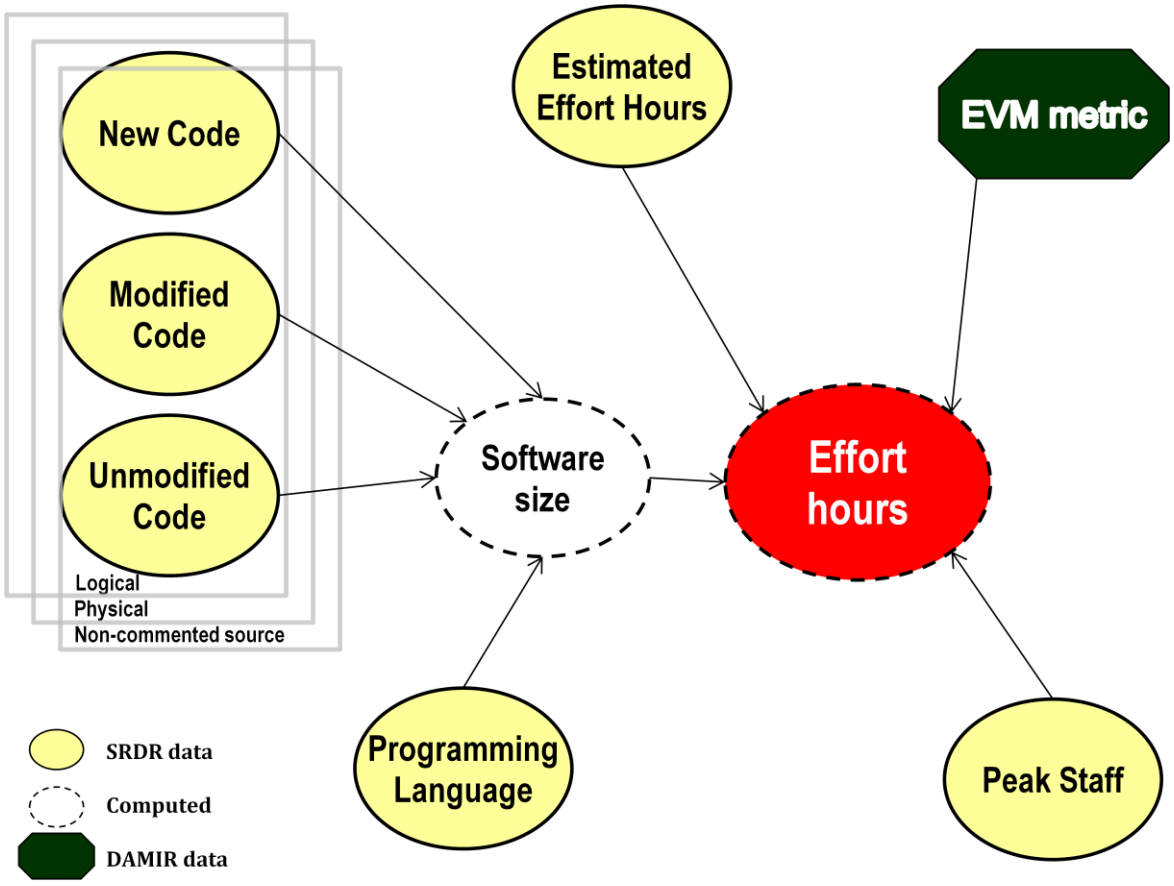

Figure 6: Methodology

I analyzed these variables using Pearson and Spearman Rank correlation matrices to determine relationships among the data sets. I identified potential independent variables using correlation. I generated regression CER equations and checked the returned values. If 'Adjusted R2' was greater than 0.55, I checked whether the addition of EVMS metrics improved the CER accuracy and whether the residuals behaved as they should, using statistical visualization in the OSS tool at http://www.r-project.org/ called The R Project for Statistical Computing. I calculated the Mean Magnitude of Relative Error (MMRE), the average of the Magnitude of Relative Error (MRE) for each record, and PRED(25), the number of records with MRE ≤ 0.25, along with PRED(30), the number of records with MRE ≤ 0.30, for each CER generated.

4. Research Results

Dr. Daniel Carr's R scripts from his courses and advice from the George Mason Statistical Consulting Center steered my statistical analyses. [Carr 2007; Sutton 2010] Of the Earned Value metrics investigated, I chose Schedule Performance Index (SPI) and the Cost Performance Index (CPI), with values from -1.0 to +1.0 across programs. Possible independent variables were peak staff, estimated hours, software size, software requirements, number of COTS programs, and experience level percentage (i.e., percent of staff that was highly experienced, nominally experienced, and entry-level).

Software size measures used different scales. As software counting type, such as physical, logical, and non-commented source, affects resulting size, the conversion ratios proposed by NASA to derive logical SLOC from physical (and related non-commented) SLOC by primary programming and secondary language was used to normalize the initial code sizes as much as possible. [http://software.gsfc.nasa.gov/docs/QSM-class/Day%201-Cost/04a-Size.ppt 2009] Combining programming languages affects software size and resulting effort to create and modify interconnected modules. [Vouk 1984]

CMMI Level 4 had two programs with one language, twenty-nine programs with two languages, and three programs with more than two languages; and CMMI Level 5 had eleven programs with one language, seventeen with two languages, and two programs with more than two languages. Software size counts from an automatic counter are not necessarily stable; multiple runs produce different counts. [Reifer 2009] COTS package code does not count in calculations of software size. Counted unmodified code should only be existent code from prior industry software efforts. Effective source lines of code formulas vary by the percentages assigned to modified and unmodified lines of code. The lowest possible software size count assigns zero percent to modified and unmodified lines; the largest possible assigns one hundred percent to modified and unmodified. [Gallo, Koza, Holzman, and Hardin 2008] To bound software size, I used the lowest possible software size and the highest possible, called new source lines of code in thousands (KNEW) and total source lines of code in thousands (KSLOC).

The code count used in this dissertation was Logical Statements (LS) from Non-commented source lines of code, Physical, or other. CMMI Level 5 started with fifteen LS, four physical LOC, ten non-commented source LOC, and one effective LOC. CMMI Level 4 started with eight LS and twenty-six physical LOC. Translation to LS used tables from http://software.gsfc.nasa.gov/docs/QSM-class/Day%201-Cost/04a-Size.ppt (Table 2-3) and the 24th International Forum on COCOMO and Systems/Software Cost Modeling, in November 2009, at http://csse.usc.edu/csse/event/2009/COCOMO/presentations/Workshop%20Summary%2

0-%20Metrics%20Unification%20&%20Productivity%20Domains.ppt, both accessed on February 28, 2010.

For multivariate regression to work well, the data should be in similar measurement units, scaled correspondingly. [Carr 2007] Therefore, software sizes and effort hours divided by one thousand provided similar measurement units to the other data. To avoid multi-collinearity, Pearson correlation analysis tested for linear relationships and Spearman rank correlation analysis tested for monotone relationships. [Sutton 2010] These correlations by CMMI levels are shown below, where EstHrsK stands for Estimated Hours in Thousands (K), EstPstaff stands for Estimated Peak Staff, KNEW or EstInitNewCodeK stands for thousands of estimated new source lines of code, and KSLOC stands for thousands of estimated lines of source lines of code.

Table 4-1: Pearson Correlation for both CMMI levels

PEARSON - LEVEL 5	EstHrsK	EstPstaff	KNEW	KSLOC	ActHrsK
EstHrsK	1	0.61	0.42	0.25	0.65
EstPstaff		1	0.40	0.65	0.79
KNEW			1	0.37	0.51
KSLOC				1	0.71
ActHrsK					1

PEARSON LEVEL 4	EstHrsK	EstPstaff	KNEW	KSLOC	ActHrsK
EstHrsK	1	0.96	0.94	0.99	0.97
EstPstaff		1	0.92	0.96	0.98
KNEW			1	0.92	0.93
KSLOC				1	0.96
ActHrsK					1

The preceding table shows much stronger relationships for CMMI Level 4 among the independent variables and between the possible independent variables and the dependent variable than for CMMI Level 5. Peak staff for both levels correlates linearly to final ef-

fort hours and to KSLOC. KSLOC for both also correlates to final effort hours as does estimated hours. KNEW has a weaker correlation to final effort hours for CMMI Level 5 than for CMMI Level 4.

Table 4-2: Spearman Rank Correlation for both CMMI levels

SPEARMAN - LEVEL 5	EstHrsK	EstPstaff	KNEW	KSLOC	ActHrsK
EstHrsK	1	0.66	0.76	0.65	0.79
EstPstaff		1	0.71	0.79	0.82
KNEW			1	0.78	0.77
KSLOC				1	0.78
ActHrsK					1

SPEARMAN LEVEL 4	EstHrsK	EstPstaff	KNEW	KSLOC	ActHrsK
EstHrsK	1	0.57	0.63	0.55	0.93
EstPstaff		1	0.50	0.71	0.65
KNEW			1	0.74	0.71
KSLOC				1	0.66
ActHrsK					1

For both levels, there is a monotone relationship between final, actual effort hours and initial, estimated effort hours. For both, peak staff relates monotonically also. The monotone relationship is stronger for KNEW to final effort hours than for KSLOC to final effort hours in CMMI Level 4 and stronger for KSLOC than KNEW in CMMI Level 5.

Table 4-3: Pearson Correlation for the User Application Area Subsets

PEARSON - LEVEL 5 USER APPLICATION AREA	EstHrsK	EstPstaff	KNEW	KSLOC	ActHrsK
EstHrsK	1	0.51	0.14	0.01	0.72
EstPstaff		1	0.32	0.50	0.90
KNEW			1	0.36	0.52
KSLOC				1	0.51
ActHrsK					1

PEARSON - LEVEL 4 USER APPLICATION AREA	EstHrsK	EstPstaff	KNEW	KSLOC	ActHrsK
EstHrsK	1	0.98	1.00	0.99	0.98
EstPstaff		1	0.97	0.97	1.00
KNEW			1	1.00	0.97
KSLOC				1	0.97
ActHrsK					1

For the user application subsets, peak staff has the highest correlation to final effort hours and estimated hours the next highest correlation to final effort hours for both CMMI levels. For CMMI Level 4, the software sizes and estimated hours relate linearly to initial, estimated peak staff and to final effort hours. For CMMI Level 5, software size does not strongly correlate linearly to initial, estimated peak staff or to final effort hours.

Table 4-4: Spearman Rank Correlation of User Application Area Subsets

SPEARMAN - LEVEL 5 USER APPLICATION AREA	EstHrsK	EstPstaff	KNEW	KSLOC	ActHrsK
EstHrsK	1	0.33	0.68	0.15	0.62
EstPstaff		1	0.45	0.55	0.85
KNEW			1	0.27	0.77
KSLOC				1	0.65
ActHrsK					1

SPEARMAN - LEVEL 4 USER APPLICATION AREA	EstHrsK	EstPstaff	KNEW	KSLOC	ActHrsK
EstHrsK	1	0.91	0.98	0.98	1.00
EstPstaff		1	0.92	0.92	0.91
KNEW			1	1.00	0.98
KSLOC				1	0.98
ActHrsK					1

Monotonic relationships exist for the user application area's peak staff to final effort hours, for software size to final effort hours, and for estimated hours to final effort hours in both CMMI levels. For CMMI Level 4, the strongest correlation is between estimated and final effort hours and between new and total source lines of code, indicating that KNEW equals KSLOC. For CMMI Level 5, the weak correlation between KNEW and KSLOC indicates that KNEW is not equal to KSLOC. Estimated hours monotonically correlate to KNEW in both CMMI levels.

Table 4-5: Pearson Correlation of System Application Area Subsets

PEARSON - LEVEL 5 SYSTEM APPLICATION AREA	EstHrsK	EstPstaff	KNEW	KSLOC	ActHrsK
EstHrsK	1	0.51	0.14	0.01	0.72
EstPstaff		1	0.32	0.50	0.90
KNEW			1	0.36	0.52
KSLOC				1	0.51
ActHrsK					1

PEARSON - LEVEL 4 SYSTEM APPLICATION AREA	EstHrsK	EstPstaff	KNEW	KSLOC	ActHrsK
EstHrsK	1	0.98	1.00	0.99	0.98
EstPstaff		1	0.97	0.97	1.00
KNEW			1	1.00	0.97
KSLOC				1	0.97
ActHrsK					1

The highest correlation is between peak staff and final effort hours in both CMMI levels. The next highest correlation is between estimated hours and final effort hours. Although software size correlates strongly to final effort hours in CMMI Level 4, a weaker correlation exists in CMMI Level 5 between software size and final effort hours. The perfect correlation in CMMI Level 4 between KNEW and KSLOC indicates KNEW equals KSLOC whereas the weak correlation in CMMI Level 5 indicates KNEW is not equal to KSLOC.

Table 4-6: Spearman Rank Correlation of System Application Area Subsets

SPEARMAN - LEVEL 5 SYSTEM APPLICATION AREA	EstHrsK	EstPstaff	KNEW	KSLOC	ActHrsK
EstHrsK	1	0.33	0.68	0.15	0.62
EstPstaff		1	0.45	0.55	0.85
KNEW			1	0.27	0.77
KSLOC				1	0.65
ActHrsK					1

SPEARMAN - LEVEL 4 SYSTEM APPLICATION AREA	EstHrsK	EstPstaff	KNEW	KSLOC	ActHrsK
EstHrsK	1	0.91	0.98	0.98	1.00
EstPstaff		1	0.92	0.92	0.91
KNEW			1	1.00	0.98
KSLOC				1	0.98
ActHrsK					1

The strongest monotone correlation is peak staff to final effort hours in CMMI Level 5 whereas the strongest monotone correlation is estimated initial hours to final effort hours in CMMI Level 4. Software size correlates to final effort hours in both CMMI levels. Estimated hours correlate to KNEW in both CMMI levels, stronger in CMMI Level 4 than in CMMI Level 5.

Table 4-7: Pearson Correlation of Support Application Area Subsets

PEARSON - LEVEL 5 SUPPORT APPLICATION AREA	EstHrsK	EstPstaff	KNEW	KSLOC	ActHrsK
EstHrsK	1	0.58	0.74	0.01	0.49
EstPstaff		1	0.17	0.67	0.83
KNEW			1	0.05	0.37
KSLOC				1	0.86
ActHrsK					1

PEARSON - LEVEL 4 SUPPORT APPLICATION AREA	EstHrsK	EstPstaff	KNEW	KSLOC	ActHrsK
EstHrsK	1	-0.22	0.31	0.31	0.77
EstPstaff		1	0.60	0.60	0.30
KNEW			1	1.00	0.74
KSLOC				1	0.74
ActHrsK					1

The support application area subsets correlations differ from those in other subsets. Although peak staff again correlates to final effort hours in CMMI Level 5, it does not correlate in CMMI Level 4. Estimated hours correlate to final hours in CMMI Level 4. Software size of KSLOC correlates in both CMMI levels to final effort hours but only KNEW in CMMI Level 4 correlates to final effort hours. KNEW correlates to estimated hours in CMMI Level 5 but not in CMMI Level 4. The correlation between KNEW and KSLOC in CMMI Level 4 is 1.00, indicating KNEW equals KSLOC; not the case in CMMI Level 5 where the correlation between KNEW and KSLOC is 0.05. Software size correlates to peak staff in CMMI Level 4, but only KSLOC correlates to peak staff in CMMI Level 5.

Table 4-8: Spearman Rank Correlation of Support Application Area Subsets

SPEARMAN - LEVEL 5 SUPPORT APPLICATION AREA	EstHrsK	EstPstaff	KNEW	KSLOC	ActHrsK
EstHrsK	1	0.75	0.75	0.61	0.64
EstPstaff		1	0.46	0.82	0.93
KNEW			1	0.46	0.39
KSLOC				1	0.96
ActHrsK					1

SPEARMAN - LEVEL 4 SUPPORT APPLICATION AREA	EstHrsK	EstPstaff	KNEW	KSLOC	ActHrsK
EstHrsK	1	0.27	0.33	0.33	0.91
EstPstaff		1	0.08	0.08	0.27
KNEW			1	1.00	0.59
KSLOC				1	0.59
ActHrsK					1

The Spearman Rank correlation in CMMI Level 5 has the highest value between total software size (KSLOC) and final effort hours with the second highest between peak staff and final effort hours. KSLOC is highly correlated monotonically to peak staff in CMMI Level 5. Estimated hours correlate to peak staff and to KNEW in CMMI Level 5. The support application area for CMMI Level 4 has the highest Spearman Rank correlated value between estimated hours and final hours.

Depending on the CMMI level, there is a linear relationship, a monotone one, or both. To test the first hypothesis, I used R to visualize the relationship between each of the initial, estimated independent variables and the final effort hours.

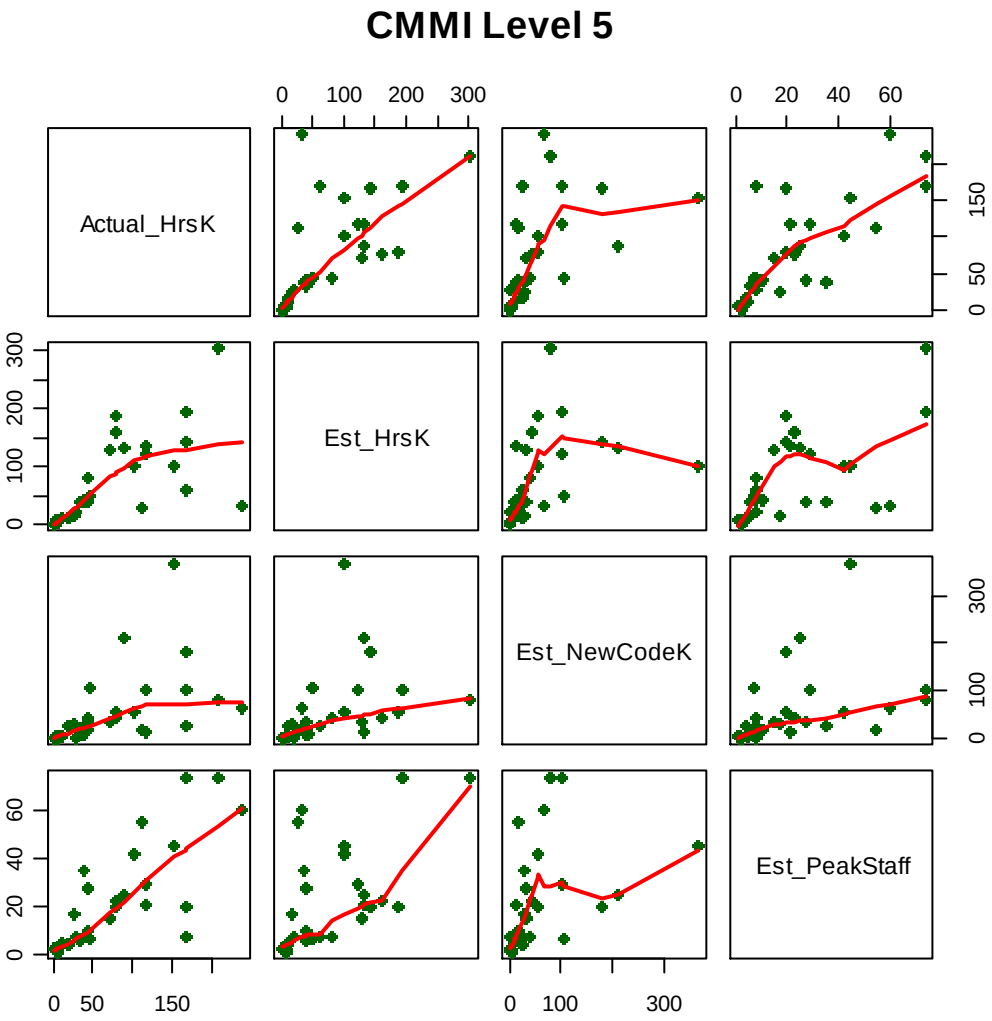

Figure 7: R Pairs Plot for CMMI Level 5

The only software size shown on the preceding pairs plot is new source lines of code representing the lowest possible software size. The variable name Est_NewCodeK represents the same values as are in the variable name KNEW. Variable values listed on the plots have x-axis and y-axis as indicated by the variable names. For example, the top

right graph has estimated peak staff on the x-axis and final effort hours on the y-axis with axes reversed on the bottom left graph.

Table 4-9: CMMI Level 5 Single Variable Results: KNEW

CMMI Level 5 Independent Variable	SPI?	CPI?	Scale	Intercept	Coefficient	Coefficient (SPI)	Coefficient (CPI)	Adjusted R2	p-value	Residuals versus Fitted	QQ Plot?	Scale-Location Plot?	Leverage over 0.5 Cook's Distance?	Original Scale: MMRE = MAPE	Original Scale: PRED(25)	Original Scale: PRED(30)
KNEW	No	No	Orig.	51.67	0.43	N/A	N/A	0.23	0.00		N/A	N/A	N/A	N/A	N/A	N/A
KNEW	No	No	Log	1.65	0.68	N/A	N/A	0.70	0.00	Shallow inverted U shape	Follows line from (-1,2)	Mostly straight	No	0.98	0.20	0.27
KNEW	Yes	Yes	Log	471.89	0.48	-377.90	-55.29	0.77	0.00	Shallow inverted U shape	One outlier on line from (-2,3)	Straight	No	0.75	0.30	0.37
KNEW	Yes	No	Log	410.99	0.47	-372.66	N/A	0.77	0.00	Shallow inverted U shape	One outlier on line from (-2,3)	Mostly straight	No	0.77	0.33	0.43
KNEW	No	Yes	Log	99.89	0.44	N/A	-47.75	0.70	0.00	Shallow inverted U shape	Follows line from (-1,2)	Mostly straight	No	0.99	0.17	0.20

The accuracy for the logarithmically transformed variable representing the logical statements of new code to predict the transformed effort hours had high MMRE values and low PRED values. The accuracy measures improved the most when both EVMS added (also transformed logarithmically in keeping with the multiplicative nature of the CER),

lowering MMRE and raising PRED. Singly, SPI improved accuracy and raised the adjusted R2 value, increasing the residual range to (-2, +3) due to an outlier.

Table 4-10: CMMI Level 5 Single Variable Results: Estimated Hours

CMMI Level 5 Independent Variable	SPI?	CPI?	Scale	Intercept	Coefficient	Coefficient (SPI)	Coefficient (CPI)	Adjusted R2	p-value	Residuals versus Fitted	QQ Plot?	Scale-Location Plot?	Leverage over 0.5 Cook's Distance?	Original Scale: MMRE = MAPE	Original Scale: PRED(25)	Original Scale: PRED(30)
Est. Hrs (K)	No	No	Orig.	31.35	0.59	N/A	N/A	0.40	0.00	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Est. Hrs (K)	Yes	Yes	Orig.	235.61	0.65	-415.06	190.29	0.41	0.00	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Est. Hrs (K)	Yes	No	Orig.	403.20	0.63	-386.18	N/A	0.40	0.00	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Est. Hrs (K)	No	Yes	Orig.	-150.40	0.61	N/A	176.99	0.40	0.00	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Est. Hrs (K)	No	No	Log	0.47	0.89	N/A	N/A	0.79	0.00	Mostly straight	3 pts off line (-1,3)	Mostly straight	No	0.69	0.20	0.27
Est. Hrs (K)	Yes	Yes	Log	-0.03	0.94	-8.76	2.25	0.82	0.00	Mostly straight	Follows line from (-1,3)	Mostly straight	No	0.62	0.27	0.37
Est. Hrs (K)	Yes	No	Log	0.08	0.92	-8.50	N/A	0.82	0.00	Straight	One top outlier, line (-1,3)	Mostly straight	No	0.63	0.30	0.40
Est. Hrs (K)	No	Yes	Log	0.38	0.90	N/A	1.98	0.79	0.00	Mostly straight	3 points offline (-1,+3)	Mostly straight	No	0.69	0.23	0.27

The EVMS metrics did not improve accuracy for logarithmically transformed initially estimated effort hours as the independent variable. There was a slight improvement in MMRE by adding EVMS metrics either singly or jointly with a worsening in PRED. Us-

ing estimated hours, in the original units, to predict final effort hours did not yield an adjusted R2 over 0.41, indicating a lack of goodness of fit.

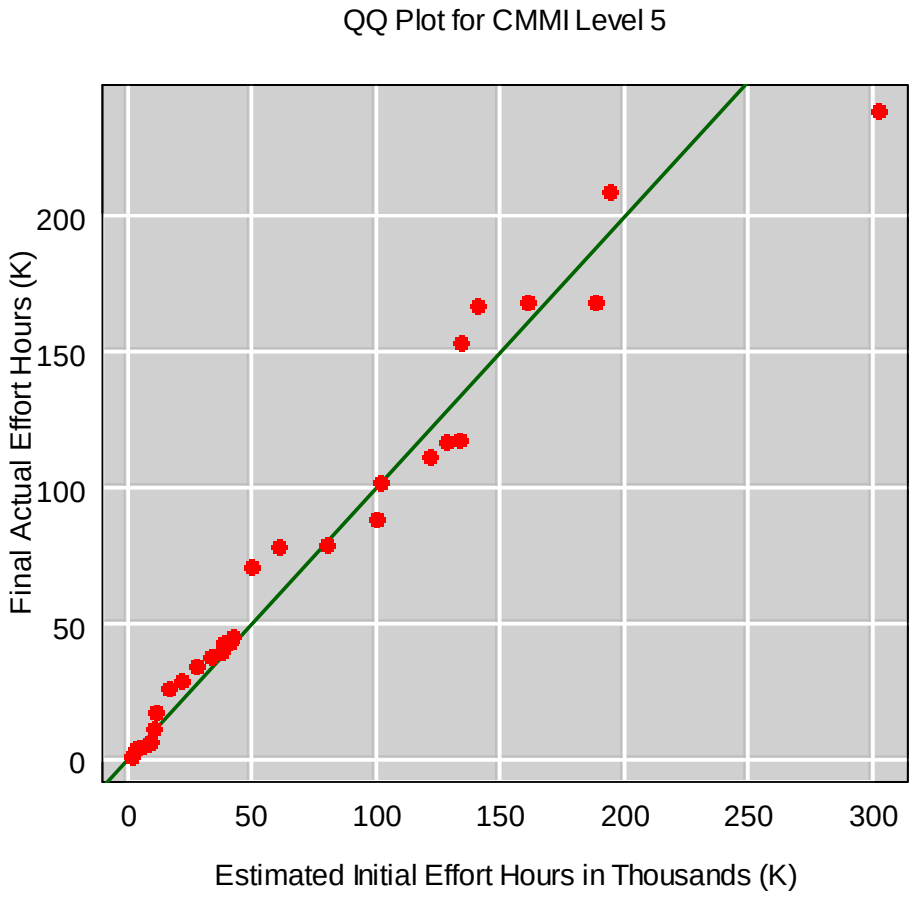

Figure 8: CMMI Level 5 QQ Plot of Estimated and Final Hours

Although this QQ plot shows a linear relationship between the estimated hours and final effort hours, the preceding table shows the results of R's lm command for linear model.

For estimated effort hours up to 150, the one-to-one correspondence line shows data mostly following the line. Beyond estimated effort hours of 150, the deviations from the line become more pronounced.

Table 4-11: CMMI Level 5 Single Variable Results: Peak Staff

CMMI Level 5 Independent Variable	SPI?	CPI?	Scale	Intercept	Coefficient	Coefficient (SPI)	Coefficient (CPI)	Adjusted R2	p-value	Residuals versus Fitted	QQ Plot?	Scale-Location Plot?	Leverage over 0.5 Cook's Distance?	Original Scale: MMRE = MAPE	Original Scale: PRED(25)	Original Scale: PRED(30)
Peak Staff	No	No	Orig.	22.10	2.44	N/A	N/A	0.60	0.00	Straight	Follow s line from (-1,3)	Mostly straight	No	1.53	0.47	0.50
Peak Staff	Yes	Yes	Orig.	-376.88	2.57	158.08	238.25	0.63	0.00	Straight	One top outlier, line (-1,4)	Straight	No	1.59	0.33	0.47
Peak Staff	Yes	No	Orig.	-142.34	2.45	169.13	N/A	0.59	0.00	Straight	3 pts off line (-1,3)	Mostly straight	#12	1.54	0.47	0.57
Peak Staff	No	Yes	Orig.	-224.91	2.55	N/A	239.94	0.64	0.00	Straight	One top outlier, line (-1,4)	Straight	#18	1.55	0.37	0.37

The accuracy metrics for peak staff as the independent variable are not within the 'accepted industry standard' of 0.25 or below. The values above 1.5 are higher than the accepted standard and adding EVMS metrics worsen MMRE. PRED(30) is improved only

by adding the EVMS metric SPI. Joint EVMS metrics or CPI alone worsen all the accuracy metrics, particularly PRED(25) and PRED(30).

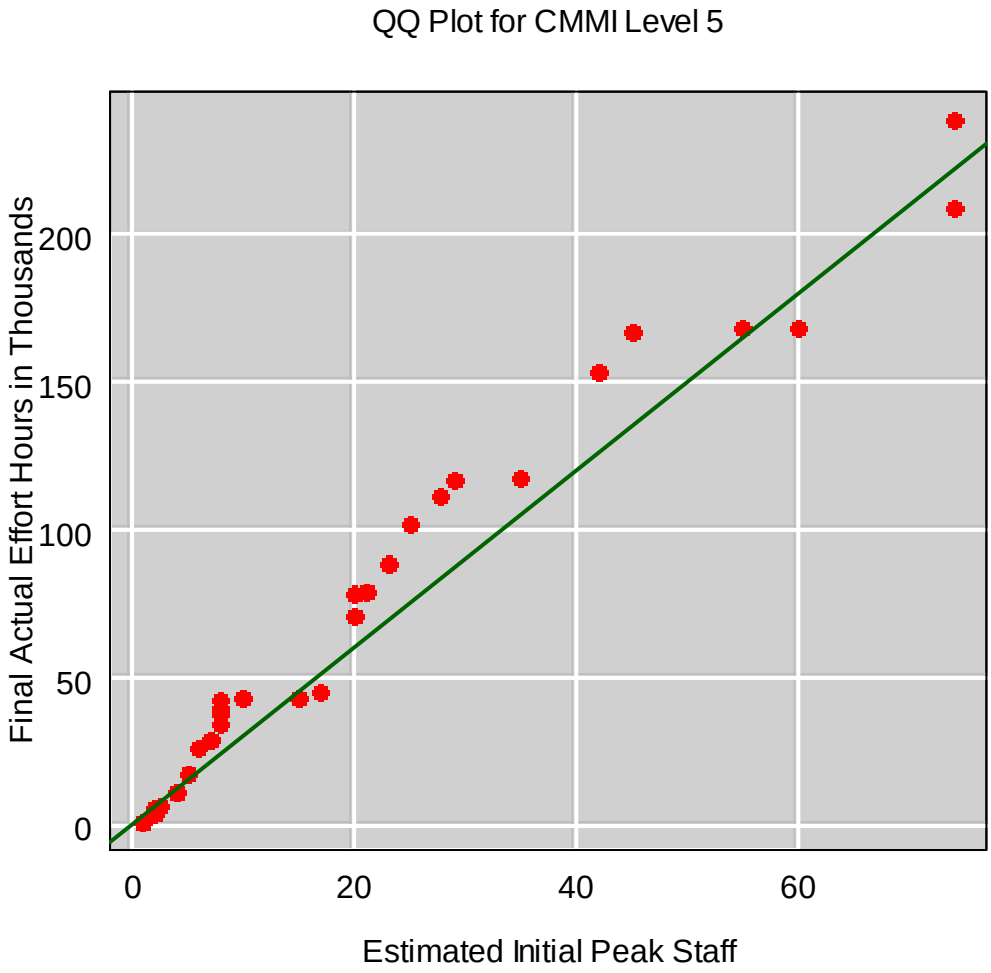

Figure 9: CMMI Level 5 QQ Plot of Peak Staff and Final Effort Hours

The reference line in the preceding figure is one-to-three. With 60 estimated initial peak staff, final effort hours approach 180, falling slightly short. Estimated initial peak staff has more than three times the final effort hours for values of peak staff between twenty

and fifty. The QQ plot shows the similarity of the data distributions. The reference line and the white grid lines provide visual orientation for the reader. [Cleveland 1993]

Table 4-12: CMMI Level 5 Single Variable Results: Peak Staff, transformed

CMMI Level 5 Independent Variable	SPI?	CPI?	Scale	Intercept	Coefficient	Coefficient (SPI)	Coefficient (CPI)	Adjusted R2	p-value	Residuals versus Fitted	QQ Plot?	Scale-Location Plot?	Leverage over 0.5 Cook's Distance?	Original Scale: MMRE = MAPE	Original Scale: PRED(25)	Original Scale: PRED(30)
Peak Staff	No	No	Log	1.32	0.96	N/A	N/A	0.74	0.00	Mostly straight	Top outlier (#12), bottom outlier (#7), line (-3,+3)	Straight	No	0.87	0.37	0.40
Peak Staff	Yes	Yes	Log	1.24	0.98	0.36	2.24	0.74	0.00	Mostly straight	Tw o outliers, line from (-2,+3)	Mostly straight	No	0.86	0.40	0.50
Peak Staff	Yes	No	Log	1.33	0.96	0.45	N/A	0.74	0.00	Mostly straight	Tw o outliers, line from (-2,+3)	Straight	No	0.88	0.40	0.40
Peak Staff	No	Yes	Log	1.23	0.98	N/A	2.24	0.75	0.00	Mostly straight	Tw o outliers, line from (-2,+3)	Mostly straight	No	0.85	0.40	0.50

Accuracy metrics for transformed peak staff predicting transformed effort hours do not improve with the addition of EVMS metrics. Accuracy improves slightly for CPI alone or for joint EVMS metrics. Adding SPI does not change the accuracy metrics.

Table 4-13: CMMI Level 5 Single Variable Results: KSLOC

CMMI Level 5 Independent Variable	SPI?	CPI?	Scale	Intercept	Coefficient	Coefficient (SPI)	Coefficient (CPI)	Adjusted R2	p-value	Residuals versus Fitted	QQ Plot?	Scale-Location Plot?	Leverage over 0.5 Cook's Distance?	Original Scale: MMRE = MAPE	Original Scale: PRED(25)	Original Scale: PRED(30)
KSLOC	No	No	Orig.	41.45	0.13	N/A	N/A	0.48	0.00	N/A	N/A	N/A	N/A	N/A	N/A	N/A
KSLOC	No	No	Log	1.27	0.57	N/A	N/A	0.74	0.00	Mostly straight	Top outlier (#12) & bottom outlier (#4) from (-3,2)	Mostly straight	No	0.63	0.37	0.43
KSLOC	Yes	Yes	Log	1.10	0.58	-3.82	1.08	0.73	0.00	Mostly straight	Top outlier (#12) & bottom outlier (#4) from (-2,3)	Mostly straight	No	0.61	0.40	0.43
KSLOC	Yes	No	Log	1.14	0.58	-3.74	N/A	0.74	0.00	Mostly straight	Top outlier (#12) & bottom outlier (#4) from (-3,3)	Straight	No	0.63	0.30	0.43
KSLOC	No	Yes	Log	1.24	0.57	N/A	1.00	0.74	0.00	Mostly straight	Top outlier (#12) & bottom outlier (#4) from (-3,3)	Mostly straight	No	0.62	0.40	0.43

Untransformed KSLOC had a low adjusted R2 when used to predict untransformed effort hours. The adjusted R2 value improved for the logarithmically transformed KSLOC, EVMS, and effort hours with p-values < 0.001 indicating statistically significant models.

CPI improved MMRE and PRED(25) slightly with no effect on PRED(30); other EVMS metrics did not improve accuracy.

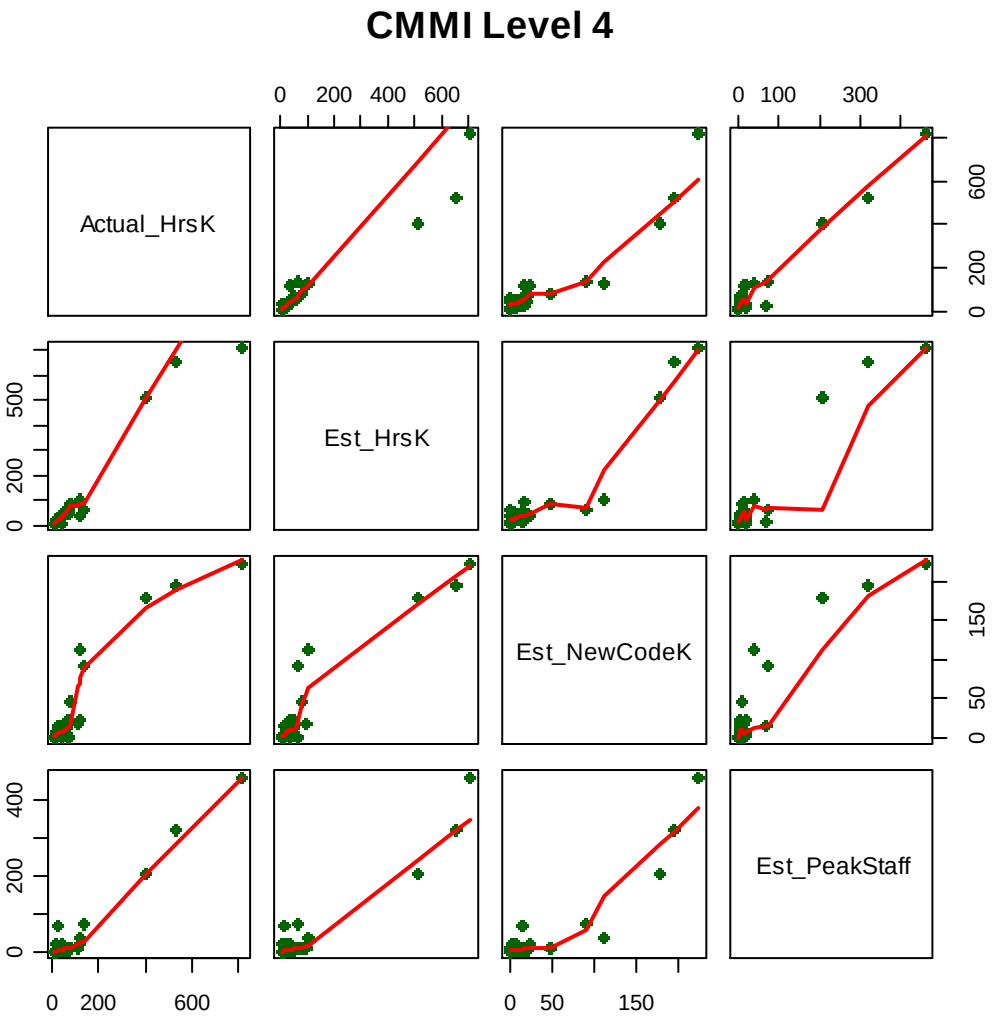

Figure 10: CMMI Level 4 Pairs Plot

As for CMMI Level 5, the only software size shown on the preceding pairs plot is the new source lines of code where the variable shown as Est_NewCodeK is the same as

KNEW. Another similarity to CMMI Level 5 is the preponderance of low values in the data sets for the variables. The few higher valued data's effect on CERs diminishes when the data undergo a logarithmic transformation.

Table 4-14: CMMI Level 4 Single Variable Results: KNEW

CMMI Level 4 Independent Variable	Scale	Intercept	Coefficient	Coefficient (SPI)	Coefficient (CPI)	Adjusted R2	p-value	Residuals versus Fitted	QQ Plot?	Scale-Location Plot?	Leverage over 0.5 Cook's Distance?	Original Scale: MMRE	Original Scale: PRED(25)	Original Scale: PRED(30)
KNEW	Original	11.74	2.66	N/A	N/A	0.86	0.00	Shallow V shape	Thick tails (-2,4)	Slopes up ~60° then up ~30°	#30	0.42	0.47	0.50
KNEW	Original	-501.45	2.68	1047.14	-502.79	0.87	0.00	Shallow V shape	Thick tails (-2,4)	Jagged	#30, #26, #34	0.69	0.12	0.12
KNEW	Original	-871.58	2.51	929.06	N/A	0.86	0.00	Shallow V shape	Thick tails (-2,4)	Slopes up ~60° then flat	#30, #26	0.47	0.24	0.24
KNEW	Original	458.72	2.84	N/A	-465.89	0.87	0.00	Shallow V shape	Thick tails (-2,4)	Jagged	#30, #34	0.64	0.15	0.15
KNEW	Log	2.86	0.48	N/A	N/A	0.64	0.00	Straight, dips, goes up 45˚	Thick tails (-2,+2)	Straight until end (up 45˚)	No	0.77	0.21	0.26
KNEW	Log	3.62	0.37	6.89	6.33	0.68	0.00	Mostly Straight	Thick tails (-3,+2)	Straight until end (up 45˚)	No	0.69	0.32	0.32
KNEW	Log	3.40	0.44	9.62	N/A	0.65	0.00	Straight, curves up at end	Thick tails (-3,+2)	Straight until end (up 45˚)	Yes, one	0.76	0.21	0.26
KNEW	Log	3.26	0.39	N/A	6.87	0.68	0.00	Straight, curves up at end	Thick tails (-2,+2)	Straight until end (up 45˚)	No	0.69	0.32	0.38

Accuracy for the logarithmically transformed variable representing the logical statements of new code had high MMRE values and low PRED values. Accuracy improved the most with the addition of the EVMS metric, CPI, also transformed in keeping with the multiplicative nature of the CER.

Table 4-15: CMMI Level 4 Single Variable Results: Estimated Hours

CMMI Level 4 Independent Variable	Scale	Intercept	Coefficient	Coefficient (SPI)	Coefficient (CPI)	Adjusted R2	p-value	Residuals versus Fitted	QQ Plot?	Scale-Location Plot?	Leverage over 0.5 Cook's Distance?	Original Scale: MMRE	Original Scale: PRED(25)	Original Scale: PRED(30)
Est. Hrs (K)	Original	10.95	0.94	N/A	N/A	0.94	0.00	Inclines upward (~45˚)	Thick tails (-2,+4)	Inclines upwards (~50˚)	#28, #29, #30	0.28	0.68	0.71
Est. Hrs (K)	Original	-879.32	0.88	697.05	236.04	0.95	0.00	Shallow incline up (~15˚)	Thick tails (-2,+6)	Inclines upwards (~50˚)	#28, #29, #31	0.24	0.76	0.82
Est. Hrs (K)	Original	-791.45	0.89	843.64	N/A	0.95	0.00	Steep incline up (~85˚)	Thick tails (-2,+4)	J' shape to start, inclines up (~50˚)	#28, #29, #32	0.28	0.68	0.76
Est. Hrs (K)	Original	-269.42	-269.42	N/A	291.15	0.95	0.00	Inclines down-wards (~45˚)	Thick tails (-2,+4)	Jagged to start, then inclines up (~50˚)	#28, #29, #33	0.23	0.71	0.74
Est. Hrs (K)	Log	0.64	0.88	N/A	N/A	0.87	0.00	Straight	Thick top tail (-1,+4)	Slopes down 45˚ then flat	#31	0.36	0.35	0.50
Est. Hrs (K)	Log	1.62	0.75	8.19	4.25	0.91	0.00	Straight	Thick top tail (-2,+4)	Mostly straight	#31	0.27	0.12	0.18
Est. Hrs (K)	Log	1.36	0.81	10.73	N/A	0.89	0.00	Slightly inverted V	Thick top tail (-2,+4)	Mostly straight	#31	0.31	0.53	0.59
Est. Hrs (K)	Log	1.17	0.78	N/A	5.18	0.90	0.00	Straight	Thick top tail (-2,+4)	Mostly straight	#31	0.30	0.50	0.53

Estimated effort hours predict actual, final effort hours for CMMI Level 4 data. The accuracy approaches or exceeds industry-standard values with or without the addition of EVMS metrics. For transformed hours, SPI improves PRED (30) accuracy more than CPI; the MMRE is improved for SPI or CPI alone. The most improvement occurs for

SPI with CPI. For the original scale, MMRE accuracy is lowest for CPI but PRED (30) is highest with both EVMS metrics. SPI improves PRED (30) accuracy more than CPI.

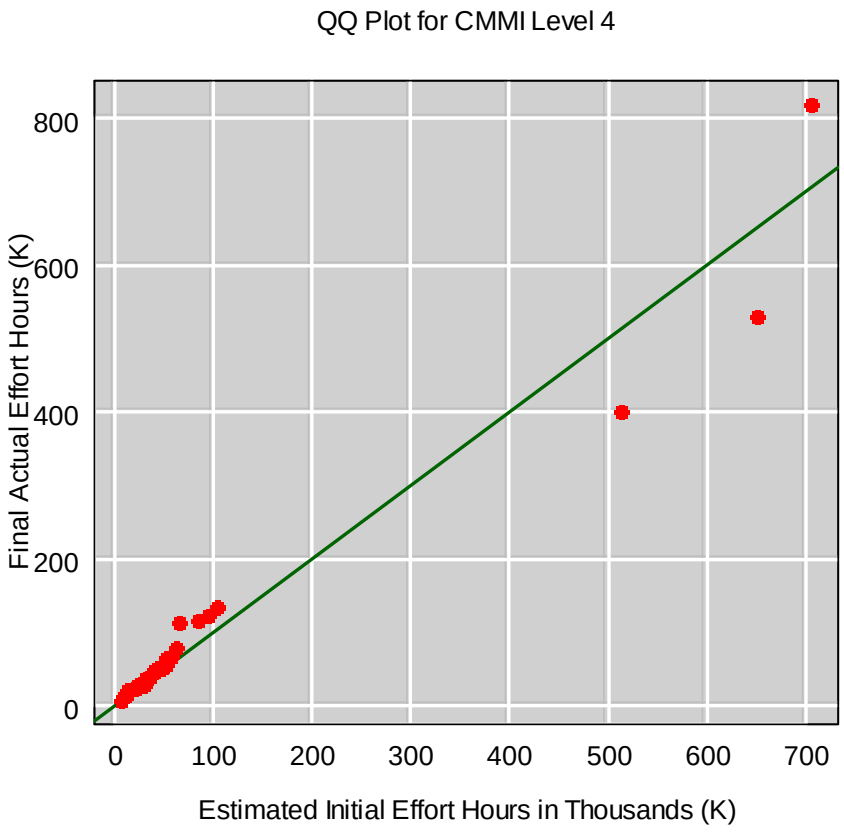

Figure 11: CMMI Level 4 QQ Plot of Estimated and Final Effort Hours

The preceding plot of estimated initial effort hours to final effort hours highlights the concentration of low values between one hundred to two hundred thousand hours.

Table 4-16: CMMI Level 4 Single Variable Results: Peak Staff

CMMI Level 4 Independent Variable	Scale	Intercept	Coefficient	Coefficient (SPI)	Coefficient (CPI)	Adjusted R2	p-value	Residuals versus Fitted	QQ Plot?	Scale-Location Plot?	Leverage over 0.5 Cook's Distance?	Original Scale: MMRE	Original Scale: PRED(25)	Original Scale: PRED(30)
Peak Staff	Original	25.29	1.67	N/A	N/A	0.96	0.00	Mostly Straight	Thick tails (-4,+2)	Mostly straight	#30	0.62	0.53	0.59
Peak Staff	Original	-879.32	0.88	697.05	236.04	0.95	0.00	Mostly Straight	Thick tails (-2,+6)	Jagged to start, then goes up (~50˚)	#28, #29, #30	0.24	0.76	0.82
Peak Staff	Original	-791.45	0.89	843.64	N/A	0.95	0.00	Steep incline up (~85˚)	Thick tails (-2,+4)	J' shape to start, inclines up (~50˚)	#28, #29, #31	0.28	0.68	0.76
Peak Staff	Original	-269.42	0.91	N/A	291.15	0.95	0.00	Inclines dow n-w ards (~45˚)	Thick tails (-2,+4)	Jagged to start, then goes up (~50˚)	#28, #29, #32	0.23	0.71	0.74
Peak Staff	Log	2.27	0.64	N/A	N/A	0.68	0.00	Mostly Straight, inclines up ~20˚	Outliers (#26 and #32) from (-3,+1)	Straight	Yes, one (#26)	0.75	0.47	0.53
Peak Staff	Log	1.49	0.73	-12.01	0.50	0.67	0.00	Mostly Straight, inclines up ~20˚	Outliers (#26 and #32) from (-3,+1)	Mostly straight	Yes, one (#26)	0.73	0.38	0.47
Peak Staff	Log	1.43	0.74	-12.33	N/A	0.68	0.00	Mostly straight	Outliers (#26 and #32) from (-3,+1)	Mostly straight	Yes, one (#26)	0.72	0.38	0.44
Peak Staff	Log	2.42	0.60	N/A	1.78	0.67	0.00	Mostly Straight, inclines up ~20˚	Outliers (#26, #32, #15) from (-3,+1)	Mostly straight	No	0.74	0.38	0.47

EVMS metrics improve accuracy for CERs with untransformed peak staff as the independent variable to predict effort hours. CPI on MMRE and joint EVMS metrics on PRED (30) boost accuracy metrics. SPI improves PRED (30) more than CPI. However,

EVMS metrics increase the number of 'leverage' points in the CER as denoted by Cook's Distance. [Draper and Smith 1998] EVMS metrics do not significantly affect accuracy for transformed peak staff, as the independent variable. MMRE improves slightly with any EVMS metric. PRED gets worse by adding EVMS.

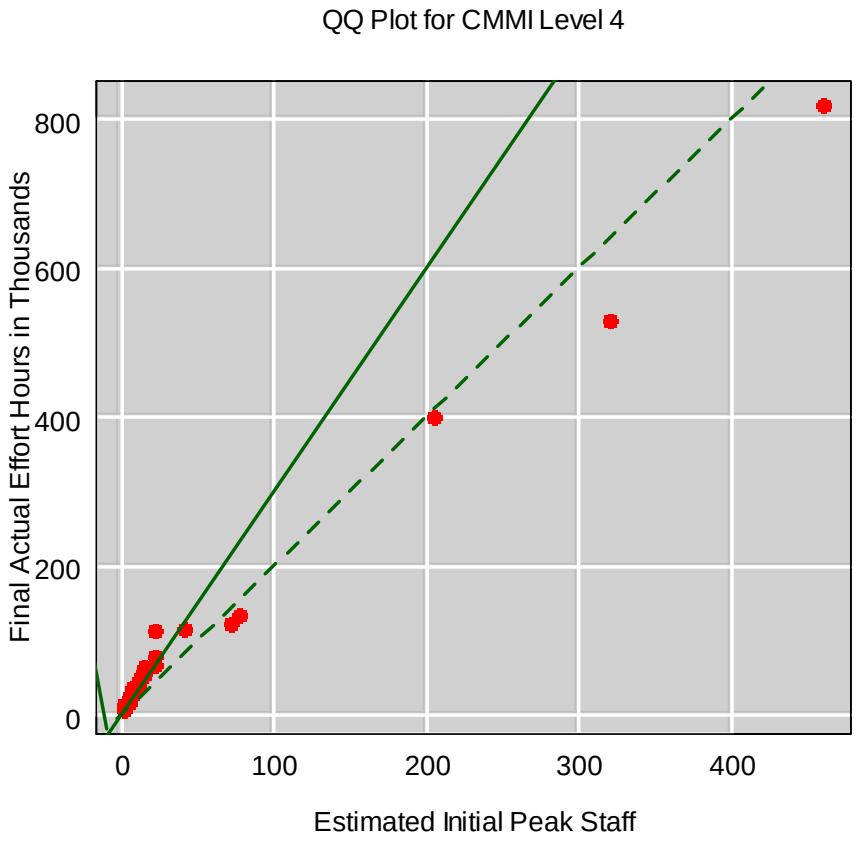

Figure 12: CMMI Level 4 QQ Plot of Peak Staff and Final Effort Hours

The solid reference line is one-to-three and the dashed reference line is one-to-two. The QQ plot shows the similarity of the distributions of peak staff and final, actual effort hours for CMMI Level 4 data.

Table 4-17: CMMI Level 4 Single Variable Results: KSLOC

CMMI Level 4 Independent Variable	Scale	Intercept	Coefficient	Coefficient (SPI)	Coefficient (CPI)	Adjusted R2	p-value	Residuals versus Fitted	QQ Plot?	Scale-Location Plot?	Leverage over 0.5 Cook's Distance?	Original Scale: MMRE	Original Scale: PRED(25	Original Scale: PRED(30
KSLOC	Original	43.27	0.19	N/A	N/A	0.92	0.00	Jagged	3 outliers on (-4,+4)	90˚ line to ~50˚ line	Jagged	0.68	0.38	0.41
KSLOC	Original	-1053.00	0.18	782.20	362.70	0.93	0.00	Mostly straight	4 outliers on (-4,+4)	90˚ line to ~45˚ line	#30, #26, #28	0.56	0.41	0.56
KSLOC	Original	-935.75	0.18	1027.42	N/A	0.92	0.00	Mostly straight	3 outliers on (-2,+4)	90˚ line to ~40˚ line	#30, #26, #29	0.69	0.38	0.47
KSLOC	Original	-137.35	0.19	N/A	429.04	0.93	0.00	Mostly straight	Tw o outliers on (-4,+4)	Jagged	#30, #28	0.53	0.44	0.47
KSLOC	Log	2.84	0.37	N/A	N/A	0.65	0.00	Mostly straight	3 outliers on (-3,+3)	Mostly straight	None	0.74	0.26	0.35
KSLOC	Log	3.57	0.29	7.35	5.17	0.67	0.00	Mostly straight	4 outliers on (-3,+2)	Mostly straight	One (#26)	0.64	0.38	0.44
KSLOC	Log	3.35	0.34	9.16	N/A	0.66	0.00	Mostly straight	3 outliers on (-3,+2)	Mostly straight	One (#26)	0.66	0.38	0.41
KSLOC	Log	3.19	0.31	N/A	5.67	0.67	0.00	Mostly straight	3 outliers on (-3,+2)	Mostly straight	None	0.69	0.38	0.41

With the untransformed KSLOC as the independent variable predicting software effort hours, although the adjusted R2 values are high with low p-values, indicating a well fitting CER, outlying points in the residuals expand the scale beyond the expected (-2, +2) region for a normal distribution. Due to the outlier values, and to the poor accuracy me-

tric, the CER with untransformed KSLOC is not robust. Transforming the software size and the final effort hours improves residuals' behavior. Accuracy improves with the addition of EVMS metrics, but in none of the CERs with KSLOC is the accuracy near industry-standard levels.

I created three application areas (i.e., System, Support, or User) to sub-divided the data within each CMMI level. Each record could belong to only one application category. Signal processing, operating system augmentation, missile computers, flight control applications, radar control and identification applications, and space payload applications were system application areas. Mission planning, mission applications, and testing programs were support application areas. Display and control applications, simulators, external communications, information management, network management, and architecture applications were user application areas for user-interactions and decisions. The following table shows the distribution of application types by CMMI level.

Table 4-18: Generated Application Types by CMMI Levels

Application Types	**CMMI Level 5**	**CMMI Level 4**
Support	7	11
System	14	11
User	9	12
Total	**30**	**34**

The results in the application areas can be compared to the results for the entire CMMI data sets to determine whether the third hypothesis, which postulates that application area subsets will improve CER accuracy, applies.

Table 4-19: CMMI Level 5 Application Area Subsets

CMMI Level 5 Subsets	# *Rec*	Scale	Intercept	Coefficient	Adjusted R2	p-value	Original Scale: MMRE = MAPE	Original Scale: PRED(25)	Original Scale: PRED(30)
User: KNEW	9	Original	69.83	0.29	0.16	0.15	N/A	N/A	N/A
System: KNEW	14	Original	21.65	1.65	0.26	0.04	N/A	N/A	N/A
Support: KNEW	7	Original	61.28	0.43	-0.03	0.41	N/A	N/A	N/A
User: KNEW	9	Log	2.06	0.58	0.52	0.02	N/A	N/A	N/A
System: KNEW	14	Log	1.63	0.77	0.65	0.00	1.41	0.29	0.29
Support: KNEW	7	Log	0.79	0.81	0.79	0.00	16.76	0.00	0.00
User: Est. Hours (K)	9	Original	39.55	0.48	0.44	0.03	N/A	N/A	N/A
System: Est. Hours (K)	14	Original	15.91	0.70	0.45	0.00	N/A	N/A	N/A
Support: Est. Hours (K)	7	Original	56.42	0.59	0.09	0.26	N/A	N/A	N/A
User: Est. Hours (K)	9	Log	1.52	0.64	0.50	0.02	9.10	0.11	0.22
System: Est. Hours (K)	14	Log	0.20	0.95	0.85	0.00	4.67	0.00	0.00
Support: Est. Hours (K)	7	Log	0.47	0.94	0.75	0.01	36.52	0.00	0.00
User: Peak Staff	9	Original	18.56	2.41	0.78	0.00	0.44	0.67	0.67
System: Peak Staff	14	Original	26.72	1.75	0.14	0.10	N/A	N/A	N/A
Support: Peak Staff	7	Original	32.09	2.61	0.63	0.02	3.77	0.43	0.43
User: Peak Staff	9	Log	1.55	0.87	0.76	0.00	9.41	0.22	0.22
System: Peak Staff	14	Log	1.61	0.85	0.72	0.00	3.18	0.07	0.07
Support: Peak Staff	7	Log	0.67	1.20	0.75	0.01	51.30	0.14	0.14
User: KSLOC	9	Original	48.63	0.11	0.15	0.16	N/A	N/A	N/A
System: KSLOC	14	Original	59.91	0.03	0.02	0.26	N/A	N/A	N/A
Support: KSLOC	7	Original	46.36	0.14	0.68	0.01	4.58	0.29	0.29
User: KSLOC	9	Log	1.47	0.50	0.19	0.13	N/A	N/A	N/A
System: KSLOC	14	Log	1.69	0.48	0.60	0.00	0.86	0.26	0.37
Support: KSLOC	7	Log	0.69	0.68	0.97	0.00	0.36	0.29	0.43

For all CERs predicting software effort hours, accuracy improves for the user application subset with untransformed peak staff as the independent variable and MMRE improves for the support application area where logarithmically transformed KSLOC (software size) is the independent variable. Outside of these cases, accuracy does not improve after separating the data into application area subsets for this set of CMMI Level 5 data.

Table 4-20: CMMI Level 4 Application Area Subsets

CMMI Level 4 Subsets	# Rec	Scale	Intercept	Coefficient	Adjusted R2	p-value	Original Scale: MMRE = MAPE	Original Scale: PRED(25)	Original Scale: PRED(30)
User: KNEW	12	Original	4.83	2.97	0.94	0.00	0.27	0.50	0.67
System: KNEW	11	Original	40.07	0.80	0.49	0.01	N/A	N/A	N/A
Support: KNEW	11	Original	42.29	0.98	0.49	0.01	0.38	0.64	0.64
User: KNEW	12	Log	2.64	0.62	0.89	0.00	0.52	0.17	0.17
System: KNEW	11	Log	3.26	0.27	0.21	0.09	N/A	N/A	N/A
Support: KNEW	11	Log	3.59	0.17	0.06	0.23	N/A	N/A	N/A
User: Est. Hrs (K)	12	Original	1.41	0.95	0.95	0.00	0.14	1.00	1.00
System: Est. Hrs (K)	11	Original	12.26	1.02	0.49	0.01	N/A	N/A	N/A
Support: Est. Hrs (K)	11	Original	5.80	1.11	0.55	0.01	0.23	0.73	0.82
User: Est. Hrs (K)	12	Log	0.32	0.94	0.99	0.00	0.13	0.83	0.83
System: Est. Hrs (K)	11	Log	1.76	0.58	0.33	0.04	N/A	N/A	N/A
Support: Est. Hrs (K)	11	Log	0.92	0.81	0.69	0.00	2.01	0.00	0.00
User: Peak Staff	12	Original	20.26	1.71	1.00	0.00	0.43	0.42	0.50
System: Peak Staff	11	Original	14.02	2.39	0.36	0.03	N/A	N/A	N/A
Support: Peak Staff	11	Original	54.49	0.41	-0.01	0.37	N/A	N/A	N/A
User: Peak Staff	12	Log	2.13	0.74	0.96	0.00	0.30	0.58	0.58
System: Peak Staff	11	Log	2.46	0.50	0.07	0.22	N/A	N/A	N/A
Support: Peak Staff	11	Log	3.77	0.09	-0.09	0.68	N/A	N/A	N/A
User: KSLOC	12	Original	21.92	0.20	0.94	0.00	0.42	0.33	0.33
System : KSLOC	11	Original	42.60	0.24	0.00	0.35	N/A	N/A	N/A
Support: KSLOC	11	Original	42.29	0.98	0.49	0.01	0.38	0.64	0.64
User: KSLOC	12	Log	-27.31	69.63	0.76	0.00	1.04	0.00	0.00
System: KSLOC	11	Log	3.43	0.11	-0.09	0.66	N/A	N/A	N/A
Support: KSLOC	11	Log	3.59	0.17	0.06	0.23	N/A	N/A	N/A

For CMMI Level 4, accuracy improves for several of the independent variables. In particular, untransformed KNEW approaches industry standard levels for the User application area. Untransformed KNEW's accuracy metrics for the Support application area are better than they are for the full set. Transformed KNEW's accuracy, like that of the full set, does not meet industry standards.

Estimated Hours (in thousands) – transformed and untransformed exceeds industry standard levels for the User application area. Untransformed estimated hours' accuracy metrics exceed industry standard levels as well as for the Support application area. However, for the transformed estimated hours, accuracy is worse for both metrics for the Support application area subset than it is for the full set. Untransformed peak staff's MMRE metric is better for the User application area than it is for the full set; the full set's PRED (30) is better than the User application area. Transformed peak staff as the independent variable has MMRE approaching the industry standard level for the User application area. Untransformed KSLOC had a better MMRE than the full set for the User application area; the full set's PRED (30) is higher than the User application area. For the Support application area, untransformed KSLOC has better accuracy than the untransformed KSLOC for the whole set. The whole set's transformed KSLOC as an independent variable to predict final software effort hours has a MMRE equal to 0.74 which is better than the User application area subset's MMRE of 1.04. Additionally, the PRED (30) of zero

for the User application area is much worse than the PRED (30) of 0.35 for the full set. Thus, partitioning by application area does not universally improve accuracy.

Based on the data, single parameters (namely, initially estimated effort hours, peak staff, and software size) in a CER can predict final, actual effort hours. These CERs rely on data with large amounts of small-values and small amounts of large-values. Logarithmic transformations of estimated hours and software size, particularly for CMMI Level 5 data, were necessary to achieve valid results. Accuracy metrics used are not the only applicable ones. Threats to validity of the results found are numerous.

The small data sets threaten internal validity. Data was taken from DoD databases with paired initial and final records and from the two highest CMMI levels and CERs from CMMI Level 4 data are not identical to CERs from CMMI Level 5 data. The DoD data set has varied amounts of data from contractors with varied CMMI levels and other CERs use only final reported data parameters. "External validity threats arise when experimenters draw incorrect inferences from the sample data." [Creswell 2009] The timing of this dissertation allowed 34 records at CMMI Level 4 and 30 at CMMI Level 5 - records subject to the business rules for entry into the DoD SRDR repository. Future records in the SRDR repository may or may not be similar to the records analyzed for this dissertation.

Construct validity is manifest in "inadequate definitions and measures of variables." [Creswell 2009] A definition of new code occurred in most, but not all, of the SRDR da-

ta dictionaries. Modification levels of 25%, 30%, and 50% were common as demarcations when to count modified code as new code. The same definition occurred most frequently for CMMI Level 4 data, possibly due to the set of programs being half the size of the CMMI Level 5 set. With lines of code, the type of line could be logical, physical, non-commented source, and others. Lines of code converted to logical statements employed the most current translation tables available. These measures of software threaten construct validity more than the measures for any of the other variables.

5. Conclusions

A significant contribution is combining and correlating initial data with final effort to estimate software effort for DoD contractors with a high CMMI levels. Most of the historical studies used final data to predict final effort. A few others used expert's predicted effort to compare to final effort. Using initial parameters to predict final effort allows DoD to expect, as is common in SCE, that the future will progress like the past.

Another significant contribution is the correlation analysis of the input parameters for the higher CMMI levels of the contractors. By assuming high quality based on CMMI rating, this dissertation focused on contractors with ongoing quality improvement efforts. By comparing the top two CMMI levels separately, documented characterizations can baseline future studies by CMMI level.

CMMI Level 4 records are more consistent and interrelated than Level 5 records in this data set. By using similar languages and traditional development processes, the CERs residurals were better behaved and fit better for Level 4. CMMI Level 5 contractors seemed to concentrate on systems development whereas Level 4's efforts spread evenly between the three application subsets. Characteristics of future data sets will need to be compared to the characteristics described in this dissertation.

The integration of Earned Value Management System (EVMS) metrics, for this subset of the data, as an input parameter for software cost estimating is new in DoD. Discovering whether this metric has utility in DoD software cost estimation is a contribution to knowledge. Further, documenting the effect this metric has on the software estimating performance may provide a baseline for future studies.

After creating and testing CERs using initial, expected parameters to predict actual, reported final effort hours, normalizing between physical and logical line counts by languages suggested by Lum, Baker and Hihn [2008], there are differences in the proposed CERs by CMMI levels. The contribution to knowledge is the act of explicitly considering these input variables in relation to each other as well as in relation to the output variable of software effort hours.

As this data set expands, other DoD researchers can take my results as a baseline to see how the relationships among the variables change affecting CERs. Separate studies by high CMMI levels of software project parameters and metrics will expand our knowledge of the impact of these levels on the production of software and of the impact of software environments. As these data sets grow and change, some parameters need to stay constant, in particular those focused on in this dissertation, to allow future comparisons. Creating and using standards in measurement of software parameters will alleviate threats to construct validity in the future.

6. Future Research

As the data on software developers achieving top CMMI levels expand, continued study of CERs by CMMI level may become common in SCE. As there are two representations for CMMI (staged and continuous), it may be useful to investigate whether the representation type makes any difference in the CERs. Repeating this method with the same data not separated by CMMI level would create CERs to compare to those in this dissertation. Repeating this method with similar or different data sets by process maturity level may validate the utility of CMMI level partitioning. Either by process maturity level or not, different application area foci may be in order. Using application areas or environments such as ground, air, and space may provide useful CERs for future SCE. Different application area subsets may illuminate patterns in the data or show other statistical relationships among the variables of interest. Different application areas may produce different CER accuracy levels. Different accuracy measures applied to similar or different data sets would provide results comparisons.

As experimentation with EVMS's measures continue in SCE, other EVMS variables may prove more useful than SPI and CPI, used in this research. CV or SV values could be useful in CERs, either with other parameters or by themselves. These values could be divided by the contract value or by the number of effort hours, either estimated or final.

Comparing each of the initially estimated parameters to final parameters may shed light on SCE over- or under-estimation, particularly by viewing them by a category of interest. For example, initially estimated peak staff may relate differently to final peak staff at various CMMI levels. As the use of COTS expands, so should COTS measures. As measures evolve, new measures appear, or measures become standardized, particularly for software size, new sets of CERs will emerge. Experimenting with software development learning curves may lead to their use by one or more data set parameters. Use of similar methods to those used in this dissertation with different statistical tools would reveal the impact of tools. Use of different methods with the same statistical tools would reveal the impact of methods.

List of References

List of References

Abdel-Hamid T. and Madnick S., Lessons Learned from Modeling the Dynamics of Software Development, Communications of the ACM, Volume 32, Issue 12, December 1989, pp. 1426-1455

Abts C. M., Model Description: The COCOTS Extension of COCOMO II, University of Southern California (USC) Center for Software Engineering and Texas A&M University Working Paper, October 2002

Abts C. M., Extending the COCOMO II software cost model to estimate effort and schedule for software systems using commercial-off-the-shelf (COTS) software components: The COCOTS model, Dissertation, Department of Industrial and Systems Engineering, University of Southern California, 2004, 290 pages

Agarwal R., Kumar M., Yogesh M., Mallick S., Bharadwaj R. and Anantwar D., Estimating software projects, ACM SIGSOFT Software Engineering Notes, Volume 26, Issue 4, July 2001, pp. 60-67

Agrawal M. and Chari K., Software Effort, Quality, and Cycle Time: A Study of CMM Level 5 Projects, IEEE Transactions on Software Engineering, Volume 33, Number 3, March 2007, pp. 145-156

Ahmed F., Bouktif S., Serhani A. and Khalil I., Integrating Function Point Project Information for Improving the Accuracy of Effort Estimation, Proceedings of the 2nd International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2008), Volume 00, Valencia, Spain, September 29-October 4, 2008, pp. 193-198

Albert C. and Brownsword L., Evolutionary Process for Integrating COTS-Based Systems (EPIC): An Overview, Carnegie Mellon Software Engineering Institute COTS-Based Systems Initiative, Pittsburg, Pennsylvania, July 2002, 62 pages, http://www.sei.cmu.edu/library/abstracts/reports/02tr009.cfm, Accessed on October 4, 2009

Anda B., Benestad H. and Hove S., A Multiple Case Study of Software Effort Estimation based on Use Case Points, Proceedings of the IEEE International Symposium on Empiri-

cal Software Engineering (ISESE 2005), Noosa Heads, Australia, November 17-18, 2005, pp. 407-416

Andreou A., Papatheocharaous E. and Skouroumounis C., Evolving Conditional Value Sets of Cost Factors for Estimating Software Development Effort, Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007), Patras, Greece, October 29-31, 2007, pp. 165-172

Andreou A. and Papatheocharaous E., Software Cost Estimation using Fuzzy Decision Trees, Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008), L'Aquila, Italy, September 15-19, 2008, pp. 371-374

Angelis L., Stamelos I. and Morisio M., Building a Software Cost Estimation Model Based on Categorical Data, Proceedings of the 7th International Symposium on Software Metrics (METRICS 2001), London, April 4-6, 2001, pp. 4-15

Armel K. (Editor), The QSM Software Almanac, Application Development Series, 2006 IT Metrics Edition, Quantitative Software Management, Inc. (QSM), 2006, 106 pages

Ayala Martinez C., Systematic construction of goal-oriented COTS taxonomies, Technical University of Catalunya (Campus Nord 08034), Barcelona, Spain, February 2008, 243 pages, at http://www.tdx.cat/TDX-0428108-124137, Accessed on September 23, 2009

Bachman D., Single Point Adjustments: A New Definition with Examples (Tutorial), Acquisition Review Quarterly (ARQ), September 22, 2001, pp. 177-196, at http://www.dau.mil/pubs/arq/2001arq/Bachman.pdf, Accessed on January 4, 2009

Banker R., Davis G. and Slaughter S., Software Development Practices, Software Complexity, and Software Maintenance Performance: a Field Study, Management Science, Volume 44, Number 4, April 1998, pp. 433-450

Baskeles B., Turnhan B. and Bener A., Software Effort Estimation Using Machine Learning Methods, 22nd International Symposium on Computer and Information Sciences (ISCIS 2007), November 7-9 2007

Beims M. and Dabney J., Reliable Tailored-COTS via Independent Verification and Validation, NATO Research and Technology Organization (RTO), Information Systems Technology (IST) Symposium on Commercial-Off-the-Shelf Products in Defense Applications ("The Ruthless Pursuit of COTS"), RTO MP-48, Brussels, Belguim, April 3-5 2000, pp. 10:1-7, http://ftp.rta.nato.int/public//PubFulltext/RTO/MP/RTO-MP-48///MP-48-10.pdf, Accessed on September 23, 2009

Bennatan E. M., On Time Within Budget: Software Project Management Practices and Techniques, Third Edition, John Wiley and Sons, Inc., New York, 2000, 341 pages

Berlin S., Raz T., Glezer C. and Zviran M., Comparison of estimation methods of cost and duration in IT projects, Information and Software Technology, Elsevier, 2009, pp. 738-748

Bernstein P., Against the Gods: The Remarkable Story of Risk, John Wiley and Sons, Inc., New York, 1996, 383 pages

Boehm B., Abts C., Brown A. W., Chulani S., Clark B., Horowitz E., Madachy R., Reifer D. and Steece B., Software Cost Estimation with COCOMO II, Prentice Hall PTR, Upper Saddle River, New Jersey, 2000, 502 pages

Boehm B., Software Engineering Economics, Prentice Hall, Upper Saddle River, New Jersey, 1981, 767 pages

Boehm B., A view of 20th and 21st Century Software Engineering, Proceedings of the 28th International Conference on Software Engineering (ICSE 2006) Keynote Talks, Shanghai, China, May 20-28 2006, pp. 12-29

Boetticher G. D. and Lokhandwala N., Assessing the reliability of a human estimator, Proceedings of the 3rd international Workshop on Predictor Models in Software Engineering (PROMISE 2007), Minneapolis, Minnesota, May 20-26 2007, pp. 5-12

Bollinger T., Software in the year 2010, IT Professional, Volume 6, Issue 6, IEEE Educational Activities Department, Piscataway, New Jersey, November-December 2004, pp. 11-15

Bollinger T., The Interplay of Art and Science in Software, Computer, Volume 30, Number 10, October 1997, pp. 125-128

Booch G., Best of Booch, Cambridge University Press, Cambridge, United Kingdom, 1998, 236 pages

Book S. A., Briefing titled "The Morass of Software Costing", November 14, 2001, Email from Dr. Daniel Nussbaum (danussba@nps.edu), Naval Postgraduate School, sent on December 6, 2006

Braga P., Oliveira A. and Meira S., Software Effort Estimation using Machine Learning Techniques with Robust Confidence Intervals, Proceedings of the 7th International Conference on Hybrid Intelligent Systems (HIS 2007), Kaiserlautern, Germany, September 17-19, 2007, pp. 352-357

Briand L., El Emam K., Surmann D., Wieczorek I. and Maxwell K., An Assessment and Comparison of Common Software Estimation Modeling Techniques, Proceedings of the

21st International Conference on Software Engineering (ICSE 1999), Los Angeles, California, May 16-22, 1999, pp. 313-322

Briand L., El Emam K. and Bomarius F., COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking, and Risk Assessment, Proceedings of the 20th International Conference on Software Engineering Benchmarking, and Risk Assessment (ICSE 1998), Kyoto, Japan, April 19-25, 1998, pp. 390-399

Briand L., Langley T. and Wieczorek I., A Replicated Assessment and Comparison of Common Software Cost Modeling Techniques, Proceedings of the 22nd International Conference on Software Engineering (ICSE 2000), Limerick, Ireland, June 4-11, 2000, pp. 377-386

Brooks F., The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition, Addison Wesley Longman, Inc., 1995, 322 pages

Bryan J. and Locke E., Goal setting as a means of increasing motivation, Journal of Applied Psychology, Volume 51, Issue 3, June 1967, pp. 274-277

Bryant M., What Does Triangulation Do To A Cost Estimate? (Some Insights From Evaluating Cost Uncertainty), Air Force Cost Analysis Agency Working Paper, 2008, 12 pages

Cain J. and McCrindle R., An Investigation into the Effects of Code Coupling on Team Dynamics and Productivity, Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment (COMPSAC 2002), Oxford, England, August 26-29, 2002, pp. 907-913

Carr D., Class lectures and learning materials, George Mason University, Statistical Graphics and Data Exploration Course (CSI-773), Fall Semester 2007

Chand D. and Gowda R., An exploration of the impact of individual and group factors on programmer productivity, Proceedings of the 1993 ACM conference on Computer Science, Indianapolis, Indiana, February 16-18, 1993, pp. 338-345

Chiu N. and Huang S., The adjusted analogy-based software effort estimation based on similarity distances, Journal of Systems and Software, Volume 80, Number 4, April 2007, pp. 628-640

Chulani S., Santhanam P., Moore D., Leszkowicz G. and Davidson G., Deriving a Software Quality View from Customer Satisfaction and Service Data (2001), Proceedings of the 12th European Software Control and Metrics Conference (ESCOM 2001), Volume 1, London, April 2-4 2001, pp. 225-232,

http://www.escom.co.uk/conference2001/papers/chulani.pdf, Accessed on September 23, 2009

Clark B., The effects of software process maturity on software development effort, PhD Dissertation, University of Southern California, 1997, 163 pages

Cleveland W., The Elements of Graphing Data, Wadsworth Advanced Book Program, Bell Telephone Laboratories, Murray Hill, New Jersey, 1985, 323 pages

Cloos J., Cost Data: Government and Industry Lecture, George Mason University Graduate Course, Military Operations Research: Cost Analysis (OR 651), Alexandria, Virginia, February 2006

Cockburn A., Agile Software Development: The Cooperative Game, Second Edition, Pearson Education, Inc., Boston, Massachusetts, 2006, 467 pages

Conte S., Dunsmore H. and Shen V., Software Engineering Metrics and Models, Benjamin/Cummings Publishing Company, Inc., Menlo Park, California, 1986, 396 pages

Creswell J., Research Design: Qualitative Quantitative and Mixed Method Approaches, Third Edition, Sage Publications, Inc., Thousand Oaks, California, 2009, 296 pages

DACS - The Data & Analysis Center for Software, Software Tech News: New Directions in Software Estimation, October 2008, https://www.softwaretechnews.com/stn_view.php?stn_id=47, Accessed on October 27, 2008

Davis A. M., Two Hundred One Principles of Software Development, McGraw-Hill, Inc., New York, 1995, 240 pages

DCMA or Defense Contract Management Agency (DCMA/PID), Department of Defense Earned Value Management Implementation Guide, October 2006, 106 pages, http://guidebook.dcma.mil/79/EVMIG.doc, Accessed on December 2, 2008

de Barcelos Tronto I. F., da Silva J. D. S. and Sant'Anna N., Comparison of Artificial Neural Network and Regression Models in Software Effort Estimation, Proceedings of the International Joint Conference on Neural Networks, Orlando, Florida, August 12-17 2007, pp. 771-776

Defense Science Board, Office of the Undersecretary of Defense, Report of the Defense Science Board Task Force on Military Software, Defense Science Board, Office of the Undersecretary of Defense (DSB/OUSD(A)), September 1987, 84 pages, http://www.dtic.mil/cgi-

bin/GetTRDoc?AD=ADA188561&Location=U2&doc=GetTRDoc.pdf, Accessed on January 4, 2009

Deng D., Purvis M. and Purvis M., Software Metric Estimation: An Empirical Study Using An Integrated Data Analysis Approach, International Conference on Service Systems and Service Management, June 9-11, 2007, pp. 1-6

DoD – Deparment of Defense, Risk Management Guide for DoD Acquisition (Sixth Edition, Version 1.0), August 2006, www.dau.mil/pubs/gdbks/risk_management.asp, Accessed on 12/20/2008

DoDD 8000.01, Department of Defense Directive 8000.01, "Management of DoD Information Resources and Information Technology", February 27, 2002

DoDI 5000.02, Department of Defense Instruction 5000.02, "Operation of the Defense Acquisition System", December 8, 2008, www.dtic.mil/whs/directives/corres/pdf/500002p.pdf, Accessed on December 20, 2008

DoDI 5000.04-M-1, Department of Defense Instruction 5000.04-M-1, "Cost and Software Data Reporting (CSDR) Manual", April 18, 2007, www.dtic.mil/whs/directives/corres/pdf/500004m1p.pdf, Accessed on December 20, 2008

Draper N. and Smith H., Applied Regression Analysis, 3rd edition, John Wiley & Sons, New York, 1998, 706 pages

Fairley R. and Willshire M., Iterative Rework: The Good, the Bad, and the Ugly, Computer, Volume 38, Number 9, Los Alamitos, California, September 2005, pp. 34-41, http://doi.ieeecomputersociety.org/10.1109/MC.2005.303, Accessed on September 23, 2009

Ferens D. and Christensen D., Does Calibration Improve Predictive Accuracy?, Software Technology Support Center (STSC) Cross Talk, The Journal of Defense Software Engineering, April 2000, http://www.stsc.hill.af.mil/crosstalk/2000/04/ferens.html, Accessed on December 15, 2008

Ferens D., Tech Views, DoD Software Tech News - New Directions in Software Estimation, The Data and Analysis Center for Software (DACS), Volume 11, Issue 3, Rome, New York, October 2008, http://www.softwaretechnews.com/stn_view.php?stn_id=47&article_id=113, Accessed on September 23, 2009

Fischman L., McRitchie K., Galorath D. D., Inside SEER-SEM, Software Technology Support Center (STSC) Cross Talk, The Journal of Defense Software Engineering, April

2005, http://www.stsc.hill.af.mil/crosstalk/2005/04/0504Fischman.html, Accessed on December 22, 2009

Fisher G., Cost Considerations in System Analysis, RAND Report, December 1970, 348 pages

Foss T., Stensrud E., Kitchenham B. and Myrtveit I., A Simulation Study of the Model Evaluation Criterion MMRE, IEEE Transactions on Software Engineering, Volume 29, Number 11, November 2003, pp. 985-995

Gallo M., Koza E., Holzman M. and Hardin P., An Approach for Building a Normalized Software Database using SRDRs, 41st Annual DoD Cost Analysis Symposium, Williamsburg VA, 20-22 February 2008, http://www.technomics.net/pdf/SRDR%20Software%20Database.pdf, Accessed on April 25, 2010

GAO - United States Government Accountability Office, Defense Acquisitions Assessments of Selected Weapon Programs, March 2008, http://www.gao.gov/new.items/d08467sp.pdf, Accessed on January 4, 2009

Gao X. and Lo B., An integrated software cost model based on COCOMO and function point approaches, Proceedings of the Software Education Conference, Dunedin, New Zealand, November 22-25, 1994, pp. 86-93

Gaylek J., Long L., Bell K., Hsu R. and Larson R., Software Cost and Productivity Model, Aerospace Report Number ATR-2004(8311)-1, February 20 2004

Gencel C. and Demirors O., Functional Size Measurement Revisited, ACM Transactions on Software Engineering and Methodology, Volume 17, Number 3, Article 15, June 2008, pp. 15:1-36

Gilb T., Estimating software attributes: some unconventional points of view, ACM SIGSOFT Software Engineering Notes, Volume 11, Number 1, New York, January 1986, pp. 49-59, http://doi.acm.org/10.1145/382300.382312, Accessed on September 23, 2009

Gruschke T., Empirical Studies of Software Cost Estimation: Training of Effort Estimation Uncertainty Assessment Skills, Proceedings of the 11th IEEE International Software Metrics Symposium (METRICS 2005), September 19-22 2005, pp. 3-5

Gupta D., Kaushal S. and Sadiq M., Software estimation tool based on three-layer model for software engineering metrics, Proceedings of the 4th IEEE International Conference on Management of Innovation and Technology (ICMIT 2008), September 21-24 2008, pp. 623-628

Heemstra F. J. and Kusters R. J., Function Point Analysis: Evaluation of a Software Cost Estimation Model, European Journal of Information Systems, Volume 1, Issue 4, 1991, pp. 229-237

Hihn J. and Lum K., Improving Software Size Estimates by Using Probabilistic Pairwise Comparison Matrices, Proceedings of the 10th International Symposium on Software Metrics (METRICS 2004), September 11-17 2004, pp. 140-150

Hillier F. and Lieberman G., Introduction to Operations Research, Eighth Edition, McGraw-Hill, New York, 2005, 1061 pages

Huang S., Chiu N. and Liu Y., A comparative evaluation on the accuracies of software effort estimates from clustered data, Information and Software Technology, Volume 50, Elsevier, 2008, pp. 879-888

Huang X., Capretz L., Ren J. and Ho D., A Neuro - Fuzzy Model for Software Cost Estimation, Proceedings of the 3rd International Conference on Quality Software (QSIC 2003), November 6-7 2003, pp. 126-133

Humphrey W., The Personal Software Process (PSP), Carnegie Mellon University Software Engineering Institute (CMU/SEI-2000-TR-022), Pittsburg, Pennsylvania, November 2000, 54 pages, http://www.sei.cmu.edu/reports/00tr022.pdf, Accessed on October 4, 2009

Huo M., Verner J., Zhu L. and Ali Babar M., Software Quality and Agile Methods, Proceedings of the 28th Annual International Computer Software and Applications Conference: Design and Assessment of Trustworthy Software-Based Systems (COMPSAC 2004), Hong Kong, China, September 27-30 2004, pp. 520-525

Idri A., Khoshgoftaar T. and Abran A., Can Neural Networks be easily Interpreted in Software Cost Estimation?, Proceedings of the 2002 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2002), May 12-17, 2002, pp. 1162-1167

Ikoma M., Ooshima M., Tanida T., Oba M. and Sakai S., Using a Validated Model to Measure the Agility of Software Development in a Large Software Development Organization, Proceedings of the 31st International Conference on Software Engineering (ISCE 2009), Vancouver, Canada, May 16-24 2009, pp. 91-100

Jalali O., Evaluation Bias in Effort Estimation, In Partial Fulfillment of the Requirements for the Degree of Master in Science in Computer Science, Lane Department of Computer Science and Electrical Engineering, West Virginia University (WVU), 2008, 142 pages

Jayaraman A. and Llopart M., Characteristics that Make One Estimation Technique Better than Others, Carnegie Mellon University School of Computer Science Institute for Software Research (CMU-ISR), Pittsburg, Pennsylvania, May 2007, 10 pages

Jeffery R., Ruhe M. and Wieczorek I., Using Public Domain Metrics to Estimate Software Development Effort, Proceedings of the 7th International Symposium on Software Metrics (METRICS 2001), April 4-6 2001, pp. 16-27

Jeffery D., Time-Sensitive Cost Models in the Commercial MIS Environment, IEEE Transactions on Software Engineering, Volume 13, Issue 7, Piscataway, New Jersey, July 1987, pp. 852-859

Jensen R. and Dupaix L., Normalizing Defense Cost and Resource Center Data: A Feasibility Study, Software Technology Support Center (STSC), 2008

Jones C., Estimating Software Costs: Bringing Realism to Estimating, McGraw-Hill Publishing, New York, 2007, 644 pages

Jones C., Software Cost Estimating Methods for Large Projects, Software Technology Support Center (STSC) Cross Talk, The Journal of Defense Software Engineering, April 2005, http://www.stsc.hill.af.mil/crosstalk/2005/04/0504Jones.html, Accessed on September 29, 2006

Jones C. J., 42nd Department of Defense Cost Analysis Symposium (DoDCAS) presentation, "MAIS vs. MDAP - Understanding the Difference", 2009, 12 slides, http://209.48.244.135/DODCAS%20Archives/42nd%20DODCAS%20(2009)/Software%20and%20Data/5a_Jones_Presentation.pdf, Accessed on October 29, 2009

Jorgensen M., Practical Guidelines for Expert-Judgment-Based Software Effort Estimation, IEEE Software, Volume 22, Number 3, May-June 2005, pp. 57-63

Kalb, G.E., The pursuit of accurate source lines of code sizing, Proceedings of the IEEE National Aerospace and Electronics Conference (NAECON), Volume 2, 1988, pp. 698 - 700

Kemerer C. F., An Empirical Validation of Software Cost Estimation Models, Communications of the ACM, Volume 30, Issue 5, May 1987, pp. 416-429

Keung J. and Kitchenham B., Experiments with Analogy-X for Software Cost Estimation, Proceedings of the 19th Australian Conference on Software Engineering (ASWEC 2008), March 26-28 2008, pp. 229-238

Khalifa M. and Verner J., Drivers for Software Development Usage, IEEE Transactions on Engineering Management, Volume 47, Issue 3, August 2000, pp. 360-369

Kitchenham B., Mendes E. and Travassos G., Cross versus Within-Company Cost Estimation Studies: A Systematic Review, IEEE Transactions on Software Engineering, Volume 33, Number 5, May 2007, pp. 316-329

Kitchenham B., Pickard L., MacDonell S., and Shepperd M., "What accuracy statistics really measure", Software - Proceedings of the IEE, Volume 148, Issue 3, June 2001

Lan C., Tseng C. and Lai K., Developing a Negotiation-based Intelligent Tutoring System to Support Problem Solving: A Case Study in Role-play Learning, Proceedings of the 8th IEEE international Conference on Advanced Learning Technologies (ICALT 2008), Volume 00, Number 1, Section 5, 2008, pp. 356-360

Lane J. and Boehm B., Modern Tools to Support DoD Software Intensive Systems of Systems Cost Estimation, The Data and Analysis Center for Software (DACS Report), August 2007, https://www.thedacs.com/techs/abstracts/abstract.php?dan=347336, Accessed on October 4, 2009

Lewis J.P., Large Limits to Software Estimation, ACM Software Engineering Notes, Volume 26, Number 4, July 2001, pp. 54-59

Li J. and Ruhe G., Decision Support Analysis for Software Effort Estimation by Analogy, Proceedings of the 3rd International Workshop on Predictor Models in Software Engineering (PROMISE 2007), Minneapolis, Minnesota, May 20, 2007, pp. 6-15

Lipke W., EVM and Software Project Management - Our Story, Software Technology Support Center (STSC) Cross Talk, The Journal of Defense Software Engineering, Hill Air Force Base, Utah, November 2002, http://www.stsc.hill.af.mil/crosstalk/2002/11/lipke.html, Accessed on May 11, 2009

Liu Q. and Mintram R., Preliminary Data Analysis Methods in Software Estimation, Software Quality Journal, Volume 13, Number 1, March 2005, pp. 91-115

Liu Q. and Mintram R., Using Industry Based Data Sets in Software Engineering Research, Proceedings of the 3rd International Workshop on Summit on Software Engineering Education (SSEE 2006), Shanghai, China, May 20, 2006, pp. 33-36

Loerch A., Learning Curves, Encyclopedia of Operations Research and Management Science, Second Edition, Editors: S. I. Gass and C. M. Harris, Kluwer Academic Publishing, Boston, Massachusetts, 2001, pp. 445-448

Long L. G. and Lucas R. H., Cost Estimating Relationships (CERs) For Software Development, Air Force Materiel Command, Space and Missile Systems Center

(AFMC/SMC), Aerospace Report Number: TOR-96(8504-01)-3, Los Angeles California, May 20, 1996, 32 pages

Lopez-Martin C., Yanez-Marquez C. and Gutierrez-Tornes A., Fuzzy Logic Systems for Software Development Effort Estimation Based Upon Clustering of Programs Segmented by Personal Practices, Proceedings of the Electronics, Robotics and Automotive Mechanics Conference (CERMA 2006), September 2006, pp. 367-372

Lott C., Breathing New Life into the Waterfall Model, IEEE Software, Volume 14, Issue 5, September-October 1997, pp. 103-105

Lum K. T., Baker D. R. and Hihn J. M., The Effects of Data Mining Techniques on Software Cost Estimation, Proceedings of the IEEE International Engineering Management Conference (IEMC - Europe 2008), Estoril, Portugal, June 28-30, 2008, pp. 1-5

Lurie P., Estimating Relationships II: Complex Models Brief, George Mason University Course: Operations Research: Cost Analysis (OR-651), Lecture, Alexandria, Virginia, March 2006

Madachy R., System Dynamics Modeling of an Inspection-Based Process, Proceedings of the 18th International Conference on Software Engineering (ICSE 1996), Berlin, Germany, March 25-29, 1996, pp. 376-386

Mair C. and Shepperd M., Making Software Cost Data Available for Meta-Analysis, Proceedings of 8th International Conference on Empirical Assessment in Software Engineering (EASE 2004), May 2004, 9 pages, http://dec.bournemouth.ac.uk/ESERG/Technical_Reports/TR04-02/TR04-02.pdf, Accessed on December 22, 2008

Marciniak J. and Reifer D., Software Acquisition Management: Managing the Acquisition of Custom Software Systems, John Wiley & Sons, Inc., New York, 1990, 290 pages

Martin C. L., Pasquier J. L., Yanez C. M. and Tornes A. G., Software Development Effort Estimation Using Fuzzy Logic: A Case Study, Proceedings of the 6th Mexican International Conference on Computer Science (ENC 2005), September 26-30, 2005, pp. 113-120

McConnell S., Software Estimation: Demystifying the Black Art, Microsoft Press, Redmond, Washington, 2006, 308 pages

McLoone P. and Rohde S., Performance Outcomes of CMMI-Based Process Improvements, United States Department of Defense (DoD), The Data and Analysis Center for Software (DACS), Software Tech News: Performance Outcomes from Process Improvements, Volume 10, Number 1, Rome, New York, March 2007,

https://www.softwaretechnews.com/stn_view.php?stn_id=41&article_id=76, Accessed on April 24, 2009

Mendes E., DiMartino S., Ferrucci F. and Gravino C., Effort Estimation: How Valuable is it for a Web Company to Use a Cross-company Data Set, Compared to Using Its Own Single-company Data Set?, Proceedings of the 16th International World Wide Web Conference (WWW 2007), May 8-12, 2007, pp. 963-972, http://www.cs.auckland.ac.nz/weta/techreports/2005/WETA-TR-01.pdf, Accessed on January 4, 2009

Mertes K., Calibration of the Checkpoint Model to the Space and Missile Systems Center (SMC) Software Database (SWDB), Air Force Institute of Technology (AFIT), September 1996, 121 pages

Meyer M. and Booker J., Eliciting and Analyzing Expert Judgment: A Practical Guide, American Statistical Association - Society for Industrial and Applied Mathematics, Series on Statistics and Applied Probability (ASA-SIAM), Philadelphia, Pennsylvania, 2001

Meyers B. and Oberndorf P., Managing Software Acquisition: Open Systems and COTS Products, Addison-Wesley Professional: informIT, July 2001, 288 pages

Miller F., Paradis R. and Whalen K., Iterative Development Life Cycle (IDLC): A Management Process for Large-Scale Intelligent System Development, Proceedings of the 1991 IEEE International Conference on Tools for AI, San Jose, California, November 1991, pp. 520-521

MIL-HDBK-881A, Department of Defense (Military) Handbook, Work Breakdown Structures for Defense Materiel Items, Office of the Undersecretary of Defense (Acquisition, Technology, and Logistics) (OUSD(AT&L)), July 30, 2005, http://www.acq.osd.mil/pm/currentpolicy/wbs/MIL_HDBK-881A/MILHDBK881A/WebHelp3/MIL-HDBK-881A%20FOR%20PUBLICATION%20FINAL%2009AUG05.pdf, Accessed on January 9, 2009

Minkiewicz A., The Evolution of Software Size: A Search for Value, Software Tech News – New Directions in Software Estimation, Volume 11, Number 3, October 2008, pp. 18-22, http://www.softwaretechnews.com/stn_view.php?stn_id=47&article_id=116, Accessed on January 4, 2009

Minkiewicz A., Tackling the Cost Challenges of System of Systems, Software Technology Support Center (STSC) Cross Talk, The Journal of Defense Software Engineering, May 2006, http://www.stsc.hill.af.mil/crosstalk/2006/05/0605Minkiewicz.html, Accessed on January 4, 2009

Misra S., Modeling Design / Coding Factors That Drive Maintainability of Software, Software Quality Journal, Volume 13, Springer Science and Business Media, Inc., Netherlands, 2005, pp. 297-320

Miyazaki Y., Terakado M. and Ozaki K., Robust Regression for Developing Software Estimation Models, Journal of Systems and Software, Volume 27, Number 1, Elsevier Science Inc., New York, October 1994, pp. 3-16

Mohagheghi P., Anda B. and Conradi R., Effort estimation of use cases for incremental large-scale software development, Proceedings of the 27th International Conference on Software Engineering (ICSE 2005), Edinburgh, Scotland, United Kingdom, May 15-21, 2005, pp. 303-311

Molokken K. and Jorgensen M., A Review of Surveys on Software Effort Estimation, Proceedings of the 2nd International Symposium on Empirical Software Engineering (ISESE 2003), Rome, Italy, September 30-October 1, 2003, pp. 223- 230

Morgan J., A Cost Estimation Model for the Fourth Generation Language (4GL) Software Development Environments, PhD Dissertation, George Mason University, 1997, 457 pages

Moser S. and Nierstrasz O., The Effect of Object Oriented Frameworks on Developer Productivity, Computer, Volume 29, Number 9, September 1996, pp. 45-51
Motsko M., Oberndorf P., Pario E. and Smith J., Rules of Thumb for the Use of COTS Products, Carnegie Mellon University Software Engineering Institute (CMU SEI), December 2002, 37 pages, http://www.sei.cmu.edu/publications/documents/02.reports/02tr032.html, Accessed on January 5, 2009

Moul J., Presentation, Through Life Support and Costing 2009 Defense Conference sponsored by Defense IQ – a Division of IQPC (International Quality & Productivity Center), London, United Kingdom, June 23-25, 2009

Musilek P., Pedrycz W., Sun N. and Succi G., On the Sensitivity of COCOMO II Software Cost Estimation Model, Proceedings of the 8th International Symposium on Software Metrics (METRICS 2002), Ottawa, Canada, June 4-7, 2002, pp. 13-20

Myrtveit I., Stensrud E. and Shepperd M., Reliability and Validity in Comparative Studies of Software Prediction Models, IEEE Transactions on Software Engineering, Volume 31, Number 5, May 2005, pp. 380-391

Najberg A., The TASC Software Cost and Requirements Estimator (TASCOR) A Top-Down, Structured Approach to the Software Cost Estimation Process, Proceedings of the

1988 National Aerospace and Electronics Conference (NAECON 1998), Volume 2, Dayton, Ohio, May 23-27, 1988, pp. 686-691

Naseem A., Want to Make Your System Highly Available? Add COTS to the Middle (ware), COTS Journal Online, May 2004, http://www.cotsjournalonline.com/articles/view/100114, Accessed on September 24, 2009

Neale J. and Liebert R., Science and Behavior: An Introduction to Methods of Research, 3rd Edition, Prentice-Hall Series in Social Learning Theory, Englewood Cliffs, New Jersey, 1986, 324 pages

Oh S., Pedrycz W. and Park B., Self-Organizing Neuro-Fuzzy Networks Based on Evolutionary Fuzzy Granulation, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, Volume 33, Issue 2, March 2003, pp. 271-277

Ormon S., Development of a Hierarchical Model-Based Decision-Support Tool for Assessing Uncertainty of Cost Estimates, Masters of Science in Industrial Engineering Thesis, Mississippi State University, May 2002, 84 pages

Pagliano G. and O'Rourke R., Evolutionary Acquisition and Spiral Development in DoD Programs: Policy Issues for Congress, Congressional Research Service, David D. Acker Library and Knowledge Repository, Defense Acquisition University (DAU), Fort Belvoir, Virginia, April 8 2004, http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA435457&Location=U2&doc=GetTRDoc.pdf, Accessed on September 24, 2009

Park R., Software Size Measurement: A Framework for Counting Source Statements, Carnegie Mellon University Software Engineering Institute (CMU SEI), Pittsburgh, Pennsylvania, September 1992, 242 pages, http://www.sei.cmu.edu/pub/documents/92.reports/pdf/tr20.92.pdf, Accessed on January 5, 2009

Pendharkar P. C., Subramanian G. H. and Rodger J. A., A Probabilistic Model for Predicting Software Development Effort, IEEE Transactions on Software Engineering, Volume 31, Number 7, July 2005, pp. 615-624

Peters L., O'Connor E., Pooyan A. and Quick J., The relationship between time pressure and performance: A field test of Parkingson's Law, Journal of Occupational Behavior, John Wiley & Sons, Ltd. (pre-1986), Volume 5, Issue 4, ABI/INFORM Global, October 1984, pp. 293-299

Pfleeger S. L., Wu F. and Lewis R., Software Cost Estimation and Sizing Methods, RAND Corporation, 2005, 127 pages

Pillai K. and Nair V., A Model for Software Development Effort and Cost Estimation, IEEE Transactions on Software Engineering, Volume 23, Number 8, August 1997, pp. 485-497

Popp M., Software Resource Data Report (SRDR) - how it is changing NAVAIR software estimating, Department of Defense Cost Analysis Symposium (DoDCAS), February 2008, http://209.48.244.135/DODCAS%20Archives/41st%20DODCAS%20(2008)/Track%204%20-%20Data%20and%20Analytic%20Tools/T4S5a_Popp.pdf, Accessed on January 5, 2009

Putnam L, Tutorial Software Cost Estimating and Life-Cycle Control: Getting the Software Numbers, Institute of Electrical and Electronics Engineers, Inc. (IEEE), New York, 1980, 349 pages

Putnam L. and Myers W., Measures for Excellence: Reliable Software on Time within Budget, Yourdon Press, Prentice Hall PTR, Upper Saddle River, New Jersey, 1992, 378 pages

Putnam L., Jr., 2007 DoDCAS Data Trends in Software Development, Department of Defense Cost Analysis Symposium (DoDCAS), February 2007, http://209.48.244.135/DODCAS%20Archives/40th%20DODCAS%20(2007)/Track%203/Session1.1Putnam.pdf, Accessed on January 5, 2009

Reifer D., Informal discussion at the 13th Annual Practical Software and Systems Measurement (PSM) Users' Group Conference: Measurement in a Dynamic Business and Government Environment, June 26, 2009

Reifer D. (Editor), Software Management, 6th Edition, IEEE Computer Society Press, 2002, 592 pages

Rico D., Using Cost Benefit Analyses to Develop Software Process Improvement (SPI) Strategies, The Data and Analysis Center for Software (DACS), September 18 2000, https://www.thedacs.com/techs/abstracts/abstract.php?dan=347008, Accessed on October 4, 2008

Rose L., COTS Integration Questions and Checklist, Software Productivity Consortium, May 2000, https://acc.dau.mil/CommunityBrowser.aspx?id=146865, Accessed on December 18, 2008

Royce W., Successful Software Management Style: Steering and Balance, IEEE Software, Volume 22, Number 5, September-October 2005, pp. 40-47

Ruhe M., Jeffery R. and Wieczorek I., Cost Estimation for Web Applications, Proceedings of the 25th International Conference on Software Engineering (ICSE 2003), May 3-10 2003, pp. 285-294

Sackman H., Erikson W. J. and Grant E. E., Exploratory experimental studies comparing online and offline programming performance, Communications of the ACM, Volume 11, Number 1, 1968, pp. 3-11

SAF or Deputy Assistant Secretary of the Air Force (Cost and Economics), Air Force Cost Risk and Uncertainty Analysis Handbook, April 2007

Sage A., Lecture 7: COTS and Cost Estimation in Systems Integration and Architecting, George Mason University Architecture-Based Systems Engineering (IT-850), October 2005

Salter C., A Decision Framework for the Selection of Commercial-Off-the-Shelf Technology for Organizational Processes, PhD Dissertation, George Mason University, May 2001

Schaeffer M., Osiecki L., Richter K. and Baldwin K., Improving the Integrity of the CMMI Product Suite, Defense AT&L Magazine, July-August 2007, pp. 14-16, http://www.dau.mil/pubs/dam/2007_07_08/scha_ja07.pdf, Accessed on October 4, 2009

Schoedel R., PROxy Based Estimation (PROBE) for Structured Query Language (SQL), Carnegie Mellon University/Software Engineering Institute Technical Note, CMU/SEI-2006-TN-017, May 2006, http://www.sei.cmu.edu/reports/06tn017.pdf, Accessed on December 22, 2009

Shepperd M., Software project economics: a roadmap, Future of Software Engineering (FOSE 2007), IEEE Computer Society International Conference on Software Engineering, 2007, pp. 304-315

Silva A. and Stam A., Nonparametric Two-Group Classification: Concepts and a SAS-Based Software Package, The American Statistician, Volume 52, Issue 2, May 1998, pp. 185-197, http://www.jstor.org/stable/2685479, Accessed on February 11, 2009

Selby R., Hafen G., Mink A., Nicol M., Flowe R., Lile R. and R. Gold, "Software Risk and Estimation Workshop Outbrief," Software in Acquisition Workshop, October 16-17, 2007, http://www.acq.osd.mil/sse/ssa/docs/SoftwareRiskandEstimationOutbrief20071018.pdf, Accessed on December 10, 2008

Spurlock M. A., Introduction to Software Cost Estimation: How long will it take and how much will it cost?, May 14 2003, http://www.dau.mil/conferences/presentations/2003/T7-SoftwareCostEstimating-MarthaSpurlock.pdf, Accessed on December 1, 2008

Strike K. T., El Emam K. and Madhavji N., Software Cost Estimation with Incomplete Data, IEEE Transactions on Software Engineering, Volume 27, Number 10, October 2001, pp. 890-908

S3DB - Space Systems Software Database Data Collection Instruction Manual, Software Technology Support Center (STSC) product developed for Air Force Cost Analysis Agency (AFCAA), January 1, 2005

Stutzke R., Software Estimating Technology: A Survey, Software Technology Support Center (STSC) Crosstalk, The Journal of Defense Software Engineering, Hill Air Force Base, Utah, May 1996, http://www.stsc.hill.af.mil/crosstalk/1996/05/estimati.asp, Accessed on August 31, 2009

Subramanian G., A Methodology to Attain Site Specificity and Model Simplicity in Software Development Effort Estimation, A Dissertation submitted to the Temple University Graduate Board in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy, Philadelphia, Pennsylvania, January 1991

Sutcliffe A., The Domain Theory: Patterns for Knowledge and Software Reuse, Lawrence Erlbaum Associates, Inc., Mahwah, New Jersey, 2002, 398 pages

Tadayon N., Neural Network Approach for Software Cost Estimation, International Conference on Information Technology Coding and Compression (ITCC 2005), April 4-6 2005, pp. 815-818

Tan Y. and Mookerjee V., Comparing Uniform and Flexible Policies for Software Maintenance and Replacement, IEEE Transactions on Software Engineering, Volume 31, Number 3, March 2005, pp. 238-255

Tomer A., Goldin L., Kuflik T., Kimchi E. and Schach S., Evaluating Software Reuse Alternatives: A Model and Its Application to an Industrial Case Study, IEEE Transactions on Software Engineering, Volume 30, Number 9, September 2004, pp. 601-612

Tucker J. (Director), Panel on Statistical Methods in Software Engineering, Committee on Applied and Theoretical Statistics Board on Mathematical Sciences, National Research Council, Statistical Software Engineering, National Academy Press, Washington D. C., 1996

Twala B., Cartwright M. and Shepperd M., Comparison of various methods for handling incomplete data in software engineering databases, Proceedings of the 2005 International

Symposium on Empirical Software Engineering (ISESE 2005), Noosa Heads, Australia, November 17-18, 2005, pp. 105-114

Valerdi R., The Constructive Systems Engineering Cost Model (COSYSMO), Dissertation in Industrial and Systems Engineering, University of Southern California, August 2005, 137 pages

Venkatachalam A. R., Software Cost Estimation using Artificial Neural Networks, Proceedings of the 1993 International Joint Conference on Neural Networks (IJCNN 1993), Volume 1, October 25-29 1993, pp. 987-990

Vick S., Degrees of Belief: subjective probability and engineering judgment, American Society of Civil Engineers (ASCE) Press, 2002, 455 pages

Vijayakumar S., Use of historical data in software cost estimation, Computing & Control Engineering Journal, Volume 8, Number 3, June 1997, pp. 113-119

Vosburgh J., Curtis B., Wolverton R., Albert B., Malec H., Hoben S. and Liu Y., Productivity Factors and Programming Environments, Proceedings of the 7th International Conference on Software Engineering, IEEE Computer Society, Orlando, Florida, March 26-29 1984, pp. 143-152

Vouk M., On the Cost of Mixed Language Programming, ACM SIGPLAN Notices, Volume 19, Number 12, December 1984, pp. 54-60

Wang Y., Song Q. and Shen J., Grey Learning Based Software Stage-Effort Estimation, IEEE International Conference on Machine Learning and Cybernetics, Volume 3, Hong Kong, China, August 19-22, 2007, pp. 1470-1475

Wieczorek I. and Ruhe M., How Valuable is Company-Specific Data Compared to Multi-Company Data for Software Cost Estimation, Proceedings of the 8th IEEE Symposium on Software Metrics (METRICS 2002), June 4-7, 2002, pp. 237-246

Yahya M. A., Ahmad R. and Lee S. P., Effects of Software Process Maturity on COCOMO II's Effort Estimation from CMMI Perspective, IEEE International Conference on Research, Innovation and Vision for the Future (RIVF), July 13-17, 2008, pp. 255-262

Yakimovich D., Bieman J. M. and Basili V. R., Software architecture classification for estimating the cost of COTS integration, Proceedings of the 21st International Conference on Software Engineering (ICSE 1999), Los Angeles, California, 1999, http://www.cs.umd.edu/~basili/publications/proceedings/P83.pdf, Accessed on November 28, 2008

Yourdon E., Managing High-Intensity Internet Projects, Prentice Hall, Upper Saddle River, New Jersey, 2002, 226 pages

Yu W.D, Smith D. and Huang S., Software Productivity Measurements, Proceedings of the 15th Annual International Computer Software and Applications Conference (COMPSAC 1991), Toyoko, Japan, September 11-13 1991, pp. 558-564

Zhao Y. F., Tan H. and Zhang W., Software cost estimation through conceptual requirement, Proceedings of the Third International Conference on Quality Software, November 6-7, 2003, pp. 141-144

Zubeck J., Enhanced Unified Modeling Language Model-Checking for Business Software Applications, PhD Dissertation, George Mason University, 2006, 245 pages

Appendix A: Cost Estimating Methods

Algorithmic

Literary references to algorithmic experiments abound. As these methods are data-driven, standard data sets are available; some have a nominal fee to use. The PRedictOr Models In Software Engineering (PROMISE) repository provides historical data sets for researchers and practitioners to use in their software cost estimation experiments. [Boetticher and Lokhandwala 2007] There were eighty-nine total data sets on July 26, 2009, up from eighty-six on April 18. [http://promisedata.org/?cat=11 2009] The International Standards Benchmarking Steering Group (ISBSG) maintains a database of commercial and government software projects and distributes data sets for a nominal fee. [http://www.isbsg.org/products 2009] Researchers and practitioners, with any data set, may examine any parameter(s) as independent or dependent variable(s). The literature has cases where the data is not available publically. [Chui and Huang 2007; Jones 2005; Mukhopadhyad and Kekre 1992; Tomer, Goldin, Kuflik, Kimchi and Schach 2004; Vijayakumar 1997] There is also literature exploring software cost estimation results from a single institution versus multiple institutions based on hypothesized differences. [Briand, El Emam, Surmann, Wieczorek and Maxwell 1999; Kitchenham, Mendes and Travassos 2007; Mendes, DiMartino, Ferrucci and Gravino 2007; Wieczorek and Ruhe 2002]

Much of the literature on algorithmic methods compares parametric cost estimating relationships using ordinary least squares regression in unit space or log space with other algorithmic methods. Other algorithmic methods, found in my literature search, are:

(a) Learning algorithms such as neural networks and genetic algorithms; methodologies include fuzzy logic, back-propagation, radial basis function, support vector regression, and bagging predictors [Baskeles, Turnham and Bener 2007; Berlin, Raz, Glezer and Zviran 2009; Braga, Oliveira and Meira 2007; Deng, Purvis and Purvis 2007; Huang, Capretz, Ren and Ho 2003; Idri, Khoshgoflaar and Abran 2002; Oh, Pedrycz and Park 2003; Pendharkar, Subramanian and Rodger 2005; Tadayon 2005; de Barcelos Tronto, da Silva and Sant'Anna 2007; Venkatachalam 1993; Wang, Song and Shen 2007]

(b) Statistical techniques such as robust regression, Kolmogorov-Smirnov test, classification and regression trees (CART), factor analysis including principal component analysis (PCA), evolving self-organizing map (ESOM), cluster analysis with k-means, stepwise analysis of variance (ANOVA), and correlation analysis, parametric and non-parametric [Briand, Langley and Wieczorek 2000; Briand, El Emam, Surmann, Wieczorek and Maxwell 1999; Deng, Purvis and Purvis 2007; Huang, Chiu and Liu 2008; Jeffery, Ruhe and Wieczorek 2001; Liu and Mintram 2006; Lopez-Martin, Yanez-Marquez and Gutierrez-Tornes 2006; Ruhe, Jeffery and Wieczorek 2003; Sackman, Erikson and Grant 1968; Wieczorek and Ruhe 2002]

(c) Simulation techniques such as Monte Carlo modeling and system dynamics modeling [Briand, El Emam and Bomarius 1998; Hihn and Lum 2004; Musilek, Pedrycz, Sun and Succi 2002; Silva and Stam 1998; Abdel-Hamid and Madnick 1989; Madachy 1996]

(d) Mathematical formulas: the Cobb-Douglas equation and software cost estimation equations such as Boydston, Walston-Felix, Bailey-Basili, Doty, Albrecht-Gaffney, Kemerer, Matson-Barret-Melichamp, Basic COCOMO, Intermediate COCOMO, COCOMO 2.0 models, Putnam's SLIM, Norden-Rayleigh curve fitting, and Halstead's complexity. [Nidiffer 2006]

Algorithmic "methods are divided into functions and arbitrary function approximators (AFA). According to Myrtveit et al., 'arbitrary function approximators do not make any assumptions regarding the relationship between the predictor and response variables' while functions assume otherwise." [Jalali 2008]

System dynamics uses simulation to shed light on complex, interrelated activities and events. One simulation tool, Timed Colored Petri Nets, models temporal system behaviors. "Models are represented as networks modified with positive and negative feedback loops. Elements within the models are expressed as dynamically changing levels or accumulations (the nodes), rates or flows between the levels (the lines connecting the nodes), and information relative to the system that changes over time and dynamically affects the flow rates between the levels (the feedback loops)." [Boehm, Abts, Brown,

Chulani, Clark, Horowitz, Madachy, Reifer and Steece 2000] Systems dynamics modeling has been applied to the software environment by Ray Madachy's simulation of Brooks' law [declaring 'Adding more people late in the project makes it later'] in 1999. Tarek Abdel-Hamid paired with Stuart Madnick to model dynamic software project management in 1991.

Learning-oriented techniques are algorithmic: techniques such as neural networks and machine learning. Made popular by the idea of neurons in the human body, neural networks use training sets, which are input data subsets, to determine algorithmic parameter values to minimize the differences between predicted and actual values. "Extremely large data sets are needed to accurately train neural networks with intermediate structures of any complexity. Also, for negotiation and sensitivity analysis, the neural networks provide little intuitive support for understanding the sensitivity relationships between cost driver parameters and model results." [Boehm, Abts, Brown, Chulani, Clark Horowitz, Madachy, Reifer and Steece 2000] To explore machine-learning algorithms, researchers set aside a data portion to use as a test. In one case, the researchers experimentally set aside 10 records out of 60 and then performed a second, comparative, experiment with 20 records set aside. The observed phenomenon was error increases as training set size decreases. [Baskeles, Turnhan and Bener 2007] Machine learning has also been criticized for using an "unclear and closed structure of the computation process." [Berlin, Raz, Glezer and Zviran 2009]

Checklists are one of the basic, common, algorithmic methods, though the steps may vary. Seven steps for software cost estimating documented by Barry Boehm (to use several independent techniques, compare them, iterate, and follow-up) expanded in the time since he proposed them. [Boehm 1981] Recently, McConnell published a checklist with twelve steps and Jones, ten steps. The checklists recommend the analyst perform and iterate estimation activities in each creator's order. The actual order differs, but the recommended estimation activities are extremely similar. While Jones recommends using a size prediction to estimate software development workload and explicitly characterizing worker's assignment scope, McConnell recommends estimating ranges using worst case as a lower bound, best case as a higher bound, and most likely case as a central tendency measurement.

Table A-1: Comparison of Estimation Checklists [McConnell 2006; Jones 2007

McConnell 2006 **Checklist for Individual Estimates**	**Jones 2007** **Standard Sequence for Software Cost Estimates**
1. Is what's being estimated clearly defined?	Step 0: Analyze the requirements
2. Does the estimate include all the kinds of work needed to complete the task?	Step 1: Start with…size prediction using an estimating tool…[or] by extrapolation from function point totals…[or] by analogy with similar projects…[or] using…intuition…[or] statistical methods or Monte Carlo simulation
3. Does the estimate include all the functionality areas needed to complete the task?	Step 2: Identify the activities to be included…work that will be performed
4. Is the estimate broken down into enough detail to expose hidden work?	Step 3: Estimate software defect potentials and removal methods
5. Did you look at documented facts (written notes) from past work rather than estimating purely from memory?	Step 4: Estimate staffing requirements…a characteristic assignment scope, or amount of work that can be done by a single employee

McConnell 2006 **Checklist for Individual Estimates**	**Jones 2007** **Standard Sequence for Software Cost Estimates**
6. Is the estimate broken down into enough detail to expose hidden work?	Step 5: Adjust assumptions based on capabilities and experience…experts will have larger assignment scopes and higher production rates
7. Is the productivity assumed in the estimate similar to what has been achieved on similar assignments?	Step 6: Estimate effort and schedules
8. Does the estimate include a Best Case, Worst Case, and Most Likely Case?	Step 7: Estimate development costs
9. Is the Worst Case really the worst case? Does it need to be made even worse?	Step 8: Estimate maintenance and enhancement costs
10. Is the Expected Case computed appropriately from the other cases?	Step 9: Present your estimate to the client and defend it against rejection
11. Have the assumptions in the estimate been documented?	
12. Has the situation changed since the estimate was prepared?	

Analogy

Expert-based techniques provide a range of estimating options: "surveys have been consistent in reporting expert judgment as the most common prediction technique in the literature". [Shepperd 2007] Reasoning by analogy is a form of inductive reasoning: "an analogy is a statement of a logical relationship between two similar things that are compared with each other." [http://www.samford.edu/schools/netlaw/dh2/logic/logic1.html 2009] Issues relating to choice of analogies are amounts of missing data, data quality, data homogeneity, and relevance of data selection framework. The following table provides alternatives to consider as these issues arise.

Table A-2: Estimation by Analogy Alternatives [Li and Ruhe, *PROMISE '07*]

Issue to decide	**Alternatives to consider**
Impact analysis of missing values	Preliminary knowledge
Dealing with missing values	Deletion and imputation techniques: NULL value

Issue to decide	Alternatives to consider
Object selection	Hill climbing, simulated annealing, forward and backward sequential selection algorithms
Converting continuous attributes to discrete attributes	Rough Set Analysis (RSA)-based attribute weighting; based on interval, frequency, or both; other machine learning techniques
Attribute weighting and selection	RSA-based, Wrapper, hill-climbing, genetic algorithm
Determining similarity measures	Distance-based, local-global similarity principle
Retrieving analogs	Using similarity measures or rule-based heuristics
Determining closest analogs	Fixed number of analogs; learning process; data availability
Analogy adaptation strategy	Mean, weighted mean, median, linear extrapolation
Choosing [accuracy] evaluation criteria	Some conventional criteria: e.g., MMRE, PRED
Comparison methods in general	Accuracy-based methods

The use of heuristics is common in making analogies. Heuristics are rules-of-thumb; heuristic methods designed "to fit a specific problem type rather than a variety of applications" are "likely to discover a very good feasible solution, but not necessarily an optimal solution". [Hillier and Lieberman 2005] Meta-heuristic methods provide structure and guidance to tackle specific problems. Heuristic and meta-heuristic methods guide analogy choices, using means such as hill climbing, simulated annealing, and genetic algorithms. Rough set analysis explores data relationships as discernable, complementary, similar, or negatively similar using forward inclusion and backward inclusion. [Orlowska 1998]

Expert-based

When data is sparse or untrustworthy, experts, whether alone or in combination with other experts or other methods, can generate or verify software cost, effort, and schedule estimates. To obtain expert opinions, questioning is required. Often careful phrasing of the question can evoke pertinent responses. "Simple process changes such as reframing

questions can lead to more realistic estimates". [Jorgensen 2005] Querying experts and understanding their responses "…can be difficult to do and subject to numerous biases". [SAF 2007] The following table categorizes these biases as motivational and cognitive.

Table A-3: Motivational and Cognitive Bias [SAF 2007]

Motivational Bias	Cognitive Bias
Social pressure (face to face)	Inconsistency (opinion changes over time)
Impressions (not face to face)	Anchoring
Wishful thinking	Relating to irrelevant analogies
Misunderstandings	Underestimation
Career goals	
Project advocacy	
Competitive pressures	

On the one hand, motivational bias, based on people's need for social acceptance, can occur along organizational or personal preference lines. [Meyer and Booker 2001; Vick 2002] Social pressures, job pressures, and competitive pressures lead to groupthink and group-behaviors. An individuals' preferred methods of thinking and operating influences his or her interpretation (or misinterpretation) of details and situations. On the other hand, cognitive bias, based on psychological studies of human information processing constraints, can lead to inconsistent or inappropriate judgments if an expert relies on too few analogies or on irrelevant ones. [Meyer and Booker 2001]

For software cost estimation, experts are gathered and queried in a structured process called the Delphi Method. To arrive at a group consensus on the estimate, the facilitator

elicits individual responses to standardized queries, squashes extraneous discussions, and provides relevant, often anonymous, feedback to the participants. [http://www.iit.edu/~it/delphi.html 2007] With the Delphi Method, as with all expert-based methods, the validity of the estimate depends on the knowledge, wisdom, experience, and judgment of experts – and on the future unfolding as envisioned, often by extrapolating from the past. This remains the Achilles Heel of expert-based methods. "One main disadvantage of this estimation technique is that it cannot be used in projects involving a new domain where there is no existing expert knowledge." [Jayaraman and Liopart 2007]

Engineering-level

The engineering level of estimation involves disaggregation of software activities from the top down or the bottom up. WBS levels range from a low of one to well over ten. Practically speaking, three to six levels are sufficient to be mutually independent and collectively exhaustive. 'Web-Help' on MIL-STD-881A, The Department of Defense Handbook, Work Breakdown Structures for Defense Materiel Items, states "just as the system is defined and developed throughout its life cycle, so is the work breakdown structure. The WBS will be developed and maintained based on the systems engineering efforts throughout the system's life cycle." [http://www.acq.osd.mil/pm 2009] Engineering-level estimates can rely on algorithmic, analogous, expert-based, and other methods,

depending on the lowest level of the WBS or the discrete WBS element. For commercial items, appropriate vendors provide quotes upon request. Given the range of the quotes, different techniques may be used: an average, a weighted average, a maximum quote, or, if there are enough quotes, a probability distribution of values can be generated and a value or a range can be randomly chosen from the distribution or chosen based on pre-determined criteria. [Bryant 2008] Engineering level estimates ensure coverage of explicit WBS levels.

Others

Other techniques, such as Price to Win and Parkinson's Law, influence managerial choices and decisions in large, complex, software-intensive DoD programs; however, they are not usual tools for cost analysts. Price to Win, based on the bidder's assessment of what the customer is willing to spend, may not satisfy either the bidder or the customer. [Marciniak and Reifer 1990] If the system decision criterion is to go with the cheapest bid, Price to Win is a strategy contracts use to win an award. Parkinson's Law marries effort with corresponding, available task times. Referred to as a "goal-setting phenomenon", it is not a traditional cost-estimating method. [Bryan and Locke 1967] If schedules lengthen, Parkinson's Law increases costs: people on the payroll continue to get paychecks.

Along with PSP and PROBE, Bayesian methods are hybrid methods. Hybrid methods combine expert-based inputs with established mathematic algorithms or models. Bayesian theory underlies the COQUALMO model, in the COCOMO family, for defect introduction. [Chulani, Santhanam, Moore, Leszkowicz and Davidson 2001] PROBE asks engineers to imagine their conceptual design taking shape and size using proxies, categorized by software operation and programming language. Software operation categories are computations, data handling, and program input-output processing. Using PSP's five size ranges, from very small to very large, subject matter experts (i.e., the engineers) take each proxy's size estimates, combine them, and use statistical linear regression to arrive at a total number of LOC in the project. [Humphrey 2000]

Appendix B: Explanation of Software Cost Models

The COCOMO model is public with documentation online. The other models are proprietary with limited documentation. Input variables for software cost models, chiefly COCOMO input variables, are the most studied independent variables in the literature. Many software estimation models have been developed and used over time. The following models are of interest due to their applicability to my research.

Table B-1: Software Cost Models: Source and Site

Software Cost Model	Source	Web Site
COCOMO II	USC/CSSE	http://csse.usc.edu/tools/COCOMOII.php
COCOTS	USC/CSSE	http://csse.usc.edu/csse/research/COCOTS/index.html
SLIM	QSM Incorporated	http://www.qsm.com/
SEER-SEM	Galorath Incorporated	http://www.galorath.com/
TruePlanning	PRICE® Systems	http://www.pricesystems.com/
KnowledgePLAN®	SPR Incorporated	http://www.spr.com/project-estimation.html
Sage	Software Engineering Incorporated	http://seisage.net/sage.htm

COCOMO II

Dr. Barry Boehm had a hand in developing the Doty Model and then COCOMO I, now called COCOMO 81. [Spurlock 2003] "The fundamental production function in software engineering is the function relating delivered source instructions as outputs to development man-months as inputs." [Boehm 1981] In 1995, Dr. Boehm at the University of Southern California's Center for Systems and Software Engineering launched the COCOMO II model suite. Inputs include a SLOC estimate, scale factors, product attributes, platform attributes, personnel attributes, and project attributes. The scale factors and

attributes are nominal, ranging from either low or very low to high, very high, extremely high, or extra-high. [http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html 2008]

Scale factors cover five areas: uniqueness of the effort, flexibility of the development, risk mitigations regarding architecture, cohesion of the team, and process maturity. If the effort is an upgrade, the prior version(s) may act as a precedent or if the effort mimics another effort, the prior effort is a precedent. More precedents lower the scale factor. Process maturity factors align inversely with CMMI ratings. [http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html 2008]

Cost drivers span requirements and constraints. Requirements include development schedule, software reliability, reusability, and computer turnaround time. Constraints include lifecycle needs of the documentation, experience levels and continuity of personnel, computer main storage available, and computer execution times. There are other cost drivers: how complex the effort is, how volatile the platform is, and whether the development occurs at one site or at multiple sites. [http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html 2008]

COCOTS

COCOTS, a member of the COCOMO II model suite, stands for Constructive COTS Model. Commercial-off-the-shelf (COTS) software can encompass a range of functions for software developers and maintainers from automated tool support for routine or specialized administrative tasks to a called-on component, vital to the performance of the software. Due to COCOMO II model cost drivers for 'use of software tools' and 'language and toolset experience', the functionality of automated tool support and the skills of the operators is embedded in the COCOMO II model and excluded from the COCOTS model. The modification of a COTS product pushes it into a professed reusable component, handled in the COCOMO II model as a "developed for reusability" cost driver and excluded from the COCOTS model. The COCOTS model applies to the called-on component(s) of the software application, with the premise of "the following COTS phenomena…

- You have no control over a COTS product's functionality or performance
- Most COTS products are not designed to interoperate with each other
- You have no control over a COTS product's evolution
- COTS Vendor behavior varies widely" [http://csse.usc.edu/csse/research/COCOTS/modeldesc.html 2009]

These phenomena apply to all COTS, regardless of their role in the software development.

The COTS products may or may not fulfill their intended role as-is. When the users have a need to adjust the COTS product to their environment or their intended use, "COTS

product tailoring and tuning" or "Glue Code development" can result along with the necessary testing, verification, and validation of the COTS adjustments as they occur within the development process. [http://csse.usc.edu/csse/research/COCOTS/modeldesc.html 2009] The COCOTS model has fourteen input parameters to estimate effort:

1. An exponential scale factor, called Application Architectural Engineering (AAREN), for the "percentage of module interfaces specified in the architecture, subjectively averaged with the percentage of known risks mitigated through the system architecting process"
2. Thirteen effort multipliers break out into:
    a. Four personnel drivers such as COTS integrator personnel capability (ACIPC), integrator personnel continuity (APCON), COTS integrator experience with the product (ACIEP), and integrator experience with COTS integration processes (AXCIP);
    b. Five COTS components drivers: COTS product maturity (ACPMT), COTS supplier product extension willingness (ACSEW), COTS product interface complexity (APCPX), COTS supplier product support (ACPPS), COTS supplier provided training and documentation (ACPTD); and
    c. Four application/system drivers: constraints on application system or subsystem reliability (ACREL), application interface complexity (AACPX), constraints on COTS technical performance (ACPER), and portability across application systems (ASPRT). [Abts 2004]

For glue code or interface code between two programs or modules, Chris Abts proposed specific values from very low to very high for each of the thirteen effort multipliers. The COCOTS Data Collection Survey and User's Manual for USC COCOTS.2002.1 spreadsheet tool calibrated parameters like COCOMO II.2000.

Table B-2: COCOTS Glue Code Model Parameters [Abts 2002]

Glue Code Parameters					
Nonlinear Scale Factor					
	Very Low (VL)	*Low (L)*	*Nominal (N)*	*High (H)*	*Very High (H)*
AAREN	4.00	3.00	2.00	1.00	0.00
Aggregate					
Cost Drivers	*Very Low (VL)*	*Low (L)*	*Nominal (N)*	*High (H)*	*Very High (H)*
	Personnel Drivers				
ACIEP	1.34	1.16	1.00	0.86	0.75
ACIPC	1.60	1.27	1.00	0.79	0.62
AXCIP		1.12	1.00	0.89	0.79
APCON	1.58	1.26	1.00	0.80	0.63
	COTS Component Drivers				
ACPMT	1.45	1.20	1.00	0.83	0.69
ACSEW		1.07	1.00	0.94	0.88
APCPX		0.82	1.00	1.22	1.48
ACPPS		1.14	1.00	0.88	0.77
ACPTD	1.20	1.09	1.00	0.91	0.84
	Application / System Drivers				
ACREL		0.88	1.00	1.14	1.30
AACPX		0.84	1.00	1.19	1.42
ACPER			1.00	1.11	1.22
ASPRT			1.00	1.07	1.14

SLIM

The software equation is: $Effort = [Size * B^{(1/3)}/Productivity\ Parameter]^3 * (1/Time^4)$.

This equation, along with a Manpower Build-up Index (MBI), is the basis for the SLIM-ESTIMATE model offered by Quantitative System Management (QSM). The Productiv-

ity Parameter equals $\frac{E}{B^{2} t_d^{4}}$, where E is effort, B is a skills factor that is also a function of size, and t_d is development time. [Putnam and Myers 1992] "A simple scale of integer values, called the Productivity Index (PI),…behaves exponentially…The development time that falls at the intersection of the MBI and size/PI is the minimum possible time for the particular project." [Putnam and Myers 1992] More development time, when staff and/or budget are constrained or slowly built up, provides a workable solution, so long as the development time is a few months plus the minimum possible time. [Putnam and Myers 1992] Shorter development time results in triage, an effort to deliver software under a compressed schedule. [Yourdon 2002]

SEER-SEM

The SEER-SEM model, marketed by Galorath Incorporated, went through an upgrade in 2008 to version 7.3. [http://www.reuters.com/article/pressRelease/idUS235245+14-Oct-2008+PRN20081014 2008] SEER-SEM stands for the System Evaluation and Estimation of Resources – Software Estimating Model. It translates inputs into effective sizing, effective technology, and staffing complexity factors to use in effort and schedule estimating equations. [http://www.stsc.hill.af.mil/crosstalk/2005/04/0504Fischman.html 2009] "Users follow a Work Breakdown Structure (WBS) describing each CSCI, CSC, and CSU (module or element) to be estimated." [http://cost.jsc.nasa.gov/pcehhtml/pceh225.htm 2009] Outputs include a minimum sche-

dule, suggested staffing categories, risk reports and graphs regarding the estimated effort, costs, and schedules, along with tradeoff calculations, over the total life cycle starting with preliminary design and ending with operations and maintenance of the software. [http://cost.jsc.nasa.gov/pcehhtml/pceh225.htm 2009] Four 'knowledge bases' can be calibrated to user specifications regarding a choice of target operating platform, designated application types, designated development methods, and designated development standards, such as the Software Engineering Institute's Capability Maturity Model Integrated (CMMI) or Internal Standards Organization (ISO)-9000 for quality. [http://cost.jsc.nasa.gov/pcehhtml/pceh225.htm 2009] "SEER-SEM utilizes a unique process that simulates a 10,000 iteration Monte Carlo for risk analysis." [http://cost.jsc.nasa.gov/pcehhtml/pceh225.htm 2009]

TruePlanning

Starting out as the PRICE-S model, TruePlanning is the software variant of a system of models, with the first upgrade to PRICE-S called TRUE-S, introduced in 2003. [http://www.pricesystems.com/news/2003_10_30.asp 2009] Inputs are: SLOC, a choice of seven application categories, a productivity factor, three complexity parameters, the operating environment, hardware utilization or capability, percentage of new code, internal integration effort factor, external integration effort factor, project start and end dates. [http://cost.jsc.nasa.gov/pcehhtml/pceh.htm 2008] There are optional inputs for financial calculations and for risk calculations. The TruePlanning model outputs effort estimates

and software development schedule estimates by life-cycle phase. Software metrics for parametric estimates are:

- Software size
- Effort in labor hours, dollars, and staff size
- Productivity
- Requirements stability
- Schedule
- Environment
- Quality, and
- Cost-Performance Index (CPI) from EVMS.

[http://www.pricesystems.com/white_papers/Implementing%20a%20Parametric%20Estimating%20System%20for%20Development_PRICE%20Format.pdf 2009]

KnowledgePLAN

Capers Jones founded Software Productivity Research (SPR) in 1984. The company announced on December 9, 1996] "the introduction of SPR KnowledgePLAN, a fundamentally new tool for software estimation" replacing the CHECKPOINT model. [http://findarticles.com/p/articles/mi_m0EIN/is_/ai_18919783# 2008] Based on SPR consultant "experience from over 6,700 software projects, KnowledgePLAN leads software managers through an intuitive planning environment for managing small or large projects" and can integrate SPR KnowledgePLAN with Microsoft Project.

[http://findarticles.com/p/articles/mi_m0EIN/is_/ai_18919783# 2008] A Project Wizard can help set-up an initial estimate, create a 'base' estimate at user-chosen levels of refinement for either effort or schedule, or create a detailed project schedule. [http://www.spr.com 2008] "KnowledgePLAN is different than most models in that it works primary in sizing by analogy or with function points instead of SLOC. The model will accept SLOC, but converts SLOC to function points using conversion factors in the model." [http://fast.faa.gov/pricing/c1919-19D.htm 2009] Other features available for analysts are the following: Integration with Crystal Reports and MS ACCESS (COTS products); Support of Open Database Connectivity (ODBC) to connect to enterprise data; Flexibility with entering or changing WBS; Domain categorization choices to customize environment and product preferences or inputs. [http://www.spr.com 2008] Outputs include an estimate of schedule, staffing, and effort in dollars or time, with optional risk analysis for selected inputs such as software size, defects, reliability, maintenance, and productivity. [http://fast.faa.gov/pricing/c1919-19D.htm 2009]

SAGE

Sage, for software schedule and cost estimation, needs four general inputs: size, management template, product template, and project constraints with over thirty parameters to portray the development environment and project. It outputs predictions of estimated and worst case outcomes, and will optionally estimate source code growth to depict cost and schedule risks. Staffing profiles can be output for development and maintenance us-

ing IEEE/EIA 12207, Systems and software engineering – Software life cycle processes. [http://seisage.net/sage.htm 2009]

Appendix C: Detail on Development Methods and Life Cycle Phases

One way of looking at development methods is Barry Boehm's view of 20th and 21st century software engineering where he extends a 'Hegelian' hypothesis from 1950 to today. [Boehm 2006] Waterfall methods sprang from software development efforts in the 1950s, reflecting a deliberate, sequential approach. Code-and-fix development methods arose in the 1960s, resembling a patchwork quilt approach to building software. A two-pronged solution included structured programming methods, involving top-down sequential approaches, and formal proof or 'programming calculus' methods to gain domain understanding. [Boehm 2006] Structured programming methods gave rise to object-oriented methods, standards, maturity models, and the notion of software factories; whereas formal proof methods gave rise to business fourth-generation software languages, Computer-Assisted-Design/Computer-Assisted-Manufacturing (CAD/CAM), and users as programmers. [Boehm 2006] Finally, concurrent processes, domain-specific software architectures, agile methods, and product-in-line reuse led to integrated systems and software engineering, hybrid agile plan-driven methods for rapid evolution environment, supported by service-driven architectures and model-driven development. [Boehm 2006]

Use of software development methods is determined by facilitating conditions in the organization. [Khalifa and Verner 2000] In DoD, the software development for large acquisition programs uses contractors, where they determine an appropriate software development method, facilitated by the program office. In staged contracts, notably used for

RAD, the "old waterfall model has accumulated a series of incremental improvements" so the customer and contractor can determine at specified intervals whether to continue the development effort. [Lott 1997] The waterfall method is not the only development method able to handle partitioned development efforts. Iterative development can adapt to the goals of the software program and take many differently named but similar forms. The accompanying methods facilitate the following software products:

(1) Prototype development methods for user interfaces;

(2) Agile development methods for daily builds;

(3) Incremental development methods for weekly builds;

(4) Spiral development methods for evolving products [Fairley and Willshire 2009]

There are different ways to categorize the life cycle phases from the beginning of an idea for a program to its retirement. The waterfall life cycle stages relate to the Agile methods cycle steps, although implementation and test phases in waterfall are mirrored by iterative small releases in Agile. [Huo, Verner, Zhu and Barbar 2004] Different software methodologies relate to different life cycle phases as knowledge increases with program maturity. [http://www.stsc.hill.af.mil/resources/tech_docs/gsam4/chap2.pdf 2003]

The DoD acquisition life cycle phases, codified in the Department of Defense Instruction 5000.02 dated December 8, 2008 recognize the applicability of evolutionary development methodologies for software and hardware. Despite this recognition, prior DoD development activities were labeled 'heavyweight' development, mainly following the waterfall

process. Earlier standards, such as the expired DoD Standard 2167A, recommended the waterfall development model for requirements generation. [Defense Science Board 1987] Pre-systems acquisition activities generally move along the following lines: (1) a material solution analysis, with rudimentary requirements to generate and explore alternatives using approximate cost-benefit analyses; (2) a meeting called the Milestone 'A' review to decide whether to proceed; and (3) an initial technology development, usually with modeling, to aim for desired characteristics. As technology development ends, another meeting, named Milestone 'B' review, determines whether to initiate the program. If the program is initiated, typical system acquisition activities are: (1) engineering and manufacturing development, usually focused on matured requirements; (2) a meeting called the Milestone 'C' review to decide whether to proceed; and (3) a production and deployment phase, usually accompanied with Low Rate Initial Production (LRIP) and Initial Operational Test and Evaluation (IOT&E) activities. After LRIP, IOT&E, or other tangible signs of success, a meeting, designated as "Go" in the figure, ascertains whether and how to field the program. Once program fielding begins, the program has an Initial Operational Capability (IOC). After the "Go" meeting, the production and deployment phase ends and an operations and support phase begins. After operations are in place, the program has Full Operational Capability (FOC). Operations and support (O&S) activities are sustainment activities.

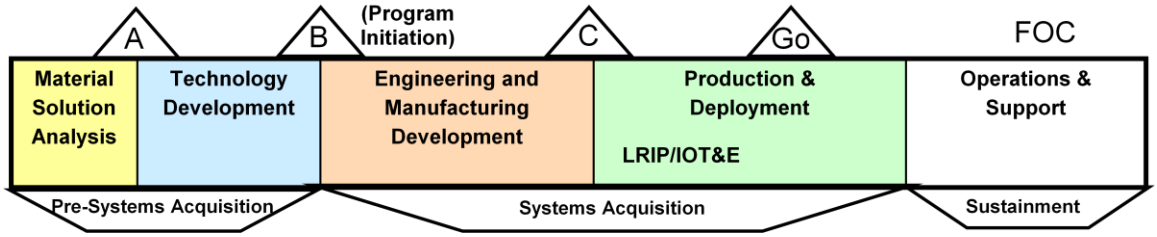

Figure 13: Standard DoD Weapon System Life Cycle [Moul 2009]

The costs of the retirement of a program or weapon system are included in life cycle costs; however, the figure above does not extend to system retirement and subsequent disposal or transition.

Steve McConnell categorized and rated different life cycle models using a scale from poor to excellent in 1996. [Rico 2000] The commonly regarded software development life cycles at that time were Waterfall (pure and modified), Code-and-Fix, Evolutionary (Prototyping and Delivery), Design (Design-to-Schedule, Design-to-Tools), Spiral, Staged Delivery, and Commercial-Off-The-Shelf (COTS). In the following table, the descriptions are terse. To explain further, ambiguity applies to requirements understanding. Innovation applies to a new resulting system. High reliability, managed growth, and managed risks apply to resulting system. Schedule applies to the development method's success in meeting a set, constrained program schedule. Low overhead and adapting to changes applies to the overall development. Process and progress visibility applies to the development effort for the stakeholders. Easy to manage applies to the overall development meaning that neither the developer nor the manager requires superhuman skills.

Table C-1: Comparison of Software Development Life Cycles [Rico 2000]

Description	Typical DOD major software-intensive program	Pure Waterfall	Code and Fix	Spiral Excellent	Modified Waterfall	Evolutionary Prototyping	Staged Delivery	Evolutionary Delivery	Design-to-Schedule	Design-to-Tools	Commercial-Off-The-Shelf
Ambiguity	Yes	Poor	Poor	Excellent	Fair-Excellent	Excellent	Poor	Fair-Excellent	Poor-Fair	Fair	Excellent
Innovation	Sometimes	Poor	Poor	Excellent	Fair-Excellent	Poor-Fair	Poor	Poor	Poor	Poor-Excellent	Poor-Excellent
High Reliability	Yes	Excellent	Poor	Excellent	Excellent	Fair	Excellent	Fair-Excellent	Fair	Poor-Excellent	Poor-Excellent
Managed Growth	Yes	Excellent	Poor-Fair	Excellent	Excellent	Excellent	Excellent	Excellent	Fair-Excellent	Poor-Excellent	N/A
Managed Risks	Yes	Poor	Poor	Fair	Fair	Fair	Fair	Fair	Fair-Excellent	Poor-Fair	N/A
Schedule	Yes	Fair	Poor	Fair	Fair	Poor	Fair	Fair	Excellent	Excellent	Excellent
Low Overhead	Yes	Poor	Excellent	Fair	Excellent	Fair	Fair	Fair	Fair	Fair-Excellent	Excellent
Adapting to changes	Yes	Poor	Poor-Excellent	Excellent	Fair	Excellent	Poor	Fair-Excellent	Poor-Fair	Excellent	Poor
Process Visibility	Yes	Poor	Fair	Excellent	Fair	Excellent	Fair	Excellent	Fair	Excellent	N/A
Progress Visibility	Yes	Fair	Poor	Poor	Fair-Excellent	Fair	Excellent	Excellent	Excellent	Excellent	N/A
Easy to Manage	Yes	Fair	Excellent		Poor-Fair	Poor	Fair	Fair	Poor	N/A	Fair

Contrast the 1996 software development life cycle descriptions with the following 2003 descriptions: waterfall (pure and modified) with just waterfall; staged delivery and Design-to-Schedule with incremental; evolutionary prototyping and evolutionary delivery with just evolutionary; and spiral excellent with just spiral. Code-and-fix, Design-to-Tools, and COTS as a software methodology left the software development life cycle lexicon. [http://www.stsc.hill.af.mil/resources/tech_docs/gsam4/chap2.pdf 2003]

One view of the software life cycle shows conception springing from maintenance. Considerable iteration, not shown, must exist between and among the circles. The view of the software development life cycle must satisfy the stakeholder's program perspective.

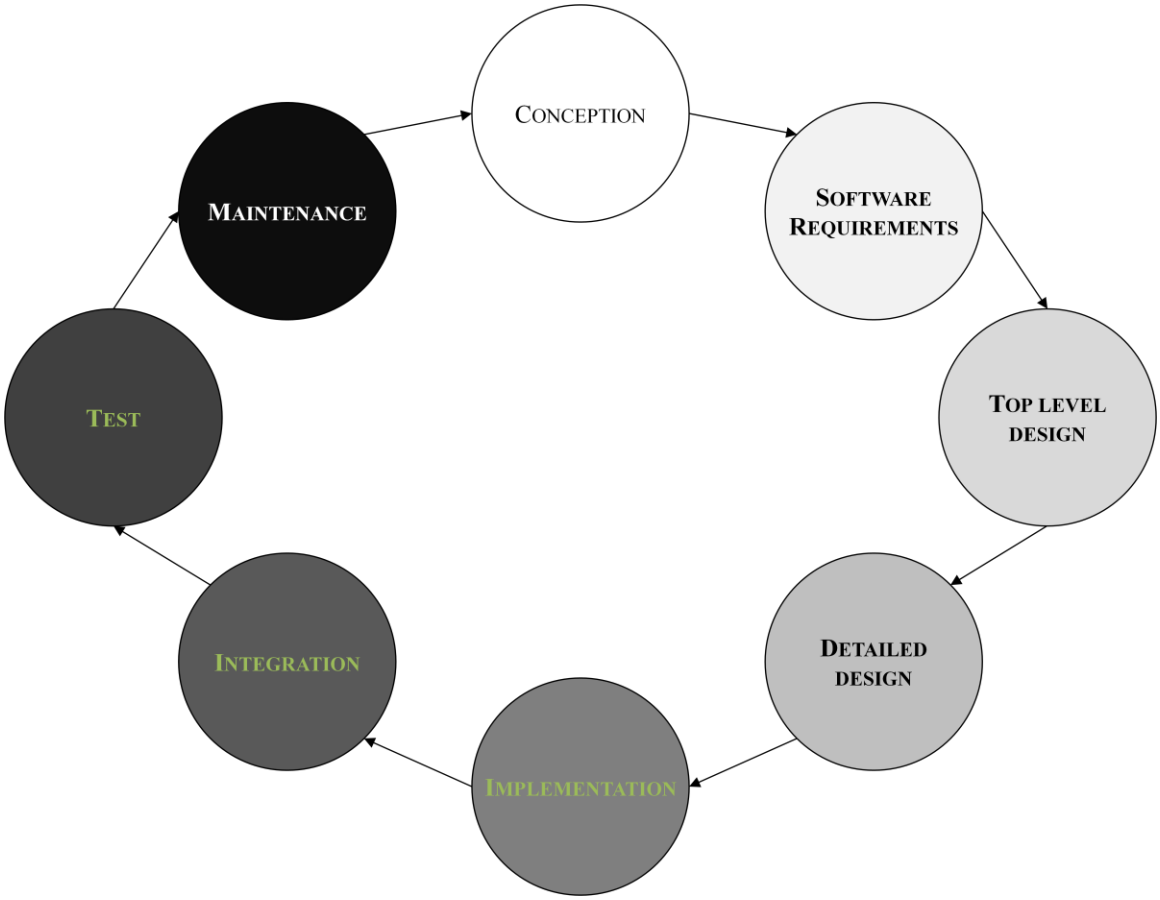

Figure 14: Life Cycle Process Circle [Bennatan 2000]

Appendix D: Cost Estimating Uncertainty and Risk

Risk explicitly considers how much the plans or the estimates can differ from reality or ground truth. [DoD 2006] "Cost risk analysis is the process of quantifying and displaying the uncertainty associated with point estimates of cost." [Lurie 2006] Less risky cost relationship equations have a well fitting model, data points inside the range of the equation's inputs, certain or reasonable parameters, narrow error variance, and independent cost drivers. [Lurie 2006] GAO recommends DoD cost estimates at major decisions or milestones present a range instead of a point estimate of costs and highlight the associated treatments of cost risks and uncertainties. [GAO 2008]

In an article more than a decade ago in the Software Technology Support Center's Crosstalk Magazine, the cone of uncertainty was born, though unnamed. Based on Barry Boehm's 1981 seminal text titled Software Engineering Economics and on Richard Stutzke's familiarity with DoD contracts, the author noted, "In general, since more information becomes available, e.g., product structure and size and team productivity, the accuracy of the estimates increases as a project proceeds." [Stutzke 1996] From program conception to the point of software acceptance, the final costs acknowledged on DoD contracts ranged from at least 0.25 times more to as much as 4 times more than original early estimates. The point of software acceptance is not the end of the program, just the end of the first iteration of the program's development. According to Steve McConnell's blog, "The Cone is a hope, but not a promise." [http://forums.construx.com/blogs/stevemcc/archive/2007/05/23/update-on-the-cone-of-uncertainty.aspx 2009]

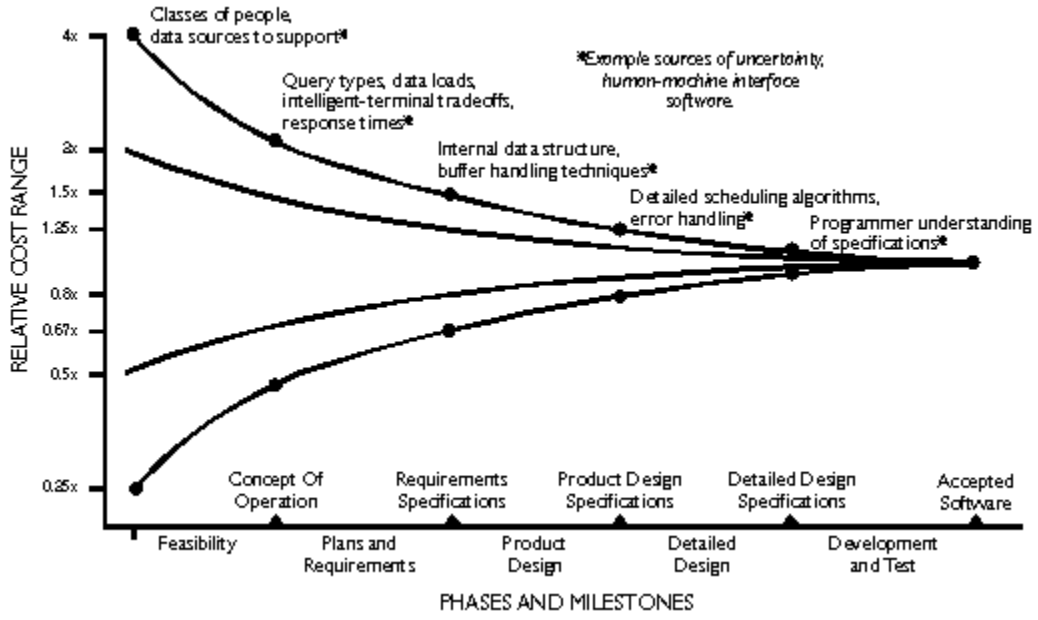

Figure 13: Cone of Uncertainty [Stutzke 1996; McConnell 2006]

Accompanying the cone of uncertainty for software cost estimates are managerial impacts arising from these uncertainties. If the estimates are relatively accurate, the project development cycle can proceed on a controlled, efficient, and credible path. However, inaccurate estimates may hamper the project or program's path by limiting management's ability to establish control, be efficient, and be credible to the stakeholders with the resources provided.

The major types of uncertainty are statistical and situational. Statistical uncertainties relate to real world elements subject to fluctuation; situational uncertainties relate to unknowable, but analyzable, future "states of the world" subject to technology developments, force structure developments, strategic developments, and other developments.

[Fisher 1970] Risks with their potential issues for software, impact on costs, and mitigation strategies are in the table below.

Table D-1: Risk Types and Mitigation Strategies

[https://learn.dau.mil/CourseWare/66_9/18_riskmgmt/18_t2crisks/crisks0051.html 2006]

Risk Type	Potential Software Risk Issue	Cost Impact / Mitigation Strategy
Program-Level	-Excessive, immature, unrealistic, or unstable requirements -Lack of involvement or understanding -Underestimation of project complexity or dynamic issues	If the requirements vary, become subject to multiple interpretations, or any aspect of project complexity is under-estimated, the life cycle lengthens and costs increase. Setting standards for documentation and stakeholder involvement may lessen misunderstandings and establish reasonable, shared expectations
Program Attributes	-Performance Shortfalls -Unrealistic cost or schedule estimates -Unrealistic cost or schedule allotments	Prior estimates do not apply. If unmet requirements are unique or compelling, a new program may arise to cover the shortfalls. Cost and schedule estimates need to be based on historical cost and schedule performance of similar programs.
Management	-Ineffective project management	Negative effects on life cycle costs Effective managers need to be brought in and allowed to take over
Engineering	-Ineffective integration, assembly and test: quality control; specialty engineering	Negative effects on life cycle costs Establish and enforce controls Measure and publish test plans and results
Work Environment	-Immature or untried design, processes or technologies -Inadequate work plans / configuration control -Inappropriate methods or tool selection or inaccurate metrics	The users, testers, developers, and maintainers should participate in decision-making processes on designs, plans, tools, use of metrics, configuration control, and technologies; once agreed to, these processes need to be implemented and followed
Other	-Poor planning -Too few or too many reviews -Too little or too much documentation -Legal or contractual issues -Obsolescence (including excessive schedules) -Unanticipated maintenance or support costs	To avoid these traps, the program manager should build rapid problem solving into his processes by using ‘what-if’ brainstorming sessions with relevant stakeholders to identify and share potential solutions to hypothesized, future issues

Appendix E: Software and COTS Definitions

DoD defines software as "*Computer programs, procedures, and possibly associated documentation and data, pertaining to the operation of a computer system*." [Ferens 2008] This definition is similar to the IEEE definition where "*software: 1. A set of computer programs, procedures, and associated documentation concerned with the operation of a data processing system; e.g., compilers, library routines, manuals, and circuit diagrams. [JP1] 2. Information (generally copyrightable) that may provide instructions for computers; data for documentation; and voice, video, and music for entertainment or education*." [Medina 2008] Another definition, based on the International Standards Organization (ISO) 9000:2005, 'Fundamentals and Vocabulary,' and ISO 19011:2002, 'Guidelines for Quality and/or Environmental Systems Auditing,' expands the meaning of software to an "*intellectual product consisting of information on a support medium*." [http://www.whittingtonassociates.com/v2/glossary.shtml 2009] Functionality via software has been steadily replacing functionality via hardware in DoD weapon system programs. In the early 2000s, at least eighty percent of the functionality of the Air Force's new fighter airplane, the F-22, was due to software; this 'soft' functionality was previously dependent on hardware. [Spurlock 2003] In the future, decision logic may be required to determine which functions of a physical entity rely only on software since software could be indistinguishable from hardware. [Bollinger 2004]

COTS is "*a [software] product that is: (1) sold, leased, or licensed to the general public; (2) offered by a vendor trying to profit from it; (3) supported and evolved by a vendor, who retains the intellectual property rights; (4) available in multiple, identical copies; and (5) used without source code modification by a consumer.*" [Meyers and Obendorf 2002] The Open Source Initiative (OSI), at http://opensource.org, describes "what it means to distribute software that is open source, namely:

- Free distribution (i.e., license cannot restrict selling or giving away)
- Source code (included)
- Derived works (i.e., software can be modified and distributed by others)
- Integrity of the author's source code (i.e., know who gets credit for source code)
- Distribution of license (i.e., forbidding addition of further restrictive licensing) …
- License must not contaminate other software (e.g., both licensed software and OSS can coexist in the same distribution)" [Ayala 2008]

For adopting and integrating COTS components, understanding of the software architecture and the COTS product begins the process of integrating them, along with simultaneous organizational activities such as training, implementing appropriate interfaces, performing Configuration Management, and maintaining mutual nondisclosure agreements between users and COTS vendors. [Rose 2000] To ease the intellectual and administrative burden inherent in understanding the architecture and the COTS products, the emergence and enforcement of interfacing standards, particularly for standards-based architectures, provides developers with the leeway to build interoperable COTS products for new applications. [Motsko, Oberndorf, Pario and Smith 2002; Naseem 2004]

One consequence of using COTS is so-called 'Vendor Lock' throughout the system's life as the customer relies upon the vendor to perform maintenance and upgrade activities in the specific application system. [Sage 2005] Compounding the COTS integration issue is "multiple COTS products are usually integrated to provide COTS solutions. Changes in a single component can significantly affect the performance of other components". [Salter 2001] Reliance on a vendor results in a watchful program managerial role and reliance on multiple vendors requires the program to adjust to evolutionary changes in any of the COTS throughout the program's life cycle. COTS components may be interdependent. [Albert and Brownsword 2002] Interface changes will be at least as important as internal COTS processing changes. There are scores of interfaces: logical, physical, functional, dynamic, internal, external, and environmental. Logical interfaces put a resource demand on the system, even when the connection is through other components, as this is a "deduced" interface. Physical interfaces relate to hardware components driven by software components and they describe a physical connection between hardware devices or between hardware and the physical operating environment, including human interactions. Functional interfaces "translate control, information, or energy" across components. [Sage 2005] Dynamic interfaces change over time and depend on the system's state or the time. Adjusting to changes in interfaces is part of the system maturation process where knowing what they used to do and what they currently do provides a foundation for software maintenance and stakeholder communications.

Effort relating to integrating COTS has four-prongs:

1. COTS market analysis assessing pre- and post-commitment what the COTS can and will do as a component of the program;
2. COTS adaptation for the program, including
    a. Interface identification and development
    b. Architectural identification and resolution to join COTS with the overall program using components, connectors, global structure, and construction within local structures
    c. Tailoring the COTS to the program's environment with new or modified code, affected or unmodified/unaffected code
    d. Tuning the COTS to perform effectively and well in the program's environment by changing the parameters or configurations to achieve performance objectives, perform automatic back-up processes, or configure for data recovery after planned or un-planned processing interrupts
    e. Developing glue code (or new software code) to coordinate concurrent processing, to bridge and mediate processing between components, to handle errors, and to control data and process flows
3. COTS assembly and test in the program's software architecture infrastructure and the program's operational working environment, including independent testing, verification, and validation activities with parallel error identification and debugging which may rework the COTS adaptations
4. COTS evolution or keeping-up with the new updates and releases of the software so the developers/maintainers are aware of potential and actual changes as are the users [Beims and Dabney 2000; Vieira and Madeira 2003; Sage 2005; Zubeck 2006; http://fast.faa.gov/pricing/c1919-19E.htm 2009]

Relationships between a controlled (COTS) and controlling system (DoD software-intensive program) contain embedded assumptions requiring accumulated knowledge to merge components into one program. [Sutcliffe 2002; Albert and Brownsword 2002] Typically, a prime contractor is responsible for assembling and integrating the major DoD software-intensive acquisition program, requiring a series of agreements relating to the contracted activities, including COTS products licenses and warranties, used in development, production, and fielding. Documentation on controlled and controlling sys-

tems must be current and shared between stakeholders to increase the amassed knowledge of the program.

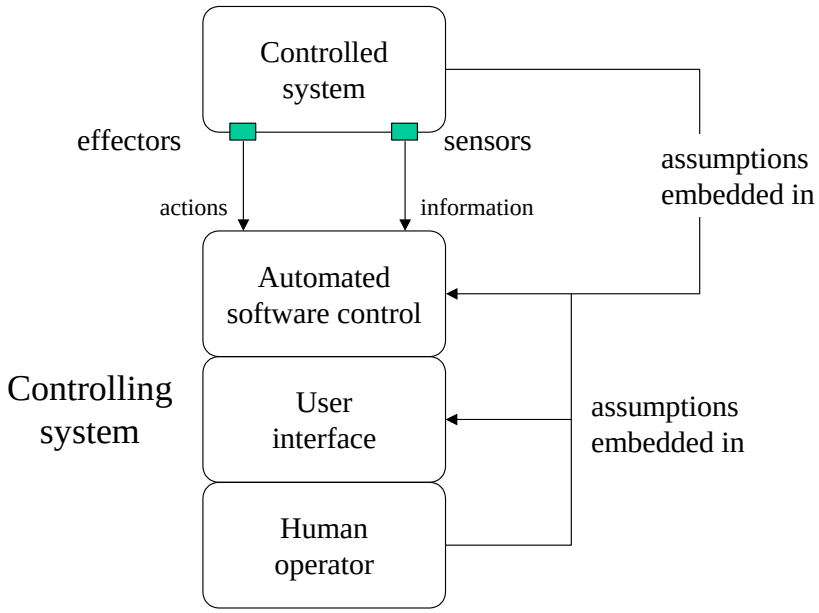

Figure 14: Relationships between controlled and controlling systems [Sutcliffe 2002]

Appendix F: Software Resources Data Report (SRDR) Table

The following table delineates whether the data element is a mandatory item in the companion data dictionary. The clarification or remarks column lists the responsible office or depicts the data type. The classification schema aligns with the SRDR reporting schema. The context section asks for the development organization's details. The product and development section asks for development type, tools and languages used, peak staff numbers, staff experience in the application domain, software size, requirements, explicit WBS mapping, work-in-progress dates, and any subcontractors.

Table F-1: Software Resources Data Reports: Data Element Descriptions

Data Element to Report (as of DID-MGMT-81739, April 20, 2007)	Reported	In Data Dictionary	Clarification / Remarks
Report Context and Development Organization			
Security classification	Mandatory	No	Managed by Program Office
System/Element Name	Mandatory	No	Assigned by DCARC
CSRD Plan Number	Mandatory	No	Assigned by DCARC
Report As-of Date	Mandatory	No	Date of data
Authorizing Vehicle	Mandatory	No	Reference or contract number
Reporting Event	Mandatory	No	In CSDR Plan, Block 14
Submission Number	Mandatory	No	Enter "1" for first time, etc.
Development Organization	Mandatory	Yes	To map development organizations, software components, and submissions
Software Process Maturity	Mandatory	As needed	To explain mechanisms and ratings
Precedents	Mandatory	No	List up to five analogous systems
SRDR Data Dictionary Filename	Mandatory	Yes	Date of last update
Comments	Optional	As needed	About report context and development organization
Product and Development Description			

Data Element to Report (as of DID-MGMT-81739, April 20, 2007)	Reported	In Data Dictionary	Clarification / Remarks
Functional Description	Mandatory	No	What is it? What will it do?
Software Development Characterization	Mandatory	No	How will it be done?
Application Type	Mandatory	No	17 categories with 119 total types
Primary and Secondary Programming Languages	Mandatory	As needed	Based on development effort, not functional size
Percentage of Overall Product Size	Mandatory	As needed	Percentage of developed product
Planned Development Process	Mandatory	As needed	If not an industry standard process
Upgrade or New Development	Mandatory	No	New is also complete replacement
Software Development Method(s)	Mandatory	As needed	Specify type and describe non-standard types in Data Dictionary
COTS/GOTS Applications Used	Mandatory	If proprietary	List if these constitute deliverable(s)
Integration Effort	Optional	As needed	May be expressed in staff-hours, new/modified glue code, or qualitative assessment of effort
Peak Staff	Mandatory	As needed	FTEs for direct labor or as explained in data dictionary
Peak Staff Date	Mandatory	No	Date expected to occur
Hours per Staff-Month	Mandatory	Yes	List accounting standard or provide details on how this was computed
Personnel Experience in Domain	Mandatory	Yes	Categorize by High, Nominal, and Entry Level and provide rationale and explanation of domain experience
Comments	Optional	As needed	
Total Number of Software Requirements	Mandatory	As needed	Do not count interfaces; define counts and units in Data Dictionary
New Software Requirements	Mandatory	No	
Total Number of External Interface Requirements	Mandatory	As needed	Details about count methods in Data Dictionary
New External Interface Requirements	Mandatory	No	
Requirements Volatility	Mandatory, was optional	Yes	Use a qualitative scale (very low, low, nominal, high, very high) and describe
Delivered Size	Mandatory	As needed	Count all code once. Delineate whether new, modified, or reused
Carryover Code	Mandatory	As needed	Delineate what and how much code was carried forward from another report
Auto-generated Code	Mandatory	No	
Subcontractor-Developed Code	Mandatory	As needed	Provide explanation if unknown

Data Element to Report (as of DID-MGMT-81739, April 20, 2007)	Reported	In Data Dictionary	Clarification / Remarks
Counting Convention	Mandatory	Yes	Robert Park's definition recommended
Size Reporting by Programming Language	Optional	No	
Comments	Optional	As needed	
Effort	Mandatory	As needed	Optional ISO 12207 activity definitions
WBS Mapping	Mandatory	No	
Subcontractor Development Effort	Mandatory if sub-contractors	No	
Schedule	Mandatory	As needed	Estimate start and end dates; Can use 1 as start date and define in SRDR Initial Developer Report or SRDR Data Dictionary
Comments	Optional	As needed	
Point of Contact Information	Mandatory	No	Full name, Department name, Telephone, E-mail, Fax, Signature, and Date of Signature

Appendix G: Software Accuracy Measures

Table G-1: Industry-Proposed Accuracy Metrics [Jalali 2008]

Measure	Description	Meaning
Absolute Residual (AR) or Magnitude of Error (ME)	The absolute value of the difference between the actual value and the predicted value	This value can only be positive. The lower the value, the better the prediction; the higher, the worse. If AR or ME equals zero, the predicted equals the actual value.
Relative Error (RE)	On the numerator, actual value minus predicted value, all divided by actual value (for each predicted value)	RE can be positive or negative. If RE is negative, it can go to negative infinity. If RE is positive, it cannot exceed a value of one. If RE is zero, the predicted value equals the actual value.
Average Relative Error (ARE)	The sum of all the individual RE values divided by *n*, the number of values calculated	When ARE is negative, there is average overestimation and when ARE is positive, there is an average underestimation. Generally, smaller ARE positive or negative values represent better matches between predicted and actual values. Small ARE values could, however, mask an imbalance with high positive and high negative differences.
Magnitude of the Relative Error (MRE)	The absolute value of RE	MRE can only be positive. The lower the MRE, the better the prediction; higher MRE means a worse prediction. If MRE is zero, the prediction equals the actual value.
Mean Magnitude of the Relative Error (MMRE)	The sum of all the individual MRE values divided by *n*, the number of values calculated	Generally, smaller MMRE positive values represent better overall agreement in predicted and actual values. However, small MMRE values could mask one or more large deltas. An industry standard is MMRE ≤ 0.25 and is "acceptable for effort prediction" [Conte, Dunsmore and Shen 1986]
Prediction at Level *L* or *l* (PRED(*L*) or PRED(*l*))	In a set of *n* projects, a value, *k*, is the number of projects whose MRE $\leq l$, so PRED(*l*) equals *k* divided by *n*	For PRED(*l*) = *k*/*n*, the *k*/*n* ratio of predicted values are within *l* percentage of actual values. The accepted industry standard is PRED(0.25) ≥ 0.75 [Conte, Dunsmore and Shen 1986]

Measure	Description	Meaning
Magnitude of Error Relative to the Estimate (MER)	MER is the absolute value of the difference between the actual value minus the predicted value divided by the predicted value	MER uses the predicted value as a denominator vice the actual value used in RE to measure the error relative to the predicted value. MER was initially called the Estimation MMRE [Kitchenham, Pickard, MacDonell and Shepperd 2001] There is no accepted industry standard for accuracy.
Mean Squared Error (MSE)	The sum of the squares of each individual actual value minus the corresponding predicted value divided by *n*, the number of values calculated.	MSE "is meaningful for regression models only. It represents the mean value of the error minimized by the regression model." [Conte, Dunsmore and Shen 1986] There is no industry standard for accuracy.
Root Mean Square Error (RMS)	The square root of MSE	RMS "is meaningful for regression models only." [Conte, Dunsmore and Shen 1986] There is no reported, accepted industry standard for accuracy.
Relative Root Mean Square Error (RRMS)	RMS divided by the average actual value	RRMS ≤ 0.25 is the accepted industry standard for accuracy of effort prediction regression models [Conte, Dunsmore and Shen 1986]
Mean of the Balanced Relative Error (BRE)	Subtract the actual value from the predicted value and examine the result. If the result is greater than or equal to zero, divide it by the actual value; if not, divide it by the predicted value.	Since negative RE values represent over-estimation, BRE aligns over-estimation with positive values and under-estimation with negative values. [Miyazaki, Terakado, Ozaki and Nozaki 1994]
Inverted Balanced Relative Errors (IBRE)	Subtract the actual value from the predicted value and examine the result. If the result is greater than or equal to zero, divide it by the predicted value; if not, divide it by the actual value.	IBRE was suggested as a companion to BRE. IBRE forces positive one to represent maximum over-estimation and negative one to represent maximum under-estimation. [Miyazaki, Terakado, Ozaki and Nozaki 1994]

CURRICULUM VITAE

Corinne C. Wallshein graduated from the University of Virginia with a Bachelor of Arts degree in Mathematics in 1978. After an intensive training program with the United States Air Force, she became a UNIX system administrator. In 1982, she transitioned to applications programming for future defense budgets and developed multiple user applications and reusable library programs. Working at the Air Force Surgeon General's office, from 1984 to 1990, she maintained the Provider Requirements Integrated Specialty Model based on estimated medical needs.

From 1990 to 2005, she performed multiple studies of strategic and tactical systems to analyze the impact of potential courses of action across various spectrums of warfare. Acting as a technical contract officer's representative, she structured, wrote, and managed a multimillion-dollar studies contract. As a human resources liaison, she administered personnel management processes. She worked on the Revised Code of Best Practice for Command and Control Assessment as the Air Force representative for a North Atlantic Treaty Organization's Studies and Analysis Team.

Recently she accomplished comprehensive resource analyses for infrastructure and acquisition programs. She has concentrated on software cost estimating since 2005. Her resource analyses include software-intensive systems, such as the Global Positioning System, Joint Primary Aircraft Training System, Defense Satellite Communications System, Tactical Data Link Gateway, Evolved Expendable Launch Vehicle, Predator, Mission Planning System, Future Combat System, Expeditionary Combat Support System, and Defense Integrated Military Human Resources System.

Committed to continuous learning and professional development, she completed Air War College in 1994. She earned her Master of Science in Operations Research and Management Science from George Mason University in 1997. Her goal is to apply the knowledge and discipline gained in her dissertation effort to benefit public and private scientific and analytic communities.