Software Effort Estimation Models for Contract Cost Proposal Evaluation

Wilson Rosa and **Corinne Wallshein**
Naval Center for Cost Analysis

ABSTRACT

Context: An effort estimation model at early phase is not very useful if you don't have a logical approach for specifying the size measure and other input variables.

Goal: This study provides a set of practical effort estimation models for software development projects during the contract bidding phase. The first set predicts effort using the entire dataset. The second set predicts effort for agile software development based on data from **20** completed projects using agile methods.

Method: The analysis explores the combined effect of product size, peak staff, and super domain. Product size is measured using the initial software requirements at contract start, and not the actual software requirements reported at contract end. The analysis is based on data from 196 completed projects from the United States Department of Defense delivered from 2005 to 2016.

Result: Statistical results showed that initial software requirements is a valid size metric for estimating both, traditional and agile software development effort. Model's accuracy improves when peak staff and super domain are added as inputs to the equation.

Conclusion: Models may be used for validating contract cost proposals for traditional or agile projects, as the input variables used in the study are often available during the bidding phase or earlier.

***Index Terms*——— cost, effort, estimation, peak staff, requirements volatility, software requirements, domain, early phase, agile, productivity, interfaces**

I. INTRODUCTION

A. Problem Statement

Selecting the appropriate size metric is instrumental in improving the accuracy and credibility of a software cost estimate at an early lifecycle phase [22]. During early inception or elaboration, however, popular size metrics ([8], [13], [35]) can be approximated but not accurately measured [16] as these require additional constructs which are available in considerably late software development phase ([3], [17], [21], [22]). In the United States Department of Defense (DoD), these constructs (e.g. UML diagrams) are captured in source documents provided by software developers after contract award [17]. For this reason, there is a need to find an alternate size measure to estimate effort during the contract bidding phase or earlier ([15], [22], [28], [29]).

B. Purpose Statement

The purpose of this study is to provide open source effort estimation models for traditional and agile software development projects at contract bidding phase. The decision to use software requirements as size measure was based on the following reasons:

1) Initial software requirements can be obtained at early phase from software documents in the DoD [12]. Examples of these include Software Requirements Specification, Requirements Traceability Matrix and System and Software Requirements Document [12].
2) Requirements-based effort estimation models and productivity benchmarks can be derived using historical project data from the Cost Assessment Data Enterprise repository (http://cade.osd.mil) owned by the Office of the Secretary of Defense.

C. Deficiencies in Past Studies

Most recent studies on early phase effort estimation have used either Function Point Analysis ([2], [10], [13], [31]), COSMIC [17], Use Case Points ([6], [19], [23], [24]), or UML artifacts ([1], [14]) as the primary size measure. Although these are widely accepted, deriving them at early phases is challenging as these rely on constructs typically available later in the life cycle ([16[, [22], [23]).

A recent survey study [33] on agile software effort estimation found that the most frequently used size measures by practitioners in descending order are Story Points, Function Points Analysis and Use Case Points. The same respondents said they use these metrics at the sprint or release planning phase of the project. The survey, however, indicated story points, Function Points Analysis, and Use Case Points, are often not used at contract bidding or earlier phase.

A handful of studies ([1], [22], [23], [28], [29]) hypothesized that software requirements may be used to predict software development effort at early phase. However, none of these provided empirical evidence to support it. Whether estimated software requirement is a valid predictor for traditional or agile software effort estimation at early phase remains an open question.

D. Significance of Proposed Study

This study will remedy the prior limitations in several ways:

- Determine whether the sizing approach is valid for predicting development effort:
    - Software requirements was derived by counting the number of "shall" statements contained in the baseline Software Requirements Specification plus number of "shall" statements contained in the baseline Interface Requirements Specifications.

- Use the initial software requirements (submitted at project initiation) as size measure instead of actual (final) software requirements (achieved at project completion). This approach eliminates the need of adjusting the estimated size to account for effort growth.
- Provide a practical framework for decision makers to evaluate contract cost proposals as the model's input variables are often available during the contract bidding phase.

E. Paper Organization

This research paper is organized into eight sections:

- Section I introduces the problem, deficiencies in past studies, and explains the study's proposed solution.
- Section II summarizes the scholarly literature of the variables used in this study for predicting software development effort. It highlights key similarities and differences to this study.
- Section III goes over the research method step by step. It briefly explains the sampling procedure, instrumentation, variables used in the study and the experimental design.
- Section IV describes the data demographics, including operating environment, super domain, and development process.
- Section V discusses the data analysis and descriptive statistics.
- Section VI presents effort estimation models using the entire dataset along with their accuracy and validity tests.
- Section VII presents agile software effort estimation models along with their accuracy and validity tests.
- Section VIII presents the research conclusion on the basis of the hypotheses. It also highlights the contribution and limitations, and outlines areas for future research.
- Section IX cites the sources used in the paper.

II. RELATED WORK

A. Studies relating Requirements to Development Effort

Malik and Boehm [21] introduced a technique for quantifying requirements elaboration for early phase software cost estimation. Their work focused on studying the first stage of requirements elaboration by deriving factors for converting high-level capability goals into low-level capability (functional) requirements using data from 20 real-client graduate-level team projects done at the University of Southern California. The work done by Malik and Boehm is similar to this study in two ways. First, it uses the number of requirements as the primary software size measure at early elaboration phase. Second, it derives software requirements [capability requirements] by counting the number of "shall" statements. Although the authors discussed that prior knowledge of the magnitude of the requirements elaboration is critical for developing early phase software cost estimates, their work did not provide statistical evidence relating software requirements to software development effort.

Malik [22] extended his previous work by examining the entire process of requirements elaboration for early phase software size estimation. The approach involved counting the number of capability goals (CG), capability requirements (CR), use cases (UC), use case steps (UCS), and source lines of code (SLOC) and, thereafter, calculating the elaboration factor for each pair of consecutive requirements levels. The sample was based on multi-level requirements data from 25 small real client e-services projects completed in the past few years by graduate students studying software engineering at University of Southern California.

The author also mapped the Cockburn metaphor to requirement elaboration levels along with source documents that can provide information for each level.

Cockburn metaphor	Requirement elaboration level	Source Document
Cloud	Capability goals	Operational Concept Description
Kite	Capability requirements	System and Software Requirements Document
Sea Level	Use cases	Software Architecture Description
Fish	Use case steps	Software Architecture Description
Clam	SLOC	Source Code

SLOC = source lines of code

According to this mapping, capability requirements are available earlier than use cases, and documented in the system and software requirements document (SSRD). The authors also provided a software process chart showing that initial SSRD may be delivered any time between late project inception and early elaboration phase. Malik's work validates the use of functional requirements for early phase software effort estimation as these are documented earlier than use cases and SLOC. However, his dissertation work did not provide statistical evidence relating number of capability requirements to software development effort.

Sharma and Kushwahab [29] proposed an approach for estimating software development effort using a size measure called requirements based complexity (RBC). RBC is derived using artifacts from the software requirements specification. According to the authors, complexity of the software has a direct bearing on the required amount of effort. The computation of RBC requires 12 input variables:

1) Input Complexity
2) Output Complexity
3) Storage Complexity
4) **Functional Requirements**
5) Non-Functional Requirements
6) Personal Complexity Attributes
7) Design Constraints Imposed
8) Software deployment location
9) **External interfaces**
10) Technical Complexity Factors
11) Size of Language
12) Environmental Complexity

The work by Sharma and Kushwahab is similar to this study in two ways. First, it uses number of functional requirements and external interfaces as size inputs. Second, it considers Software

Requirements Specification as the primary source for size estimation at early phase. Their work, however, has two shortcomings. The experimental design pose a threat to validity as the equation and individual input parameters were not tested for statistical significance. In addition, four out of 12 input parameters are qualitative and not available at early phase -- personal complexity, design constraint, technical complexity, and environmental complexity.

Abrahão and Insfran [1] introduced a measurement procedure (ReqPoints) to estimate the size of object-oriented software projects from requirements specification. The approach consists of extracting artifacts from Use Case Models (classes) and Sequence Diagrams (messages) and thereafter, converting these into unadjusted Function Points using 16 rules. Their approach, however, had three limitations. First, ReqPoints was not validated with actual software development projects. Second, Use Case Models and Sequence Diagrams are included in the system and software architecture document, which typically becomes available during the construction phase. Third, ReqPoint may not be applicable to non-object oriented software projects.

Ochodek, Nawrocki, and Kwarciak [19] investigated the use case point (UCP) measurement in order to find possible ways of simplifying it. The goal was to determine whether the UCP method could be simplified by rejecting the adjustments factors for effort estimation. The analysis was based on 14 projects for which effort ranged from 277 to 3593 man-hours. The authors concluded that using number of use case steps instead of UCP could simplify the effort estimation procedure as the MMRE value for both was virtually the same, and use case steps performed slightly higher for homogenous subsets or when multiple regression was used. The analysis approach by Ochodek and colleagues is similar to this study in one way. The size metric is based on the total sum of a specific requirements elaboration type (use case steps) without applying a complexity weight factor to account for the fact that some requirements can be more complex than others.

Valerdi [35] introduced the "Constructive Systems Engineering Cost Model (COSYSMO)" for estimating systems engineering effort. The analysis was based on data collected from six aerospace companies in the form of expert opinion and historical project data. The effort estimation equation requires four system-level size inputs including number of system requirements, interfaces, algorithms, and operational scenarios. The author added a complexity weight factor to the equation as the simple sum of requirements is not a reliable indicator of functional size. The author concluded that the four size inputs when combined, contributes significantly to the accurate estimation of systems engineering effort.

The work by Valerdi is similar to this study in two ways. First, it uses the total sum of requirements and total sum of interfaces for predicting effort. Second, it uses a similar non-linear effort equation form (without the effort multipliers). Although his work validates the use of number of requirements for predicting effort, it differs from this study in four ways:

- The study did not examine the direct effect of number of requirements on effort.
- In COSYSMO, the number of requirements is further adjusted using a qualitative complexity weight factor.
- In COSYSMO, the number of requirements include different types (i.e., functional, operational, environmental) whereas this study only accounts for functional requirements.
- COSYSMO estimates systems engineering effort whereas this study estimates software development.

Robiolo and Orosco [24] introduced an alternative method for early effort estimation based on Use Case Transactions (UCT) and Entity Objects obtained from four projects developed at the System and Technology Department of Austral University. The result shows that using number of UCT as a notion of size is valid for predicting software development effort. The analytical approach by Robiolo and Orosco is similar to this study in that the size input is based on the total sum of a specific requirements elaboration type without applying a complexity weight factor to account for the fact that some requirements are more complex than others. Their approach, however, had two limitations. The analysis is based on only four projects from a single entity. The size input rely on UML artifacts and diagrams that may not available until after elaboration phase.

B. Studies on Agile Software Effort Estimation

Usman, Mendes, Weidt, and Britto [34] conducted a systematic literature review to provide an overview of the state of the art in the area of effort estimation in agile software development. A total of 20 peer-reviewed papers were examined. The analysis revealed whenever software requirements are given in the form of stories or use case scenarios, Use Case Points and Story Points were the most frequently used size metrics respectively. Very few of those studies used traditional size metrics such as Function Points Analysis and source lines of code. The authors, however, did not address whether Use Case Points and Story Points were measured during or after the contract bidding phase. The work by Usman and colleague differs from this study in two ways. It uses software requirements in the form of stories or use cases as oppose to functional requirements. Second, it converts software requirements into Story Points or Use Case Points instead of directly using requirements as primary size metric.

Usman, Mendes, and Börstler [33] conducted a survey study to report on the state of the practice on effort estimation in agile software development, focusing on a wide range of aspects including estimation techniques and effort predictors used. The study was based on surveys collected from 60 agile practitioners from 16 different countries. The results revealed that 61% of the respondents selected story points as the preferred size metric, 17% selected Function Points Analysis (FPA), and 10% Use Case Points. Only few respondents said they're able to develop an estimate at the bidding or earlier phase. In those cases (7 out of 8), the respondents said they have used expert judgment instead of story points (or other) because of the lack of information. The results of this survey suggest

story points, Function Points Analysis, and Use Case Points, are often not available at contract bidding or earlier life cycle phase.

C. *Studies relating Application Domain to Development Effort*

Rosa, Madachy, Boehm and Clark [26] developed an empirical software effort estimation model for early phase using source lines of code (SLOC) and application domain as predictors. The analysis was based on data from 317 projects implemented within the United States Department of Defense. The dataset was normalized by grouping datset into 12 general complexity zones called application domain. Dummy variables were added to account for the impact of those 12 application domains. The result shows that the effect of SLOC on effort is "highly" significant, when treated along with application domain. The work by the authors is similar to this study in that it uses the same instrumentation, data repository, and examines the effect of size and application domain on effort. However, their work differs from this study in four ways:

- Product size measured in terms of source lines of code.
- It uses the final size (reported at project completion) rather than estimates size (reported at project initiation).
- Did not account for the effect of peak staff and requirements volatility.
- The dataset was only grouped across 12 application domains but did not regroup these into four super domains.

Rosa and colleagues [27] extended their previous work by introducing a simpler domain-driven effort estimation model. As shown in table below, their analysis framework consisted of mapping the dataset (initially reported across different application domains) into four general complexity zones called super domain. Three dummy variables were also added to account for the impact of four super domains.

Super Domain	Application Domain
Support	Software Tools Training
Automated Information Systems	Enterprise Information System Enterprise Services Custom AIS Software Mission Planning
Engineering	Test, Measurement, and Diagnostic Equipment Scientific & Simulation, Process Control, System Software
Real-Time	Communications Real Time Embedded Command & Control Vehicle Control Vehicle Payload Signal Processing Microcode & Firmware

The result shows that the effect of super domains on effort is "highly" significant, when treated along with SLOC. The work by the authors is similar to this study in two ways. First, it uses the same taxonomy for grouping the dataset into four super domains. Second, dummy variables were added to account for the impact of these four super domains. Their work, however, differs from this study in that it uses actual SLOC (reported at project completion) and did not account for the effect of other cost drivers.

D. *Studies relating Peak Staff to Development Effort*

Rodríguez, Sicilia, García, and Harrisonb [32] analyzed the relationship between peak staff and software productivity, along with other project variables. The analysis is based on 199 projects from the International Software Benchmarking Standards Group repository (ISBSG). The authors stated that peak staff is one the most influential factors in software productivity according to the ISBSG organization. The results showed that there is a statistical correlation between peak staff and effort and productivity. The analysis also revealed that enhancement projects show better productivity than new projects. The work by Rodriguez and colleagues is similar to this study in two ways. First, it uses peak staff as a predictor of productivity and effort. Second, it uses a large dataset (199) and considers the impact of platform type which is associated to application domain. Their work differs from this study in that it uses function points as oppose to software requirements.

III. RESEARCH METHOD

A. *Population and Sample*

The sample was identified as 196 projects from across 74 different companies implemented for the United States Department of Defense. This study focused on projects reported at the computer software configuration items (CSCIs).

B. *Questionnaire and Instrumentation*

The primary data collection questionnaire used in the study was from the existing *Software Resource Data Report (SRDR)* ([7, [11], [18], [20], [25]). In our earlier work, this same form was key to deriving DoD cost estimating relationships based on reported lines of code [7].

Each project used in the study contained both, initial and final SRDR forms. The SRDR Initial Developer Report was used to collect the initial functional requirements, estimated external interface requirements, and estimated peak staff reported at project start. The SRDR Final Developer Report was used to collect the actual effort, and requirements volatility reported at project completion. The SRDR questionnaires and forms are available publicly [11], and can be accessed via the links below:
http://cade.osd.mil/Files/Policy/2011-SRDRInitial.pdf
http://cade.osd.mil/Files/Policy/Initial_Developer_Report.xlsx

http://cade.osd.mil/Files/Policy/2011-SRDRFinal.pdf
http://cade.osd.mil/Files/Policy/Final_Developer_Report.xlsx

C. *Data Normalization*

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

1) Counting Software Requirements

The raw dataset reported total initial functional requirements and total initial external interface requirements in separate fields; extracted from the SRDR Initial Developer Report [11]:

> According to SRDR questionnaire respondents (software developers), initial functional requirements were determined by counting the total number of "shall" statements contained in the baseline Software Requirements Specification. Similarly, initial external interface requirements were determined by counting the total "shall" statements contained in the baseline Interface Requirements Specifications.

The "initial Software requirements" was then calculated by summing the total initial functional requirements and the total initial external interface requirements.

2) Data Grouping

The raw dataset was initially reported across different application domains ([7], [11]). The dataset was then stratified into four general complexity zones called super domains [27]. This stratification was adopted from our previous work [27]. The application domains to super domain mapping are shown in Table 1 below.

Table 1

Super Domain Taxonomy

Super Domain	Symbol	Application Domain
Support	SUPP	Software Tools
		Training
Automated Information Systems	AIS	Enterprise Information System
		Enterprise Services
		Custom AIS Software
		Mission Planning
Engineering	ENG	Test, Measurement, and Diagnostic Equipment
		Scientific & Simulation
		Process Control
		System Software
Real-Time	RT	Communications
		Real Time Embedded
		Command & Control
		Vehicle Control
		Vehicle Payload
		Signal Processing
		Microcode & Firmware

D. *Research Questions*

This study will address the following research questions:

Q1: Do initial software requirements relate to final effort?

Q2: Do initial software requirements along with initial peak staff relate to actual development effort?

Q3: Do initial software requirements along with initial peak staff and super domain relate to final effort?

E. *Variables*

The variables examined in the study are identified in Table 2.

Table 2

Variable Names and Definitions

Variable Name	Type	Definition
Final Effort (EFFORT)	Dependent	Actual labor hours as reported in the SRDR Final Developer Report. Captures all the associated engineering effort, by the developer for analyzing, designing, coding, testing, integrating, and managing the software development project.
Initial Software Requirements (REQ)	Independent	The sum of initial functional requirements and initial external interface requirements
Peak Staff (STAFF)	Independent	The initial peak staff measured in terms of full-time equivalents reported in the *Initial SRDR Developer Report.*
Super Domain (SD)	Categorical	Super domain grouping is denoted below: Support = 1, AIS = 2 Engineering= 3, Real-Time = 4

F. *Model Validity Measures*

The model validation measures are described in Table 3.

Table 3

Model Validity Measures

Measure	Symbol	Description
Coefficient of Determination	R^2	The Coefficient of Determination 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 dependent variable. A relative measure allowing direct comparison among models.
Standard Error	SEE	Standard Error of the Estimate is a measure of the difference between the observed and CER estimated effort. The SEE is to linear models as the standard deviation is to a sample mean

Variance Inflation Factor	VIF	Method used to indicate whether multicollinearity is present in a multi-regression analysis. A VIF lower than 10, indicates no multicollinearity.
Mean Magnitude of Relative Error	MMRE	This study uses MMRE as the indicator of model's accuracy. Low MMRE is an indication of high accuracy. MMRE is defined as the sample mean (M) of the magnitude relative error (MME). MME is the absolute value of the difference between actual and estimated effort divided by the actual effort.

IV. DATASET DEMOGRAPHICS

The sample was identified as 196 software projects involving seven operating environments, four super domains, eight different development processes, and 74 different software developers. These projects were completed during the time period from 2005 to 2016. The breakout according to super domain (horizontal axis) and operating environment (vertical axis) are shown in Table 4.

Table 4

Project Characteristics (Entire Dataset)

Operating Environment	Support	Automated Information Systems	Engineering	Real Time
Aircraft	10	2	9	14
C4I	5	9	32	35
Business	1	18	0	0
Ordinance	0	0	0	1
Ship	1	0	0	10
Unmanned Aerial Vehicle	5	1	2	5
Satellite	1	0	6	4
Missile	4	0	3	18
TOTAL	**27**	**30**	**52**	**87**

C4I = command, control, communication, computer, intelligence

Figure 1 below shows the software development processes captured in the dataset. The chart indicates the two most popular software processes are Spiral and Incremental. The dataset also includes 20 agile software development projects.

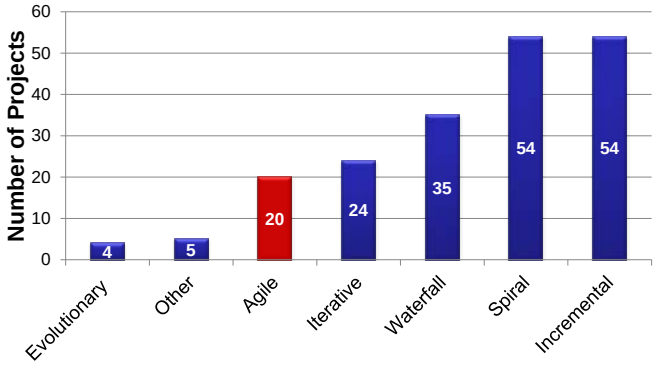

Figure 1: Projects by Software Development Process

V. DESCRIPTIVE STATISTICS

Figure 2 displays the average productivity (actual hours per initial software requirement) by super domain using the entire dataset (n=196). Higher values indicate more effort to complete a software requirement. The results indicate that real-time and engineering projects take more effort to complete than support and automated information systems (AIS). This finding is consistent with previous work ([7], [25], [26], [27]) asserting that software domain is a major productivity driver. This information is useful for crosschecking productivity claims from developers during the contract bidding phase.

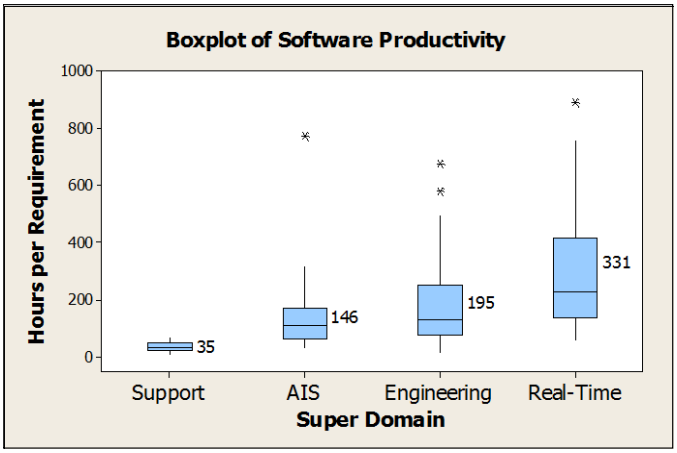

Figure 2 Software Productivity by Super Domain

Figure 3 displays the average productivity (actual hours per initial software requirement) for agile software development projects by super domain. The boxplot shows that agile software development productivity is also influenced by the application domain.

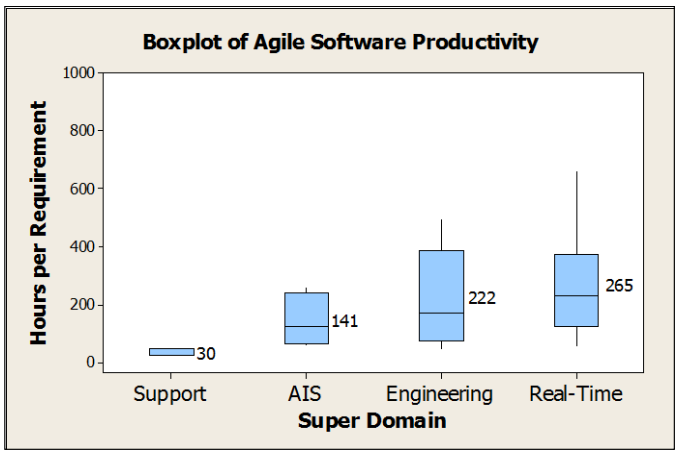

Figure 3 Agile Software Productivity by Super Domain

Table 5 below compares the average productivity (actual hours per initial software requirement) between agile and non-agile software development. The dataset was grouped by size range to control for the effect of fixed effort on smaller projects. Based on this comparison, agile software projects appear to be more productive than non-agile projects.

Table 5

Software Productivity Comparison

Size Range (Requirements)	Hours per Requirement	
	Agile	Non-Agile
1-100	415	466
101-500	159	189
501-5000	77	131
Composite	**190**	**229**

VI. EFFORT MODELS (ENTIRE DATASET)

This section displays three effort models based on the entire dataset (n=196). Also identifies the best-fitting regression model.

A. *Model Equations*

The resulting models are shown below. These effort models may be used for predicting software development effort for either agile or non-agile, defense or business systems, and projects ranging between 8 and 5254 initial software requirements.

Equation Form	Model

$$EFFORT = \mathbf{1379} \times REQ^{\mathbf{0.59}} \quad (1)$$

$$EFFORT = \mathbf{1376} \times REQ^{\mathbf{0.3662}} \times STAFF^{\mathbf{0.5225}} \quad (2)$$

$$EFFORT = \mathbf{244} \times REQ^{\mathbf{0.481}} \times STAFF^{\mathbf{0.488}} \times SD^{\mathbf{1.15}} \quad (3)$$

Where:

EFFORT = Final engineering Labor in hours

REQ = Initial software requirements

STAFF = Estimated peak staff in full-time equivalent

SD = Super Domain (1 if mission support, 2 if AIS, 3 if engineering, 4 if real-time)

B. *Model Validity*

Table 6 below shows the model validity results. All three variables examined showed a significant effect on software development effort as their t-statistics exceed the two-tailed critical values, given the coefficient alpha (0.05). In addition, all equation forms are appropriate as the VIF values indicate no multicollinearity present in the analysis.

Table 6

Model Validity (Entire Dataset)

Model	t-statistics				VIF		
	Inter	REQ	Staff	SD	REQ	Staff	SD
(1)	27.2	12.8	***	***	***	***	***
(2)	33.2	8.3	9.8	***	1.36	1.36	***
(3)	29.3	15.6	13.5	15.0	1.45	1.37	1.07

Inter = intercept

C. *Model Accuracy*

Table 7 below compares the accuracy of the models. The accuracy dramatically improved when peak staff and super domain were added to the initial model. Model (3) is the best fitting model as it has the lowest MMRE and highest R2.

Table 7

Model Accuracy (Entire Dataset)

Model	Variables	R2 (%)	MMRE (%)	CV (%)	SEE	Mean
(1)	1	45	101	55	87958	74425
(2)	2	63	71	48	64799	74425
(3)	3	83	41	32	41099	74425

VII. AGILE SOFTWARE EFFORT MODELS

This section displays three effort models derived using the agile software project subset (n=20). It also identifies the best-fitting regression model that can be constructed.

A. *Agile Model Equations*

The resulting models are shown below. These effort models are useful for predicting agile software development effort. Also appropriate for defense or business systems, and projects ranging between 10 and 4867 initial software requirements.

Equation Form	Model

$$EFFORT = \mathbf{2202} \times REQ^{\mathbf{0.5009}} \quad (4)$$

$$EFFORT = \mathbf{1045} \times REQ^{\mathbf{0.4071}} \times STAFF^{\mathbf{0.4404}} \quad (5)$$

$$EFFORT = \mathbf{200} \times REQ^{\mathbf{0.512}} \times STAFF^{\mathbf{0.478}} \times SD^{\mathbf{1.001}} \quad (6)$$

Where:

EFFORT	=	Engineering Labor in hours
REQ	=	Initial software requirements
STAFF	=	Estimated peak staff in full-time equivalent
SD	=	Super Domain (1 if mission support, 2 if AIS, 3 if engineering, 4 if real-time)

B. Agile Model Validity

Table 8 below shows the model validity results. All three variables examined have a significant effect on software development effort as their t-statistics exceed the two-tailed critical values, given the coefficient alpha (0.05). All three models are appropriate as their VIF indicate no multicollinearity present in the analysis.

Table 8

Model Validity (Agile Software Subset)

	t-statistics				VIF		
Model	Inter.	REQ	Staff	SD	REQ	Staff	SD
(4)	12.3	4.7	***	***	***	***	***
(5)	10.2	3.7	2.0	***	1.2	1.2	***
(6)	9.1	6.7	3.2	4.6	1.4	1.3	1.0

C. Agile Model Accuracy

Table 9 below compares the accuracy of the agile effort models. The accuracy dramatically improved when peak staff and super domain were added to the initial model. Model (6) is the best fitting model as it displays the lowest MMRE and highest R2 compared to the other two.

Table 9

Model Accuracy Results

Model	Variables	R2 (%)	MMRE (%)	CV (%)	SEE	Mean
(4)	1	53	64	48	51892	62140
(5)	2	63	71	47	40328	62140
(6)	3	81	32	22	19295	62140

VIII. CONCLUSION

A. Primary Findings

This study introduced effort estimation models for defense software development projects at the early elaboration phase using data from 196 completed projects reported at the CSCI level.

The regression analysis indicates "initial software requirements" is a valid size measure for predicting both, agile and non-agile software development effort at early phase.

An effort estimating model only based on software requirements is statistically significant but not very accurate. The model accuracy improves after peak staff and super domain, are gradually added to the initial model.

This study also revealed that meaningful productivity comparisons (hours per software requirements) can be made when projects are grouped by super domain. This finding is consistent with the perception that support and automated information system applications requires less effort to implement than real-time applications.

B. Threats to Validity

Although this study mitigated sampling bias, it still has five limitations:

1) This study only examined the impact of software requirements and peak staff, domain on development effort. A future investigation should analyze the impact of other cost drivers such as percent requirements reuse, process maturity, and personnel experience.
2) The study did not apply a size complexity weight factor to the initial software requirements to account for the fact that some requirements can be more complex than others. This can be achieved by asking each organization to apply discrete weights (easy, nominal, and difficult) to the estimated requirements similar to the one used in COSYSMO.
3) A non-random sample was preferred as the researcher had access to names in the population and the selection process for participants was based on their convenience and availability. However, this process limits the ability to generalize to a population.
4) Data was only collected from software projects within the United States Department of Defense. In principle, the analysis framework may apply to commercial sector systems, but this study did not have the data to test this hypothesis.
5) Although the models in this study are not highly precise, they have the advantage of providing information on its relative accuracy.

C. Model Usefulness

These models are collectively useful in different scenarios:

- Models (1) through (3) useful for predicting traditional or agile software development effort during the bidding phase.
- Models (4) through (6) useful for predicting agile software development effort during the bidding phase.

D. Model Limitations

Since the data was collected at the CSCI level, the resulting models are not appropriate for projects reported at the aggregate level. They should not be used if estimated size inputs are outside of the regression models' dataset range.

E. Future Work

There are important areas of future work to improve the usefulness of these model types for practitioners. The software granularity of modeling can be finer. We intend to develop

similar regression models for agile projects using a dataset greater than 20. We will also examine the impact of software requirements on software development effort while controlling for the effects of development process, process maturity, and percent reuse.

IX. REFERENCES

[1] S. Abrahao and E. Insfran, "A Metamodeling Approach to Estimate Software Size from Requirements Specifications," *Software Engineering and Advanced Applications, 2008. SEAA '08. 34th Euromicro Conference*, vol., no., pp.465-475, 3-5 Sept. 2008.

[2] N. Adem and M. Kasirun, "Automating Function Points analysis based on functional and non functional requirements text," in *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on*, vol.5, no., pp. 664-669, 26-28 Feb. 2010.

[3] A. Anton, "Goal-based requirements analysis," in *Requirements Engineering, 1996., Proceedings of the Second International Conference on* , vol., no., pp.136-144, 15-18 Apr 1996.

[4] B. Boehm, "Software Engineering Economics," Englewood Cliffs, NJ, Prentice-Hall, 1981.

[5] B. Boehm, C. Abts, W. Brown, S. Chulani, B. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece, "Software Cost Estimation with COCOMO II," Prentice-Hall, 2000.

[6] M. Braz and S. Vergilio, "Software Effort Estimation Based on Use Cases," Computer Software and Applications Conference, COMPSAC '06. 30th Annual International, Dept. of Comput. Sci., Fed. Univ. of Parana, Curitiba, 2006, pp. 221-228.

[7] B. Clark and R. Madachy (Eds.). (2015, Apr.). Software Cost Estimation Metrics Manual for Defense Systems. Software Metrics Inc., Haymarket, VA. [Online]. Available: http://www.sercuarc.org/wp-content/uploads/2014/05/Software-Cost-Estimation-Metrics-Manual-for-Defense-Systems.pdf

[8] Common Software Measurement International Consortium. (2009, May). *The COSMIC Functional Size Measurement Method*, Version 3.0.1., Measurement Manual. [Online]. Available: http://www.cosmicon.com/portal/public/COSMIC%20Method%20v3.0.1%20Measurement%20Manual.pdf

[9] S. Ferreira, J. Collofello, D. Shunk, G. Mackulak. (2009, Oct.). Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation. Journal of Systems and Software, Volume 82, Issue 10, pp. 1568-1577, ISSN 0164-1212. Available: http://www.sciencedirect.com/science/article/pii/S0164121209000557

[10] J. Cuadrado-Gallego, P. Rodriguez-Soria, A. Gonzalez, D. Castelo, and S. Hakimuddin, "Early Functional Size Estimation with IFPUG Unit Modified," Computer and Information Science (ICIS), 2010 IEEE/ACIS 9th International Conference on, Comput. Sci. Dept., Univ. of Alcala, Alcala de Henares, Spain, 2010, pp. 729-733.

[11] Department of Defense (2011, Nov.). Software Resource Data Report. [Online]. Available: http://dcarc.cape.osd.mil/Files/Policy/2011-SRDRFinal.pdf

[12] Department of Navy. (2010, Sep.). Software Criteria and Guidance for Systems Engineering Technical Reviews (SETR) Supplement to Guidebook for Acquisition of Naval Software Intensive Systems. [Online]. Available: http://www.secnav.navy.mil/rda/OneSource/Documents/Program%20Assistance%20and%20Tools/Handbooks,%20Guides%20and%20Reports/Page%205/supplementtoguidebook.pdf

[13] S. Furey. (1997, Apr.). Why We Should Use Function Points [software metrics]. *Software, IEEE*, 14(2), pp. 28–30. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=582971&isnumber=12658

[14] M. Heričko, and A. Živkovič. (2008, Jun.). The size and effort estimates in iterative development, Information and Software Technology. Volume 50, Issues 7–8, pp. 772-781, ISSN 0950-5849. Available: http://www.sciencedirect.com/science/article/pii/S0950584907000870

[15] International Function Point Users Group. (2000, Apr.). Function Point Counting Practices Manual, Release 4.1.1. [Online]. Available: http://perun.pmf.uns.ac.rs/old/repository/research/se/functionpoints.pdf

[16] C. Jones, "Function points as a universal software metric," *SIGSOFT Softw. Eng. Notes*38, 4 (July 2013), 2013, pp. 1-27.

[17] M, Kaya, and O. Demirors, "E-Cosmic: A Business Process Model Based Functional Size Estimation Approach," Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on, Inf. Inst., Middle East Tech. Univ., Ankara, Turkey, 2011, pp. 404-410.

[18] I. Lipkin, "Test Software Development Project Productivity Model", Ph.D. dissertation, Univ. of Toledo, Toledo, OH, 2011.

[19] M. Ochodek, J. Nawrocki, and K. Kwarciak. (2011, Mar.). Simplifying effort estimation based on Use Case Points. Information and Software Technology, Volume 53, Issue 3, pp. 200-213, ISSN 0950-5849. Available: http://www.sciencedirect.com/science/article/pii/S095058491000176X

[20] R. Madachy, B. Boehm, B. Clark, T. Tan, and W. Rosa, "US DoD Application Domain Empirical Software Cost Analysis," *2011 International Symposium on Empirical Software Engineering and Measurement*, 2011, pp. 392 – 395.

[21] A. Malik, and B. Boehm. (2011, Jul.). Quantifying requirements elaboration to improve early software cost estimation. Information Sciences, Volume 181, Issue 13, 1 July 2011, pp. 2747-2760, ISSN 0020-0255. Available: http://www.sciencedirect.com/science/article/pii/S002002550900526X

[22] A. Malik, "Quantitative and Qualitative Analyses of Requirements Elaboration for Early Software Size Estimation", PhD Dissertation, Computer Science Department, University of Southern California, 2011.

[23] A. Nassif, D. Ho, and L. Capretz. (2013, Jan.). Towards an early software estimation using log-linear regression and a multilayer perceptron model. Journal of Systems and Software, Volume 86, Issue 1, pp. 144-160, ISSN 0164-1212. Available: http://www.sciencedirect.com/science/article/pii/S0164121212002221

[24] G. Robiolo, R. and R. Orosco, "An Alternative Method Employing Uses Cases for Early Effort Estimation," *Software Engineering Workshop, 2007. SEW 2007. 31st IEEE* , vol., no., pp.89-98, March 2007-Feb. 8 2007

[25] W. Rosa, T. Packard, A. Krupanand, J. Bilbro, and M. Hodal. (2013, Feb.). COTS integration and estimation for ERP. Journal of Systems and Software, Volume 86, Issue 2, February 2013, pp. 538-550, ISSN 0164-1212. Available: http://www.sciencedirect.com/science/article/pii/S0164121212002713

[26] W. Rosa, R. Madachy, B. Boehm, B. Clark, "Simple Empirical Software Effort Estimation Model," *ESEM '14 Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, Article No. 43.

[27] W. Rosa, B. Clark, B. Boehm, and R. Madachy. (2014, Oct.) Simple-Empirical Software Cost Estimation. 29th International Forum on COCOMO and Systems/Software Cost Modeling. [Online]. Available: http://csse.usc.edu/new/wp-content/uploads/2014/10/Simple-Empirical-Software-Cost-Estimation_v2.pdf

[28] A. Sharma, and D. Kushwaha, "Early estimation of software complexity using requirement engineering document," *SIGSOFT Softw. Eng. Notes* 35, 5 October 2010.

[29] A. Sharma, and D. Kushwaha, "Estimation of Software Development Effort from Requirements Based Complexity," Procedia Technology, Volume 4, 2012, pp. 716-722, ISSN 2212-0173.

[30] TECOLOTE Inc. (2014, Jul.). Automated Cost Estimating Integrated Tools: CO$TAT. [Online]. Available: http://ww.aceit.com/Pages/Products/ProductPage.aspx?id=f638a6d8-60e9-414a-9970-7fed249b9d25

[31] M. Tsunoda, Y. Kamei, K. Toda, M. Nagappan, K. Fushida, and N. Ubayashi, "Revisiting software development effort estimation based on early phase development activities," Mining Software Repositories

(MSR), 2013 10th IEEE Working Conference on, Toyo Univ., Saitama, Japan, 2013, pp. 429-438.

[32] D. Rodríguez, M.A. Sicilia, E. García, R. Harrison. (2012, Mar.). Empirical findings on team size and productivity in software development. Journal of Systems and Software," Volume 85, Issue 3, March 2012, pp. 562-570, ISSN 0164-1212. Available: http://www.sciencedirect.com/science/article/pii/S0164121211002366

[33] M. Usman, E. Mendes, F. Weidt, and R. Britto, "Effort estimation in agile software development: a systematic literature review," In *Proceedings of the 10th International Conference on Predictive Models in Software Engineering* (PROMISE '14). ACM, New York, NY, USA, 2014, pp. 82-91.

[34] M. Usman, E. Mendes, and J. Börstler, "Effort estimation in agile software development: a survey on the state of the practice," In *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering* (EASE '15). ACM, New York, NY, USA, 2015, Article 12, 10 pages.

[35] R. Valerdi, *"The constructive systems engineering cost model (COSYSMO)", Ph.D. dissertation, Dept. Industrial and System Eng., Univ. of Southern California*, Los Angeles, CA, 2005.