Process-Related Effort and Schedule Estimating Relationships for Software Cost Estimating

Corinne Wallshein
IT Estimating Division
Naval Center for Cost Analysis
corinne.wallshein@navy.mil

Nicholas Lanham
Aviation *Estimating* Division
Naval Center for Cost Analysis
nicholas.lanham@navy.mil

Wilson Rosa
IT Estimating Division
Naval Center for Cost Analysis
wilson.rosa@navy.mi

ABSTRACT

This paper will present updated findings of software effort and duration growth factors (using estimated parameters at the beginning of the effort), effort estimating relationships (EERs), and schedule estimating relationships (SERs) with selected subsets of United States Department of Defense (DoD) Computer Software Configuration Item (CSCI) records. Focusing on early and initial estimating parameters such as requirement counts, staff hours, peak staff counts, duration in months, and source lines of code, data subsets are by software processes and tools. These subsets are process maturity as measured by Capability Maturity Model Integrated (CMMI), primary language, and, when applicable, cost-plus contract type. EERs and SERs will be displayed per goodness of fit criteria. Effort and duration distribution benchmarks are provided to guide cost analysts in normalizing and inspecting data for early and initial estimating parameters. These methods are applicable to all industry sectors.

General Terms

Management, Measurement, Software cost estimation

Keywords

Software effort estimation, software cost estimation, source lines of code, process maturity, CMMI, contract type, software language, software size, empirical analysis, regression

1. INTRODUCTION

1.1 Problem Statement

While efforts to characterize software cost and schedule have been advanced by recent papers, variability in the human endeavor of creating or modifying software to meet specific (and evolving) requirements remains. Some of this may be able to be ameliorated by examining aspects of software processes and techniques. This paper explores CMMI, primary software language, and cost-plus contract types as mechanisms to accurately predict effort and schedule by using and comparing initial parameters.

1.2 Deficiencies in Past Studies

Prior studies pairing initial and final records are extremely limited to identify main cost drivers for process oriented categories. A small set of studies [1, 2] have examined the impact of CMMI level 5 on software effort and schedule. One study found higher levels of process maturity increased productivity, decreased development time, and reduced dis-economies of scale [3] although empirical research into this issue, including that in this paper do not establish clear productivity gains or decreased development times with higher CMMI levels. A recent paper and presentation focused on paired initial and final records at an aggregated level with 40 records [6], categorized by government and information technology (IT), not by process orientation.

Another examined paired initial and final records by CSCI level [5] as a whole dataset, only focusing on percentage change between initial and final effort hours categorized by contract type. No studies found in the literature have specially examined the cost impact of government contract types on software effort and schedule.

While degree of software language familiarity is a tunable input parameter in software tools such as COCOMO and SEER-SEM, recent literature searches did not uncover peer-reviewed papers showing direct impacts of specific primary software programming languages on software effort or schedule.

1.3 Purpose of the Study

This study attempts to contribute to the knowledge base by exploring whether process oriented categories accurately estimate effort and duration and if using productivity benchmarks by these categories would be helpful to assess the validity of future DoD data submissions. This study examines the effect of process types on software cost and schedule. It provides statistics and regression models for process-oriented EERs and SERs.

1.4 Significance of the Study

This study will address deficiencies in past studies by:

- Developing simple estimation models by process type instead of complex models with many parameters.
- Using a large and recent dataset to reduce sampling error.
- Normalizing the dataset to improve consistency, and to enable valid comparisons and projections.

1.5 Paper Organization

This research paper is organized into nine sections:

- Section 1 discusses deficiencies in past studies and a proposed solution.
- Section 2 summarizes the scholarly literature of related studies. It highlights previous process-driven analyses and discusses the taxonomy used in this study.
- Section 3 reviews the research method step by step. It briefly explains the survey method, instrumentation,

data normalization framework, and criteria for selecting the best fit models.

- Section 4 describes the data demographics, including segmentation, age of data, size, effort distribution percentages, and productivity benchmarks.
- Section 5 discusses the data analysis results.
- Section 6 presents the effort estimating models.
- Section 7 presents the schedule estimating models.
- Section 8 provides the research conclusions on the basis of the hypotheses. It also highlights the contributions and limitations, and outlines areas for further research.
- Section 9 cites the sources used in the paper.

2. RELATED WORK

Agrawal and Chari [1] studied 37 projects in four CMMI level 5 organizations and found high levels of process maturity reduce the effect of a number of factors expected to impact software development effort, quality, and cycle time. Software size influenced development effort, quality, and cycle time. Models with software size, transformed by natural logarithms, on average, predicted transformed effort and cycle time around 12 percent of the actuals across organizations. The authors posited reduced variance of software development outcomes due to factors other than software size.

Wallshein and Loerch [2] studied thirty CSCI records from software projects completed from 2003 to 2008 certified from CMMI level 5 organizations. Initially estimated software size in either thousands of new lines of code (KNEW) or thousands of total source lines of code (KSLOC), after being transformed by natural logarithms, were found to be reasonable predictors of transformed final, actual software effort. Another variable, untransformed, found to be statistically valid to predict final, actual effort was initially estimated Peak Staff (PS).

Alyahya, Ahmad, and Lee [3] studied the impact of the staged representation of CMMI-based process maturity levels using the Constructive Cost Model (COCOMO) to compute software development effort. They found each higher CMMI maturity level decreased development effort, increased the productivity rate, and reduced the diseconomy of scale. They recommended collecting historical data for each process area to examine the impacts and documenting CMMI representations as staged or continuous.

Rosa, Madachy, Boehm, Clark, and Dean [4] studied the predictor PMAT for software process maturity and determined it should be considered in a schedule model as there was sufficient evidence at $\alpha = 0.05$ that correlations were significant. They also found the predictors thousands of lines of effective source lines of code (KESLOC) and full time equivalent (FTE) personnel as measured by PS to be significant as well. KESLOC and FTE were the two variables documented in the schedule models for application types.

Lanham, Wallshein, Rosa, and Popp [5] examined initial and final effort hours at the CSCI level and determined initial hours, requirements count, and PS represented statistically significant independent variables. In one of the first published papers on software development segregated by contract type, cost-plus contract types were examined for percent change between initial and final effort hours. Cost-Plus Award Fee (CPAF) contracts and Cost-Plus Fixed Fee (CPFF) contracts demonstrated significantly different average values. CPAF, typically granted for large research and development projects with an award fee to motivate the contractor to meet government's desired performance objectives, had the highest mean growth value between initial and final effort hours.

Rosa, Boehm, Clark, Madachy, Jones, McGarry, Lanham, and Wallshein [6] studied initially estimated PS and requirements as predictors of the actual effort for 40 development projects, with rolled-up CSCIs, at the early elaboration phase. Actual effort was measured by person month (PM) using the COCOMO factor of 152 hours per PM. Initially estimated effort in PM was found to highly significant to predict actual software development duration.

3. RESEARCH METHOD

3.1 Research Question

This study will address the following questions:

Is CMMI, primary software language, and contract type a good predictor of software engineering labor?

Does software development duration relate to size and staffing levels, when grouped by CMMI, primary software language, and contract type?

3.2 Quantitative Method

Second order data was used in the methods in this study to analyze effort and cost drivers of software development projects. A non-random sample was used since the researchers had access to records showing effort in all the areas described by IEEE 12207, Systems and software engineering -- Software life cycle processes, for software development.

3.3 Population and Sample

The sample was 204 paired software projects implemented for the United States Department of Defense (DoD) based on Mr. Nick Lanham's pairing algorithm. These projects were completed during the time period from 2002 to 2013. The number of projects considered for effort analysis was 219. After extensive analysis over the course of this year, fifteen records were determined to be outside the scope of analysis. Reasons included being initially recorded as a single CSCI when the paired submission was for multiple CSCIs. For benchmarking, box plots of actual effort hours to initial requirements are shown for the analyzed process orientation types.

3.4 Instrumentation

Data were compiled from a questionnaire containing over 20 items. The data collection questionnaire used in the study was obtained from the *Software Resource Data Report (SRDR)* questionnaire [16]. The source questionnaire entitled "SRDR Sample Formats" can be downloaded from the Defense Cost Analysis Resource Center (DCARC) website:

http://dcarc.cape.osd.mil/Files/Policy/2011-SRDRInitial.pdf

http://dcarc.cape.osd.mil/Files/Policy/2011-SRDRFinal.pdf

http://dcarc.cape.osd.mil/Files/Policy/Initial_Developer_Report.xlsx

http://dcarc.cape.osd.mil/Files/Policy/Final_Developer_Report.xlsx

Data on product size, effort, schedule and product attributes like required reliability, software process maturity, were recorded and new fields were added. Multiple personnel within the Naval Center for Cost Analysis researched and collected data on contract type. Lists of contract types were obtained from multiple government sources [7 8].

3.5 Data Normalization

An objective of data normalization is to improve data consistency, so that comparisons and projections are more valid. The software data set in this study was normalized using two steps:

3.5.1 Converting to Logical SLOC Count

It is considered best practice [8] to use logical SLOC as the standard counting rule for software cost estimation. Several projects were reported in either Physical or Non-Commented Source Statements (NCSS). Those projects were converted into Logical SLOC using empirical factors from recent studies [9]:

Conversion Factor
Logical SLOC = 0.66 x NCSS SLOC
Logical SLOC = 0.33 x Physical SLOC

3.5.2 Data Grouping

According to the United States Government Accountability Office [15], it is considered best practice to normalize data by similar characteristics. Products created within organizations having similar CMMI levels use similar processes certified by an outside expert. Products developed with similar primary programming languages adhere to certain rulesets and processes for the software to be compiled and execute. Furthermore, products developed under cost-plus contract types transfer the risk to the government, allowing for the development to be sanctioned by the servicing organization since the government is financially responsible. Data for contract types not explicitly labeled as cost-plus were scarce (i.e., having less than 12 records per subset) or unknown and were excluded from further analysis within the scope of this study.

To reduce variation and ensure valid comparisons, the process oriented categories are shown in the tables below.

The first table discusses the CMMI levels.

Table 1 Taxonomy for Process Maturity [10]

Process Maturity	Symbol	Definition
Defined	CMMI 3	A standard software process meets the organizations specific needs. Attention is paid to documentation, standardization, and integration. Projects follow defined process even under schedule pressure. Management recognizes these processes are the quickest route to completion.
Managed	CMMI 4	Processes are predictable. Detailed, quantitative measurements of process and product quality are collected. Management can adjust and adapt the process to specific projects without losing quality or deviating from specifications.
Optimizing	CMMI 5	Processes are continuously improving. Processes are improved through quantitative feedback and shared ideas. Mangers introduce innovative processes to better serve the organizations particular needs. Pilot projects are common.

The next table discusses the primary software language types.

Table 2 Taxonomy for Primary Software Language Types

Aggregated Primary Software Language	Symbol	Definition
Ada, Ada-83, Ada-95	Ada	Named after Ada Lovelace, a nineteenth century mathematician, and commissioned by the US DoD to defense contractor CII Honeywell Bull, Ada is a structured, object-oriented, high-level language unique in providing support for real-time embedded software with tasks and synchronous messages.[11]
C#	C#	Per http://searchwindevelopment.techtarget.com/definition/C, C# (pronounced "C-sharp") is an object-oriented programming language from Microsoft based on C++ with features similar to Java. C# is designed to work with Microsoft's .Net platform to facilitate exchange of information and services over the Web, and to enable developers to build highly portable applications. [12]
C/C++	C/C++	This hybrid of C and C++ has the least precise definition in the dataset. C was created in conjunction with the UNIX operating system and is the forerunner of C++ although not an object-oriented language itself. Both C and C++ are designed for use by systems programmers, with C++ having an object-oriented style.[11]

Aggregated Primary Software Language	Symbol	Definition
C++	C++	Per http://www.cplusplus.com/info/description/, C++ is an open ISO-standardized language, since 1998. C++ compiles directly to a machine's native code, allowing it to run fast, if optimized. C++ allows type conversions to be checked either at compile-time or at run-time. Most C++ type checking is static. C++ supports procedural, generic, and object-oriented programming paradigms. Code that exclusively uses C++'s standard library will run on many platforms with few to no changes. C++, as a language directly built off C, is compatible with almost all C code. [13]
Java	Java	Per http://searchsoa.techtarget.com/definition/Java, Java is a programming language for use in the distributed environment of the Internet. It was designed to have the "look and feel" of the C++ language, but it is simpler to use than C++ and enforces an object-oriented programming model. Java can be used to create complete applications that may run on a single computer or be distributed among servers and clients in a network. It can also be used to build a small application module or applet for use as part of a Web page. [14]

Contract types are described below. [7 8]

Table 3 Taxonomy for Cost-Plus Contract Types

Contract Types	Symbol	Definition
Cost Plus Award Fee	CPAF	The contract level of effort is uncertain and it is not feasible or effective to negotiate an adjustment formula. The likelihood of meeting objectives can be enhanced by a clear subjective fee plan established by the government.
Cost Plus Fixed Fee	CPFF	Cost uncertainty is extremely high. Establishment of predetermined targets and incentive sharing arrangements could result in a final fee out of alignment with the actual work.
Cost Plus Incentive Fee	CPIF	The cost uncertainties are so great that any fixed-price contract would force the contractor to accept an unreasonable risk, but reasonable targets and formulas for sharing costs may be negotiated.

3.6 Variables in the Study

The variables considered in the study are identified in Table 2. The variable selection procedure is described in Section 5.

Table 4 Variables in the Study

Variable	Symbol	Definition
Effort Hours	EH	Software engineering effort (in Effort Hours). Includes: • software requirements analysis, • architecture/detailed design • code and unit testing, • systems/software integration, • qualification test, • development test & evaluation, • Other direct support: documentation and configuration management, quality assurance, software verification & validation, software review and audit, and software problem resolution.
Software Development Duration	SCHED	The time required to complete all activities up to the point of development test & evaluation (DT&E) by the vendor's implementation team.
Initially Estimated Peak Staffing Level	PS	This is the peak staffing number of full time equivalent (FTE) people employed by the vendor's implementation team involved in the software development.
Initially Estimated New Logical Statements (LS) of Code	NEW	Amount of new code developed by the vendor for the software configuration item, converted to logical statements.
Initially Estimated Source Lines of Code (SLOC) in LS	SLOC	Amount of total source lines of code (SLOC) developed by the vendor for the software configuration item, converted to logical statements.
Initially Estimated Requirements	REQ	Total number of requirements for the configuration item, combining software requirements with external, interface requirements. These requirements are at various stages of maturity and elaboration, with lower levels expected for cost-plus contract types.

3.7 Effort Model Forms

Although several model forms were examined for each specified process type containing 12 or more observations, the predominant equation displaying consistently better goodness of fit was the following:

$$aEH_i = A \times (eEH_i^B) \qquad \textit{Equation (1)}$$

$$aEH_i = A \times (eNEW_i^B) \qquad \textit{Equation (2)}$$

Where:

aEH	=	Actual Engineering labor in effort hours
eEH	=	Estimated Engineering labor in effort hours
i	=	Process-orientation type (Tables 1 – 3)
eNEW	=	Estimated amount of new code in logical statements
A	=	*Productivity constant (a.k.a. coefficient)*
B	=	*S*caling factor expressing degree of the diseconomy of scale (a.k.a. exponent)

The scaling influence is found in the exponent, B. An estimated value for B < 1.0 indicates an economy of scale. An estimated value of B > 1.0 indicates a diseconomy of scale.

3.8 Schedule Model Form

A non-linear model form was used for each application type containing 12 or more observations.

$$aSCHED_i = A \times (eSCHED_i^B) \qquad \textit{Equation (3)}$$

$$aSCHED_i = A \times (eREQ_i^B) \qquad \textit{Equation (4)}$$

Where:

aSCHED	=	Actual Time (in months) to develop the Software Product
i	=	Process orientation type
A	=	Duration constant
eSCHED	=	Estimated Time (in months) to develop the Software Product
eREQ	=	Estimated requirements
B	=	Scaling factor to account for changing productivity

The primary schedule equation predicts the duration of software development phase as a function of estimated duration. The schedule begins with software requirements analysis, ends at the completion of qualification test, and precedes development test & evaluation phase.

3.9 Model Validity, Accuracy, and Selection

The measures for assessing the validity and accuracy of the effort and schedule model forms are described in Table 3 and Table 4 respectively. The best model form for a given application type, is the one that surpasses the criterion shown in Table 5.

Table 5 Model Validity Measures

Measure	Symbol	Description
T-test	T-stat	Provides a measure of the significance of the predictor variables in the regression model. The variable is significant when the t-stat is greater than the two-tailed value, given the degrees of freedom and coefficient alpha (α = 0.05)

Table 6 Model Accuracy Measures

Measure	Symbol	Description
Standard Error of the Estimate	SE	Measures the average amount of variability remaining after the regression. Standard Error of the Estimate is a measure of the difference between the observed and model estimated effort.
Coefficient of Determination	R^2	Shows how much variation in dependent variable is explained by the regression equation.
Coefficient of Variation	CV	Percentage expression of the standard error compared to the mean of the dependent variable. A relative measure allowing direct comparison among models.
Mean Absolute Deviation	MAD	Measures the average percentage by which the regression overestimates or underestimates the observed actual value.
Mean Magnitude of Relative Error	MMRE	Nearly identical to MAD, this measures the average magnitude of relative error (the absolute value of the actual value subtracted from the predicted value is divided by the actual value).
Predictive Accuracy	PRED(X)	X is found in the literature at 25 or 30 percent with the predictive accuracy computed from the number of records having a magnitude of relative error less than or equal to X. A higher PRED value is better.

Table 7 Model Selection Criterion

Measure	Criterion
MAD	≤ 45%
CV	≤ 45%
R^2	≥ 55%
t-test	> Two tailed critical value (DF, α = 0.05)

4. Data Demographics

The dataset began with 204 records selected from 2015 after determining which of the 219 paired records used in [5] should continue to be analyzed, given the effort hour distribution verifications per Table 4, of the software engineering activities in IEEE 12207, prior to record pairing. As these records were analyzed from 2015 to 2016, outliers were scrubbed from the dataset. Outliers were compared against data subset populations, double checked with original entries and programs'

documentation. The table below lays out the 2624 total CSCI records processed down into the 204 total CSCI records analyzed.

Table 8 Software Dataset

Dataset Records
2624 Total CSCI Records
911 Completed Program / Build CSCI Records
403 Completed CSCIs with Software Engineering Activity Break-Outs
219 Completed Paired CSCI Records
204 Completed Paired CSCI Records After Removing Outliers

Figure 1shows the age of the software projects by delivery year. The age of data does not pose a challenge to validity as these are recent projects completed between 2002 and 2013.

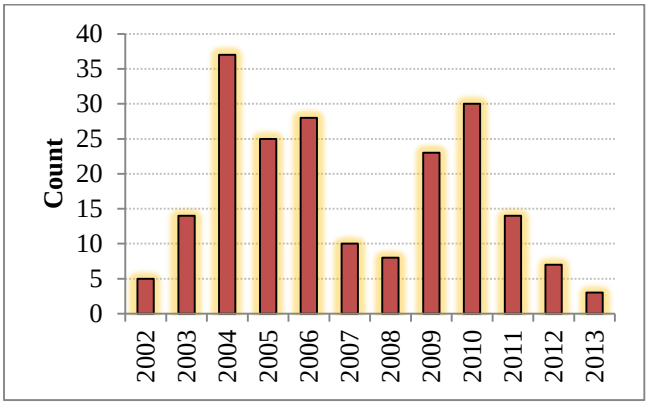

Figure 1 Age of Data

Figure 2 shows the break-out of aggregated primary software programming languages for the 204 CSCI records. Each CSCI recorded which programming language was considered to be the primary, principal, or majority language. These languages were grouped according to categories shown below. The C language represented by the aggregated categories C/C++ and C++ alone account for sixty-seven percent of the 204 CSCI records in the analyzed dataset.

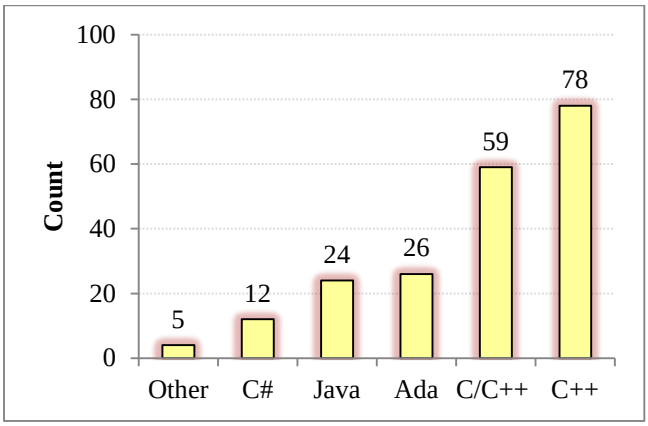

Figure 2 Aggregated Primary Language

Figure 3 shows the break-out of CMMI levels for the CSCI records. Although 20 records were not specified as to their CSCI level, there were five records specifying CMMI level 2, where processes become defined, documented, and repetitive. Ninety percent of the 204 paired CSCI records analyzed were CMMI level 3 or CMMI level 5.

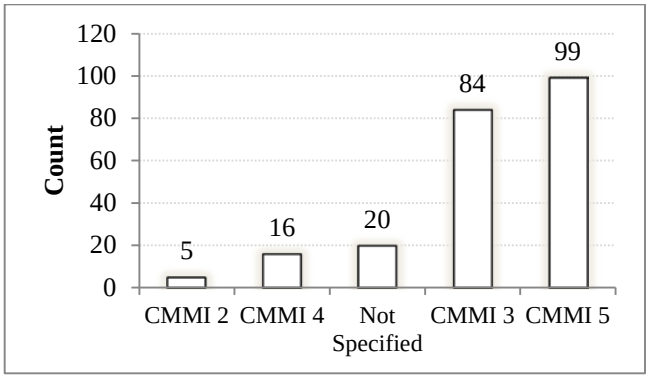

Figure 3 Capability Maturity Model Integrated (CMMI) Levels

Figure 4 shows the break-out of data by contract type. FFP, IDIQ, and FPIF stand for Firm Fixed Price, Indefinite Delivery Indefinite Quantity, and Fixed Price Incentive Fee contracts, respectively. Seventeen records were unable to be categorized as to contract vehicle type. Eighty-four percent of the CSCI records had cost-plus contracts where the government assumed the risk of the contract.

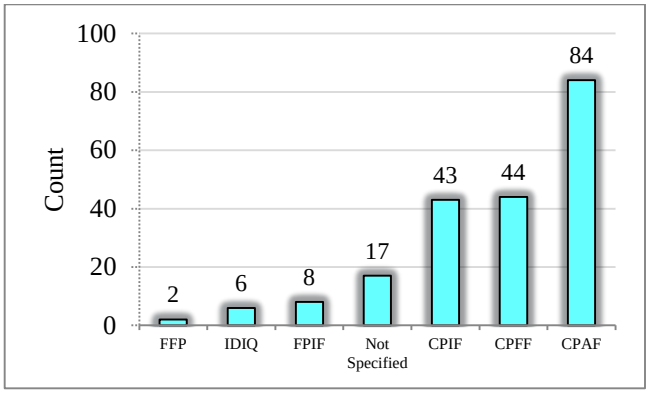

Figure 4 Contract Type

Figure 5 shows contract types from www.acq.osd.mil/dpap/ [7] where FPEPA stands for Fixed Price Economic Price Adjustment. Cost-plus contracts figure predominately when systems are in early acquisition phases.

COST RISK AND CONTRACT TYPE						
Cost Risk	**High**					**Low**
Requirement Definition	**Vague**					**Well-defined**
Production Stages	Concept Studies & Basic Research	Exploratory Development	Test/ Demonstration	Full-scale Development	Full Production	Follow-on Production
Contract Type	**Varied**	**CPFF**	**CPIF, FPIF**	**CPIF, FPIF, FFP**	**FFP, FPIF, FPEPA**	**FFP, FPIF, FPEPA**

Figure 5 Contract Type Risk

The process-oriented categories with a preponderance of data are C/C++ languages, cost-plus contracts, and CMMI levels 3 and 5.

5. Data Analysis

This section is divided into three parts. The first displays the correlation analysis for selecting the most significant variables. The second shows the ranges in the data using box plots. The third identifies the best fit model using CO$TAT Log Linear regression [19]. Log linear regression was determined to be the best fit after exploring and evaluating linear and non-linear models with the most significant variables.

5.1 Correlation Analysis

Figure 6 is the Pearson correlation for actual effort hours where all the subsets are combined.

.

Pearson Correlation for Actual Effort Hours	Actual Duration	Estimated Effort Hours	Estimated Requirements	Estimated Duration	Estimated Peak Staff	Estimated New SLOC in LS	Estimated SLOC in LS
C#	0.38	0.92	-0.02	0.46	0.79	0.84	0.74
Java	0.00	0.79	0.61	-0.09	0.56	0.34	0.19
Ada	0.26	0.92	0.35	0.27	0.50	0.71	0.27
C/C++	-0.07	0.82	0.48	-0.05	0.71	0.57	0.36
C++	0.12	0.93	0.35	0.11	0.62	0.57	0.39
CMMI 3	0.28	0.88	0.42	0.23	0.74	0.50	0.48
CMMI 4	0.31	0.95	0.21	0.18	0.04	0.59	0.74
CMMI 5	-0.12	0.85	0.33	-0.11	0.67	0.57	0.17
CPAF	0.10	0.81	0.55	0.22	0.77	0.56	0.47
CPFF	0.16	0.90	0.60	0.21	0.59	0.76	0.28
CPIF	0.08	0.88	0.58	-0.06	0.57	0.45	0.51

Figure 6 Pearson Correlation to Actual Effort Hours

Figure 7 is the Spearman Rho correlation for actual effort hours.

Spearman Correlation for Actual Effort Hours	Actual Duration	Estimated Effort Hours	Estimated Requirements	Estimated Duration	Estimated Peak Staff	Estimated New SLOC in LS	Estimated SLOC in LS
C#	0.49	0.84	0.54	0.50	0.69	0.90	0.78
Java	-0.20	0.77	-0.18	-0.18	0.68	0.60	0.62
Ada	0.12	0.92	0.02	0.22	0.78	0.81	0.43
C/C++	-0.12	0.89	-0.05	-0.12	0.75	0.76	0.49
C++	0.20	0.95	0.23	0.15	0.74	0.78	0.65
CMMI 3	0.29	0.92	0.38	0.28	0.77	0.73	0.71
CMMI 4	0.39	0.88	0.06	0.22	0.24	0.78	0.65
CMMI 5	-0.03	0.86	-0.03	-0.02	0.71	0.72	0.48
CPAF	0.16	0.90	0.16	0.24	0.79	0.78	0.72
CPFF	0.22	0.86	0.33	0.28	0.62	0.73	0.70
CPIF	0.14	0.88	0.25	0.04	0.66	0.61	0.55

Figure 7 Spearman Rho Correlation for Actual Effort Hours

Figure 8 is the Pearson correlation for actual duration where all the subsets are combined.

Pearson Correlation for Actual Duration	Estimated Effort Hours	Estimated Requirements	Estimated Duration	Estimated Peak Staff	Estimated New SLOC in LS	Estimated SLOC in LS
C#	0.45	0.35	0.91	0.34	0.54	0.54
Java	0.01	-0.19	0.94	-0.16	-0.15	0.05
Ada	-0.04	0.11	0.87	-0.37	0.41	0.16
C/C++	-0.10	0.31	0.69	0.10	0.12	0.03
C++	0.15	-0.11	0.69	-0.05	0.02	0.22
CMMI 3	0.35	-0.09	0.83	0.03	0.27	0.24
CMMI 4	0.09	0.83	0.91	0.57	0.40	0.59
CMMI 5	-0.13	0.17	0.61	-0.16	-0.02	0.06
CPAF	0.10	-0.16	0.58	-0.09	0.23	0.17
CPFF	0.17	0.13	0.90	-0.06	0.14	0.33
CPIF	0.03	0.29	0.88	-0.01	0.12	0.20

Figure 8 Pearson Correlation to Actual Duration

Figure 9 is the Spearman Rho correlation for actual effort hours.

Spearman Correlation for Actual Duration	Estimated Effort Hours	Estimated Requirements	Estimated Duration	Estimated Peak Staff	Estimated New SLOC in LS	Estimated SLOC in LS
C#	0.54	0.67	0.78	0.40	0.62	0.73
Java	-0.18	0.60	0.90	-0.38	-0.12	-0.03
Ada	0.02	0.49	0.81	-0.24	0.21	0.11
C/C++	-0.05	0.45	0.78	0.06	0.02	0.11
C++	0.23	0.71	0.79	0.03	0.10	0.27
CMMI 3	0.38	0.61	0.86	0.14	0.32	0.31
CMMI 4	0.06	0.47	0.96	0.57	0.45	0.66
CMMI 5	-0.03	0.56	0.75	-0.13	-0.01	0.03
CPAF	0.16	0.67	0.71	-0.04	0.22	0.11
CPFF	0.33	0.69	0.88	-0.03	0.08	0.26
CPIF	0.25	0.65	0.85	0.08	0.24	0.22

Figure 9 Spearman Rho Correlation for Actual Duration

Correlation values are color-coded where the darker colors indicate higher levels and lighter colors indicate lower levels.

Actual effort hours are correlated to estimated effort hours. For the C# aggregated primary programming language, actual effort hours are correlated to estimated new code, estimated SLOC, and estimated PS. For Java, actual effort hours are correlated to PS and SLOC. For Ada, C/C++, and C++ as the primary aggregated language, actual effort hours are correlated to PS and new code in logical statements. CMMI 3's actual effort hours are correlated to PS, SLOC, and new code in logical statements. CMMI 4's actual effort hours are correlated to SLOC and new code whereas CMMI 5' actual effort hours are correlated to PS and new code. CPAF actual effort hours are correlated to PS, new code, and SLOC; CPFF's are correlated to new code; and CPIF's are correlated to PS. Estimated effort hours are positively correlated to PS at 62% or greater with Pearson correlation, Spearman Rho correlation, or both, except for CMMI level 4.

Actual duration is correlated to estimated duration (in months). For the C# aggregated primary language, actual duration is correlated to SLOC. For the C++ language, actual duration is correlated to estimated requirements [eREQ]. CMMI 4 actual duration is correlated to eREQ and SLOC. CPAF actual duration is correlated to eREQ and CPIF's is correlated to new code and SLOC in logical statements.

The following conclusions can be drawn from the correlation analysis:

- The predictors, eEH and estimated new/SLOC in logical statements, should be considered in the effort model.

- The predictors, eSCHED and eREQ, should be considered in the schedule model.

5.2 Data Ranges

Actual effort hours have the following ranges for the subsets.

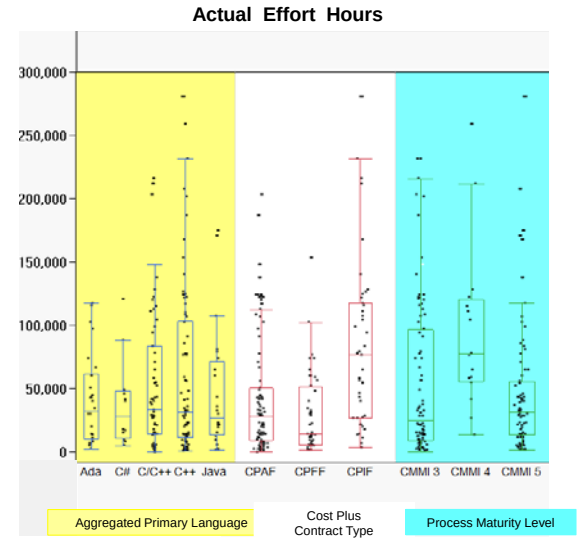

Figure 10 Actual Effort Hour Ranges

The data ranges in the subsets showed the most variation in actual effort hours for the C++ primary programming language and CPIF contract type. Box plots for actual effort hours with process maturity categories showed the maximum number of effort hours increasing with increasing maturity. Median values, represented by the vertical bars, were similar between CMMI levels 3 and 5 although the range for CMMI level 3 was much wider. CMMI level 4 had a much higher median than either CMMI level 3 or CMMI level 5. The percent change in effort hours is shown in Figure 11.

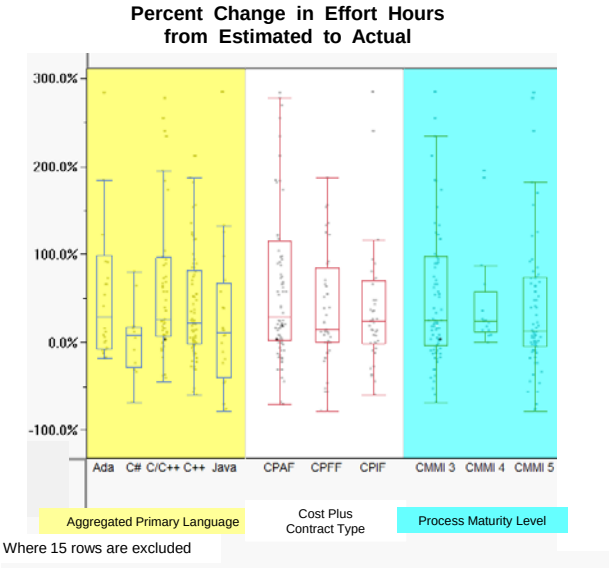

Figure 11 Effort Hours (Estimated to Actual) Percent Change

Percent change in effort hours from estimated to actual, for the majority of the data, ranged from -100% to over 200% for all the subsets except CPAF contract type and CMMI level 3. Median values for all the subsets ranged between 5% and 40%.

Actual duration in months is shown in Figure 12.

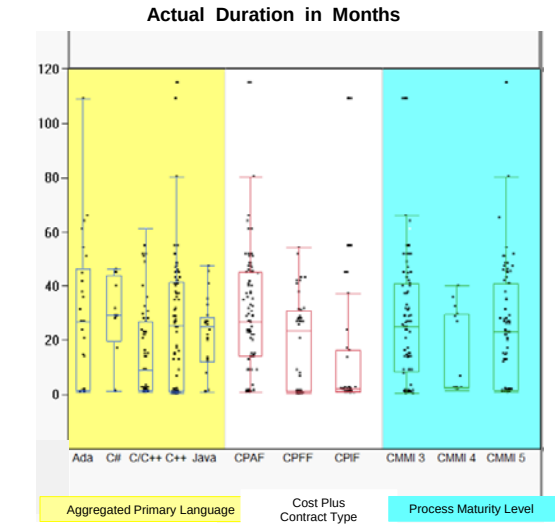

Figure 12 Actual Duration in Months

Ada had the most variation among language types. CPAF contract type had the most variation among contract types. CMMI level 5 had the most variation among process maturity types.

Figure 13 shows the percent change between estimated and actual duration.

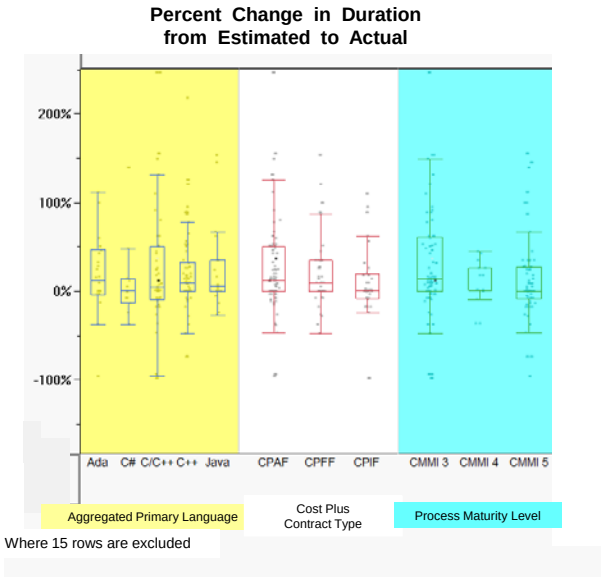

Figure 13 Duration (Estimated to Actual) Percent Change

C/C++ had the highest percent change in duration between estimated and actual months among language types. CPAF had the highest percent change in duration among contract types. And CMMI level 3 had the highest percent change in duration among process maturity types. All the subsets median values ranged between 0% and 15% change in duration.

Figure 14 shows a box plot of the analyzed process oriented categories of the actual effort hours divided by the initially estimated requirements to use as a productivity benchmark.

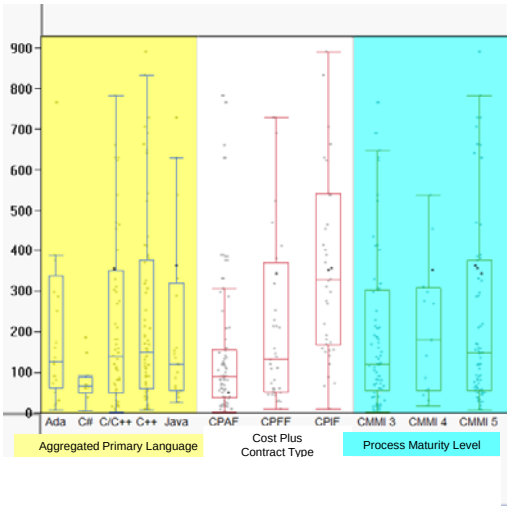

Figure 14 Actual Effort Hours per Estimated Requirements

5.3 Regression

This study used regression in the JMP tool [18] and CO$TAT tool [19] to determine which effort model fit the data best. Regressions for actual effort hours began with estimated effort hours. Regressions for actual duration began with estimated duration.

6. EFFORT MODEL RESULTS

6.1 Effort Model Functional Forms

The resulting models are shown in Table 9 below. Each model predicts effort (in hours) for a particular process category type. There are no models where the number of data points were lower than 12. Where there were blanks or zeroes for initially estimated parameters, those records were removed from the analysis. Using initially estimated effort hours, all equations exhibit economies of scale as the exponent for the independent variable is less than one.

Table 9 Effort Model Functional Forms

Model	Subset	Records	Equation
1	C#	12	aEH = 18.97 * eNEW ^0.70
2	Ada	25	aEH = 52.91 * eEH ^0.64
3	C#	12	aEH = 4.64 * eEH ^0.84
4	C/C++	55	aEH = 6.82 * eEH ^0.85
5	C++	69	aEH = 1.55 * eEH ^0.98
6	CPAF	74	aEH = 8.53 * eEH ^0.82
7	CPFF	40	aEH = 1.27 * eEH ^0.99
8	CPIF	43	aEH =18.8 * eEH ^0.75
9	CMMI 3	77	aEH = 4.46 * eEH ^0.88
10	CMMI 4	16	aEH = 7.91 * eEH ^0.84
11	CMMI 5	90	aEH = 9.08 * eEH ^0.80

6.2 Effort Model Usage and Limitations

The following guidelines are for using the regression models listed in Table 9:

- Model (1) predicts effort for the aggregated C# primary programming language with the initial variable, estimated new code. The model is applicable to project estimates ranging between 3.5 and 192 thousands of lines of new code in logical statements.
- Model (2) predicts effort for the aggregated Ada primary programming language. The model is applicable to project estimates ranging between 320 and 133,855 effort hours.
- Model (3) predicts effort for the aggregated C# primary programming language. The model is applicable to project estimates ranging between 4620 and 133,280 effort hours.
- Model (4) predicts effort for the aggregated C/C++ primary programming language. The model is applicable to project estimates ranging between 184 and 170,807 effort hours.
- Model (5) predicts effort for the aggregated C++ primary programming language. The model is applicable to project estimates ranging between 520 and 209,616 effort hours.
- Model (6) predicts effort for CPAF projects. The model is applicable to project estimates ranging between 575 and 169,583 effort hours.

- Model (7) predicts effort for CPFF projects. The model is applicable to project estimates ranging between 1896 and 101,665 effort hours.

- Model (8) predicts effort for CPIF projects. The model is applicable to project estimates ranging between 2235 and 191,013 effort hours.

- Model (9) predicts effort for CMMI level 3 projects. The model is applicable to project estimates ranging between 575 and 209,616 effort hours.

- Model (10) predicts effort for CMMI level 4 projects. The model is applicable to project estimates ranging between 11702 and 207,968 effort hours.

- Model (11) predicts effort for CMMI level 5 projects. The model is applicable to project estimates ranging between 1104 and 225,304 effort hours

6.3 Effort Model Validity

Table 10 below shows the model validity results. All regression models' independent variables are significant as the t-statistics exceed the two-tailed critical values, given the coefficient alpha (0.05) and degrees of freedom (DF). The DF were equal to the number of records minus two: one for the equation intercept and one for the independent variable.

Table 10 Effort Model Validity Results

Model	Subset	Records	Independent Variable T-Statistic (Coef/SD)	P-Value
1	C#	12	4.8554	0.0007
2	Ada	25	9.0249	0.0000
3	C#	12	6.3417	0.0000
4	C/C++	55	14.5753	0.0000
5	C++	69	23.5838	0.0000
6	CPAF	74	16.2532	0.0000
7	CPFF	40	9.7535	0.0000
8	CPIF	43	8.9502	0.0000
9	CMMI 3	77	18.7147	0.0000
10	CMMI 4	16	8.2148	0.0000
11	CMMI 5	90	14.8412	0.0000

6.4 Effort Model Reliability

Table 11 shows the accuracy test results. All effort models are reliable as their CV are less than or equal 35%. MAD was only less than or equal 45% for the aggregated primary programming language effort models (Models 1 through 5) and for the CMMI level 4 effort model (Model 10). MAD was below 60% for all models. A published standard for MMRE is less than or equal to 25% and for PRED (30) is 75% or more [17]. All effort models except for the CMMI level 4 effort model fall below this relatively high published standard.

Table 11 Effort Model Reliability Results

Model	R2 in Unit Space	SE	RMS of % Errors	MAD	CV (MAD Res/Avg Act)	MMRE	PRED(30)
1	0.69	19557	0.39	0.29	0.29	0.29	0.67
2	0.78	16825	0.55	0.40	0.29	0.40	0.48
3	0.81	15391	0.64	0.38	0.25	0.38	0.58
4	0.66	32218	0.67	0.45	0.32	0.45	0.45
5	0.85	26023	0.52	0.36	0.24	0.36	0.52
6	0.64	28742	0.78	0.51	0.35	0.51	0.46
7	0.78	15281	1.11	0.58	0.31	0.58	0.50
8	0.70	35749	0.84	0.49	0.28	0.49	0.60
9	0.76	29210	0.75	0.49	0.29	0.49	0.42
10	0.88	22811	0.27	0.22	0.17	0.22	0.75
11	0.69	28265	1.03	0.55	0.34	0.55	0.47

R^2 = coefficient of determination; SE = standard error of estimate; RMS = Root Mean Square of the percentages of error; MAD = mean absolute deviation; CV = coefficient of variation; MMRE = Mean Magnitude of Relative Error (MRE); PRED(30) = Predictive accuracy is the percent of the number of records at or above a value of 30% magnitude of relative error

7. SCHEDULE MODEL RESULTS

7.1 Schedule Model Functional Forms

The resulting models are shown in Table 12. Each model predicts schedule (in months) for a particular process oriented type.

Table 102 Schedule Model Functional Forms

Model	Subset	Records	Equation
12	Ada	24	aSCHED = 2.14 * eSCHED ^0.76
13	C#	12	aSCHED = 1.55 * eSCHED ^0.87
14	C++	69	aSCHED = 1.44 * eSCHED ^0.92
15	Java	24	aSCHED = 2.4 * eSCHED ^0.75
16	CPFF	40	aSCHED = 1.04 * eSCHED ^1.07
17	CPIF	43	aSCHED = 1.40 * eSCHED ^0.80
18	CMMI 3	77	aSCHED = 2.23 * eSCHED ^0.76
19	CMMI 4	16	aSCHED = 1.05 * eSCHED ^1.02
20	CMMI 5	90	aSCHED = 1.44 * eSCHED ^0.86
21	CMMI 4	16	aSCHED = 0.01 * eREQ ^0.99

7.2 Schedule Model Usage and Limitations

The following guidelines are for using the schedule models listed in Table 12:

- Model (12) predicts schedule for the aggregated Ada primary programming language. The model is applicable to project estimates ranging between 0.2 and 100.1 months.
- Model (13) predicts schedule for the C# primary programming language. The model is applicable to project estimates ranging between 0.5 and 52 months.
- Model (14) predicts schedule for the C++ primary programming language. The model is applicable to project estimates ranging between 0.2 and 83 months.
- Model (15) predicts schedule for the Java primary programming language. The model is applicable to project estimates ranging between 0.4 and 47.2 months.
- Model (16) predicts schedule for CPFF projects. The model is applicable to project estimates ranging between 0.4 and 41 months.
- Model (17) predicts schedule for CPIF projects. The model is applicable to project estimates ranging between 0.2 and 100.1 months.
- Model (18) predicts schedule for CMMI level 3 projects. The model is applicable to project estimates ranging between 0.3 and 100.1 months.
- Model (19) predicts schedule for CMMI level 4 projects. The model is applicable to project estimates ranging between 0.3 and 46.8 months.
- Model (20) predicts schedule for CMMI level 5 projects. The model is applicable to project estimates ranging between 0.2 and 57 months.
- Model (21) predicts schedule for CMMI level 4 projects. The model is applicable to projected requirement counts ranging between 97 and 3,685.

7.3 Schedule Model Validity

Table 13 shows the schedule model validity results. All regression models are significant as the t-statistics for the independent variables exceed the two-tailed critical values.

Table 11 Schedule Model Validity Results

Model	Subset	Records	Independent Variable T-Statistic (Coef/SD)	P-Value
12	Ada	24	6.3190	0.0000
13	C#	12	14.3584	0.0000
14	C++	69	20.0534	0.0000
15	Java	24	9.7523	0.0000
16	CPFF	40	26.9108	0.0000
17	CPIF	43	11.5055	0.0000
18	CMMI 3	77	11.8238	0.0000
19	CMMI 4	16	23.6736	0.0000
20	CMMI 5	90	17.6696	0.0000
21	CMMI 4	16	5.9674	0.0000

7.4 Schedule Model Reliability

Table 14 shows the accuracy test results. All but one of the schedule models are reliable as their CV are within the threshold of less than or equal to 45%. CPIF projects' schedule model is over 45%. It is 3% over at 48%. MAD values are larger for the schedule models ranging from a low of 15% to a high over 100% at 139%. Only half of the schedule models are within the range of the MAD threshold. These five models were Model (13) for C#, Model (14) for C++, Model (15) for Java, Model (16) for CPFF, and Model (17) for CMMI level 4.

Table 14 Schedule Model Reliability Results

Model	R2 in Unit Space	SE	RMS of % Errors	MAD	CV (MAD Res / Avg Act)	MMRE	PRED(30)
12	0.63	16.9	4.73	1.39	0.35	1.39	0.46
13	0.81	6.6	0.30	0.23	0.16	0.23	0.75
14	0.69	11.9	0.63	0.37	0.26	0.37	0.57
15	0.87	4.9	0.46	0.31	0.16	0.31	0.67
16	0.78	8.0	0.34	0.23	0.21	0.23	0.68
17	0.64	16.3	4.79	1.05	0.48	1.05	0.47
18	0.63	14.1	5.31	1.31	0.32	1.31	0.48
19	0.78	7.1	0.23	0.15	0.23	0.15	0.94
20	0.55	12.6	3.32	0.85	0.34	0.85	0.50
21	0.63	9.2	1.07	0.66	0.42	0.66	0.38

R^2 = coefficient of determination; SE = standard error of estimate; RMS = Root Mean Square of the percentages of error; MAD = mean absolute deviation; CV = coefficient of variation; MMRE = Mean Magnitude of Relative Error (MRE); PRED(30) = Predictive accuracy is the percent of the number of records at or above a value of 30% magnitude of relative error

8. CONCLUSION

8.1 Primary Findings

This study resulted in the following findings:

1. Pearson and Spearman Rho correlation analysis revealed that the most significant factor influencing software development effort in the process oriented categories was estimated effort except for the C# primary language. For C#, new code in logical statements was a significant and stand-alone factor.

2. Pearson and Spearman Rho correlation analysis also revealed that the significant factor influencing software development schedule was estimated duration across the process oriented categories. For CMMI level 4 projects, estimated requirements were a significant and stand-alone factor.

3. The median values for percent change in effort hours ranged between 5 to 40% whereas the median values for percent change in duration ranged from 0 to 15%. For effort hour and duration percent change, CPAF and CMMI level 3 had the highest rate of variation among all the other process oriented category subsets.

4. The regression results show the effect of estimated effort hours on actual effort hours. All models passed the t-statistic and p-value validity tests. Models 1 through 5 for aggregated language types, including Model 1 with new code in logical statements as the sole independent variable, along with Model 10 for CMMI level 4 passed all reliability tests. R2 in unit space for all models was above 60%.

5. The regression results show that the effect of estimated duration on software development schedule is significant. For CMMI level 4, estimated requirement counts were also significant. Models 12 through 21 passed the model validity test. Only Model 13 for C#, Model 14 for C++, Model 15 for Java, Model 16 for CPFF, and Model 17 for CMMI level 4 passed the model reliability tests.

6. The productivity benchmark box plot by aggregated language type, contract type, and CMMI level 3 through 5 may be used as a guide to evaluating whether actual effort hours per initially estimated requirement counts are within the ranges of historical data for government projects.

8.2 Challenges to Validity

Although some of the models were deemed reliable, they still have a few limitations:

1. A larger dataset (>400 projects) is required to increase model validity and accuracy. A future investigation should attempt to control for the impact of external factors such as operating environment, application domain, program phase, and maturity of estimated requirements and code counts.

2. A non-random sample was used as the researchers had access to names in the population and the selection process for participants was based on the non-zero or non-blank valued input parameters. This process limits the ability to generalize to a population.

3. Pairwise correlation using Pearson correlation and Spearman Rho correlation was used over Structured Equation Modeling as the sponsor, US Navy, does not approve the use of “R” or any other open source statistical package.

4. The cost-plus contract types represent areas of work made possible with funding from government priorities which may not translate to for-profit industry endeavors.

8.3 Future Research

The following topics may be considered for future research:

1. Continue to collect data on estimated effort and schedule to compare to actual effort and schedule and publish benchmarks by well-documented process oriented and product oriented categories.

2. Instead of developing multiple models, use collected data to create composite software effort and software schedule estimating models using quantitative values for process and product oriented categories.

3. Compare local data to government data using the process oriented types: aggregated primary software programming language, contract type, and process maturity type (and comparing staged CMMI levels to continuous CMMI levels).

8.4 Summary

- Twenty-one software estimation models were developed based on 204 paired programs implemented within the U.S.

Department of Defense. The source data provided valuable insight into the costs and schedules associated with the vendor's implementation team in the course of developing and implementing software.

- Methods are simpler and more viable to use for early estimates than traditional parametric cost models.

9. REFERENCES

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

[2] Wallshein C. and Loerch A., July 2015, Software Cost Estimating for CMMI Level 5 Developers, Journal of Systems and Software, Volume 105, pp. 72-78

[3] Alyahya M., Ahmad R., and Lee S., April 2010, Impact of CMMI Based Software Process Maturity on COCOMO II's Effort Estimation, The International Arb Journal of Information Technology, Volume 7, Number 2

[4] Rosa W., Madachy R., Boehm B., Clark B., and Dean, J., June 2014, Improved Method for Predicting Software Cost and Schedule, http://www.iceaaonline.com/ready/wp-content/uploads/2014/07/IT-3-Paper-Improved-Method-for-Predicting-Software-Effort-and-Schedule.pdf

[5] Lanham N., Wallshein C., Rosa W., and Popp M., June 2015, Exploring DoD Software Growth: A Better Way to Model Future Software Uncertainty, http://www.iceaaonline.com/ready/wp-content/uploads/2015/06/SW09-Paper-Lanham-DoD-Software-Growth.pdf

[6] Rosa W., Boehm B., Clark B., Madachy R., Jones C., McGarry J., Lanham N., and Wallshein C., June 2015, Early Phase Software Effort and Schedule Estimation Models, http://www.iceaaonline.com/ready/wp-content/uploads/2015/06/SW12-Paper-Rosa-Early-Phase-Software.pdf

[7] Contract Type, Online Brief by Air Force Materiel Command, Module Lead: OO-ALC/PKCA, August 2007, http://www.acq.osd.mil/dpap

[8] Multiple training materials on contract types at acc.dau.mil

[9] Software Cost Estimation Metrics Manual, http://softwarecost.org/images/2/25/Software_Cost_Estimation_Metrics_Manual_v15f.pdf

[10] Russell, K., January 24, 2005, Quick-Study: CMMI, Computerworld, Volume 39, Number 4, p. 28

[11] Brush, D., November 8, 2014, "Should I throw out these old COBOL books?", Library Hi Tech News, Emerald Group Publishing Limited, 0741-9058, pp. 15-18

[12] C# definition found online at http://searchwindevelopment.techtarget.com/definition/C, site accessed March 24, 2016

[13] C++ definition found online at http://www.cplusplus.com/info/description/, site accessed March 24, 2016

[14] Java definition found online at http://searchsoa.techtarget.com/definition/Java, site accessed March 24, 2016

[15] Government Accountability Office, 2009, "GAO Cost Estimating and Assessment Guide: Best Practices for Developing and Managing Capital Program Costs" http://www.gao.gov/products/GAO-09-3SP, site accessed on March 24, 2016

[16] Software Resource Data Report, 2011, http://dcarc.cape.osd.mil/Files/Policy/2011-SRDRFinal.pdf

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

[18] JMP at http://support.sas.com/documentation/onlinedoc/jmp/index.html, site accessed on March 29, 2016

[19] ACE CO$TAT at https://www.aceit.com/aceit-suite-home/product-info/costat, site accessed on March 29, 2016