Early Phase Cost Models for Agile Software Processes in the US DoD

Wilson Rosa
IT Estimating Division
Naval Center for Cost Analysis
Washington D.C, USA
wilson.rosa@navy.mil

Raymond Madachy
Department of Systems Engineering
Naval Postgraduate School
Monterey, CA, USA
rjmadach@nps.edu

Corinne Wallshein
IT Estimating Division
Naval Center for Cost Analysis
Washington D.C, USA
corinne.wallshein@navy.mil

Bradford Clark
Software Metrics, Inc.
Haymarket, VA, USA
brad@software-metrics.com

Barry Boehm
USC Center for Systems and Software Engineering
University of Southern California
Los Angeles, CA, USA
boehm@usc.edu

***Abstract*—*Background*: Software effort estimates are necessary and critical at an early phase for decision makers to establish initial budgets, and in a government context to select the most competitive bidder for a contract. The challenge is that estimated software requirements is the only size information available at this stage, compounded with the newly increasing adoption of agile processes in the US DoD.**
***Aims*: The objectives are to improve cost estimation by investigating available sizing measures, and providing practical effort estimation models for agile software development projects during the contract bidding phase or earlier.**
***Method*: The analysis explores the effects of independent variables for product size, peak staff, and domain on effort. The empirical data for model calibration is from 20 industrial projects completed recently for the US DoD, among a larger dataset of recent projects using other lifecycle processes.**
***Results*: Statistical results showed that initial software requirements is a valid size metric for estimating agile software development effort. Prediction accuracy improves when peak staff and domain are added as inputs to the cost models.**
***Conclusion*: These models may be used for estimates of agile projects, and evaluating software development contract cost proposals with inputs available during the bidding phase or earlier.**

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

I. Introduction

In the United States Department of Defense (US DoD), it is necessary and most critical to estimate software development cost in early lifecycle phases when limited data is available. These initial estimates are used to evaluate proposals for government source selection, and to establish initial program budgets. More realistic estimates in the beginning will minimize project cost overruns.

The problem is compounded because agile software processes are increasingly used in the DoD, and acquisition practices must keep pace with the changes. Agile development processes (e.g. Scrum, Extreme Programming) have been more prevalent in industry, and are now being adopted more across the DoD and other government agencies. Thus there is a dire need for new, accurate, credible cost models calibrated to actual project data. This study uses empirical data from completed industry projects extracted from a DoD database. The majority of agile projects were submitted in the last two years.

The results are significant because the data provides initial empirical-based insight into DoD agile projects. It also introduces the first cost model calibration to completed agile software projects in the database, and a first comparison of productivity with traditional processes.

The subset of agile projects is the primary focus of this work. Results of the larger study covering all projects and process types are not addressed in this paper, except for a summary comparison of productivity for agile vs. non-agile projects.

An important distinction of this approach is using early phase initial estimates for model inputs and historical calibration of the cost model. This is pragmatic since those initial estimated inputs are the only information available for the early phase budgeting. The contrast between our early phase model and traditional post-calibrated models such as COCOMO II [5] are illustrated in Table I.

TABLE I
Comparison of Model Calibrations

	Size		**Cost Factors**		**Effort**
Model Type	Initial Estimate	Final Actual	Initial Estimates	Final Actuals	Final Actual
Early Phase	x		x		x
Traditional		x		x	x

II. Background

A. DoD Empirical Data and Agile Processes

The source of actual industrial data for the DoD is the Cost Assessment Data Enterprise (CADE) repository

(http://cade.osd.mil) owned by the Office of the Secretary of Defense for Cost Assessment and Program Evaluation (OSD CAPE). The quantitative project data is contained in Software Resources Data Report (SRDR) records. The DoD acquisition process outlined in the DoD Instruction 5000.02 policy Operation of the Defense Acquisition System mandates the SRDR as a regulatory contract reporting requirement [11].

The SRDR is used to obtain both the estimated and actual characteristics of new software developments or upgrades. Both the Government program office and later the software contractor submit the SRDR. It constitutes a contract data deliverable for contractors that formalizes the reporting of software metric and resource data.

Previously, early phase cost estimation relationships for the DoD were developed from SRDR data in the CADE repository using size in source lines of code and military application domain as predictors [7]. However, none of the data was for agile projects.

In 2009, a new Defense Authorization Act required DoD to implement a new acquisition process for IT systems [36]. This new process included principles of Agile development such as early and continual involvement of the user, multiple rapidly executed increments or releases of capability, early successive prototyping to support an evolutionary approach, and a modular open-systems approach.

The current SRDR forms include identification of the software development process used with Agile being an option (see Section IV-B for more details). Supplemental details of the processes can be provided in an associated data dictionary. Agile process definitions for the DoD and implementation guidelines are provided in [36] and [37].

B. Size Metrics

Selecting the appropriate size metric is instrumental in improving the accuracy of a software cost estimate at an early lifecycle phase [22]. During early inception and elaboration, however, popular size metrics such as the COSMIC method [8] or Function Points [13] can be approximated but not accurately measured [16]. In later software development phases these can be measured with additional artifacts such as Use Case Diagrams, Class Diagrams, or other UML diagrams ([3], [17], [21], [22]).

Most recent studies on early phase effort estimation have used either Function Point Analysis ([2], [10], [13]), COSMIC [17], Use Case Points ([6], [19], [23], [24]), or UML artifacts ([1], [14]) as the primary size measure.

A recent survey study [34] on agile software effort estimation showed projects frequently use Story Points, Function Points, and Use Case Points for software size (see Section III). Although all 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]).

In the DoD these constructs are captured in artifacts 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]).

This study will overcome these limitations by providing effort estimation models for agile software development projects at contract bidding phase. The decision to use software requirements as a size measure was based on the fact that these can be obtained in early phases from software documents in the DoD [12]. Examples of these source documents include Software Requirements Specification, Requirements Traceability Matrix and System and Software Requirements Document [12].

III. Related Work

A. Agile Effort Estimation

A systematic literature review was conducted to provide an overview of the state of the art in the area of effort estimation in agile software development [33]. A total of 20 peer-reviewed papers were examined. A summary follows:

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 in [33] differs from this study in two ways. It uses software requirements in the form of stories or use cases as opposed to functional requirements. Second, it converts software requirements into Story Points or Use Case Points instead of directly using requirements as primary size metric.

A further survey study was conducted to report on the state of the practice on effort estimation in agile software development [34], 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, and 10% Use Case Points. Only a few respondents said they were able to develop an estimate at the bidding or an 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.

B. Relating Requirements to Effort Estimation

One approach for estimating software development effort uses a size measure called requirements-based complexity (RBC) [29]. 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) Personnel 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 in [29] is similar to this study in two ways. First, it uses the 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 poses a threat to the 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 – personnel complexity, design constraints, technical complexity, and environmental complexity.

A measurement procedure called ReqPoints estimates the size of object-oriented software projects from requirements specification [1]. 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. The approach, however, has 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.

An alternative method for early effort estimation is based on Use Case Transactions (UCT) and Entity Objects obtained from four projects developed at the System and Technology Department of Austral University [24]. The result shows that using the number of UCT as a notion of size is valid for predicting software development effort. The analytical approach 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. The approach, however, has two limitations. The analysis is based on only four projects from a single entity. The size input relies on UML artifacts and diagrams that may not be available until after elaboration phase.

C. Relating Application Domain to Effort Estimation

In our earlier work analyzing DoD projects [26], we 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 CADE data from 317 projects implemented within the DoD. The dataset was normalized by grouping data into 12 general complexity zones called application domains. Categorical variables were added to account for the impact of the domains. The result shows that the effect of SLOC on effort is highly significant, when treated along with application domain. The work 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, it 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.

We later extended this work by introducing a simpler domain-driven effort estimation model [27]. The analysis framework consisted of mapping the dataset (initially reported across different application domains) into four general complexity zones called super domains.

The results showed that the effect of super domains on effort is highly significant, when treated along with SLOC. The work is similar to this study by using the same taxonomy for grouping the dataset into four super domains and accounting for their impact. It differs in that it uses actual SLOC (reported at project completion) and did not account for the effect of other cost drivers.

IV. Research Method

A. Population and Sample

The method uses empirical analysis of recent DoD agile software development project data from submitted SRDRs to derive early phase cost estimation relationships (CERs) with metrics available on agile projects. Metrics on initial SRDRs for model calibration include the number of requirements (at source selection), peak staff (at proposal), requirements volatility and application domain.

The sample subset was 20 agile projects from across 14 industrial contractors for the DoD from 2008 to 2016. They are part of a larger dataset of 196 recent projects using other lifecycle processes. This study focused on agile projects reported at the Computer Software Configuration Item (CSCI) level.

B. Questionnaire and Instrumentation

The primary data collection form used in the study is the existing SRDR [11]. In our earlier work, this same form was key to deriving DoD CERs based on reported lines of code [7], [20], [26].

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

The SRDR questionnaire [11] requires developers to enter the name of the development process (e.g. Waterfall, Spiral, Incremental, and Agile). If the developer enters Agile, they must indicate whether it is part of an Agile acquisition approach, and whether it would be considered Hybrid Agile (e.g. Waterfall for Architecture and Requirements, followed by Agile for design, code, and unit test).

C. Data Normalization

The dataset was normalized for size, effort, and segmented into domains and agile categories per the steps below.

1) Converting to Person-Months: The raw dataset reported actual effort in labor hours. The reported labor hours were then converted into Person-Months using the factor 152 hours per Person-Month from COCOMO II ([4], [5]), which is commonly used in the DoD for estimation and planning.
2) 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.
3) Grouping Dataset by Super Domain: 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 I.
4) Grouping Dataset by Agile Process: The raw dataset was initially reported as either Agile, Waterfall/Agile, or Modified Agile. The dataset was then stratified into two main groups – Agile and Hybrid Agile. The Hybrid Agile group included Waterfall/Agile and Modified Agile.
5) Grouping Dataset by Agile Framework. The raw dataset and associated dictionary provided limited information about the Agile framework, as it is not a required field in the SRDR questionnaire [11]. Based on available information, projects were reported as Scrum, Sprints, Lean Software Development, or Iterative Development. Projects reported as either Sprints or Scrum were grouped together in a category called Scrum, as Scrum iterations are actually Sprints.

TABLE II
SUPER DOMAIN TAXONOMY

Super Domain	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, & 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

The goal of this research is to improve cost estimation of DoD agile software development projects early in the lifecycle. This leads to the following related research questions for measurement and analysis:

- What size measures are available in early lifecycle phases for DoD programs using agile processes?
- What other cost factors are available in early lifecycle phases for DoD agile projects?
- How well can they predict effort for agile processes?

We found the measures of requirements, peak staff, requirements volatility and application domain are recorded in initial submissions. With these, the more specific questions can be answered in this study:

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

effort?

Larger research questions that this study feeds into are:

- For overall software process understanding, how do agile and traditional processes compare for productivity, cost, schedule and quality performance in the DoD?

Looking forward for improved measurement vehicles in the DoD, the following question bears on regulatory governance for data collection:

- What size measures and other cost factors should the government mandate from contractors to better support program estimates for agile projects and others?

E. Variables

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

TABLE III
VARIABLE NAMES AND DEFINITIONS

Variable Name	Type	Definition
Person-Months (PM)	Dependent	Actual labor in Person-Months. 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 as: Support = 1, AIS = 2, Engineering = 3, Real-Time = 4

F. Model Validity Measures

The model validation measures are described in Table IV.

TABLE IV
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.
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.
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
Mean Magnitude of Relative Error	MMRE	This study uses MMRE as the indicator of models accuracy. Low MMRE is an indication of high accuracy. MMRE is defined as the sample mean (M) of the magnitude of relative error (MME). MME is the absolute value of the difference between actual and estimated effort divided by the actual effort.

G. Demographics and Descriptive Statistics

The sample contains 20 software projects involving four operating environments, four super domains, and 14 different software developers. The breakout according to super domain (horizontal axis) and operating environment (vertical axis) are shown in Table V.

TABLE V
PROJECT TYPE COUNTS

Operating Environment	Super Domain			
	Support	AIS	Engineering	Real Time
Aircraft	2	0	4	0
Business	1	3	0	0
C4I	0	1	3	5
Missile	0	0	0	1

C4I = command, control, communication, computer, intelligence
AIS = automated information systems

Fig. 1 groups the dataset by Agile framework. The majority of the projects used the standard Scrum framework [36] delivering software in iterative sprints. The other four reported using Lean Software Development adaptations [38], other iterative agile frameworks or unspecified.

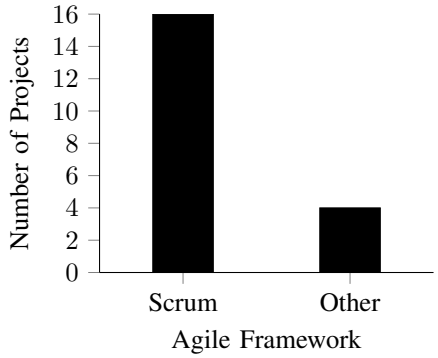

Fig. 1. Projects by Agile Framework Type

Fig. 2 groups the dataset by actual effort (in Person-Months). The majority of the projects (15) expended less than 500 person-months in software development activities. Only two projects expended more than 1000 person-months.

Figure 3 groups the dataset according to the initial software requirements reported by the developers. The chart indicates a uniform spread across project sizes.

Fig. 4 displays the average effort per requirement (actual hours per initial software requirement) for agile software development projects by super domain. The boxp and whisker plot shows agile software productivity is influenced by the application domain. The boxes are bounded by the 1st and 3rd quartiles, Q1 and Q3, with the median shown as a line in the box. The tails represent values beyond the 1st and 3rd quartiles. The lower whisker on each is the smallest data value which is larger than the lower quartile. The upper whisker is the largest data value which is smaller than the upper quartile. Data outliers beyond these ranges are indicated with marks.

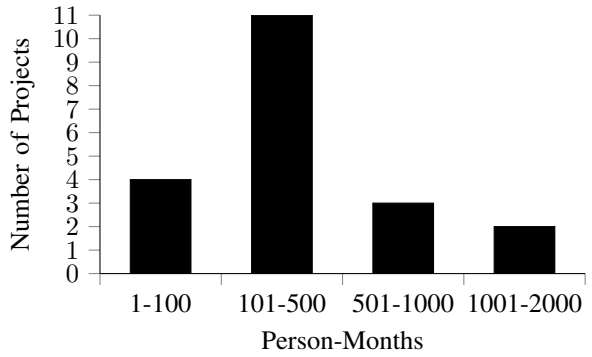

Fig. 2. Projects by Actual Effort Range

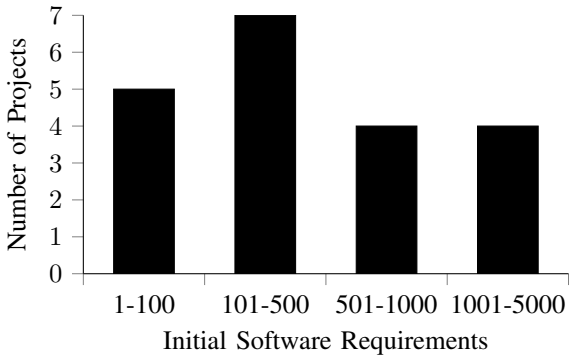

Fig. 3. Projects by Size Range

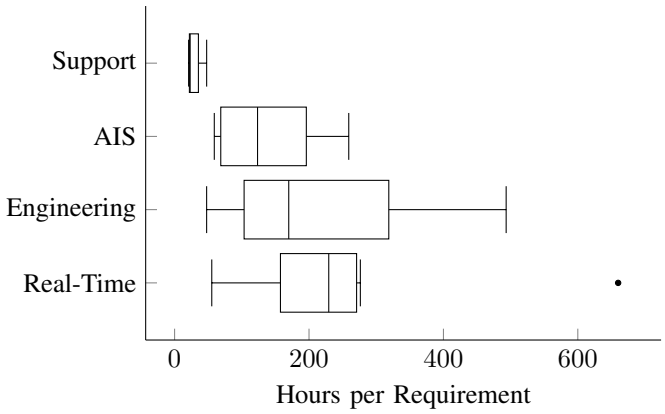

Fig. 4. Super Domain Productivities Boxplot

V. Agile Effort Estimation Models

This section displays three effort models derived using the agile software project subset (n=20) and commercial statistical package [30]. It also identifies the best-fitting regression model that can be constructed.

A. Effort Model Equations

The resulting effort models with increasing independent variables are in Equations (1), (2), and (3). These models predict agile software development effort, based on the defense and business systems projects ranging between 10 and 4900 initial software requirements in the dataset.

Product size is measured using the initial software requirements at contract start, and not the actual software requirements reported at contract end.

$$PM = 14.5 * \mathit{REQ}^{0.5009} \tag{1}$$

$$PM = 6.8 * \mathit{REQ}^{0.4071} * \mathit{STAFF}^{0.44404} \tag{2}$$

$$PM = 1.3 * \mathit{REQ}^{0.512} * \mathit{STAFF}^{0.478} * \mathit{SD}^{1.001} \tag{3}$$

where

PM =	Engineering Labor in Person-Months
REQ =	Initial software requirements at the bid phase
STAFF =	Estimated peak staff in full-time equivalent at the bid phase
SD =	Super Domain (1 if mission support, 2 if AIS, 3 if engineering, 4 if real-time)

B. Model Validity

Table VI shows the model validity results. All three variables examined have a significant effect on software development effort as indicated by their p-values. All three models are appropriate as their variance inflation factor (VIF) indicate no multicollinearity present in the analysis.

TABLE VI
Model Validity Measures

	p-value				VIF		
Model	**Inter.**	**REQ**	**Staff**	**SD**	**REQ**	**Staff**	**SD**
(1)	0.0000	0.0002	***	***	***	***	***
(2)	0.0000	0.0015	0.0559	***	1.2	1.2	***
(3)	0.0000	0.0000	0.0045	0.0003	1.4	1.3	1.0

C. Agile Model Accuracy

Table VII compares the accuracy of the agile effort models. The accuracy dramatically improved when peak staff and super domain were added to the initial model. The model in (3) is the best fitting model as it displays the lowest MMRE and highest R^2 compared to the other two. It also exhibits a smaller SEE for software cost models in general.

TABLE VII
Model Accuracy Results

Model	**# Variables**	R^2	**MMRE**	**CV**	**SEE**
(1)	1	53	64%	48%	341
(2)	2	63	52%	36%	265
(3)	3	81	32%	22%	127

D. Agile Vs. Traditional Processes

The other portion of the project data for traditional processes was handled similarly. Though not the focus of this effort, the two groups do allow for an initial comparison between the two. The caveat is the relatively small sample of 20 for agile vs. 176 for traditional (non-agile).

The comparison in Table VIII shows the average productivities between agile and non-agile software development, measured as the initial software requirements per actual Person-Month. 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 VIII
AVERAGE SOFTWARE PRODUCTIVITY COMPARISONS

Size Range (# Requirements)	Agile (Rqts./PM)	Non-Agile (Rqts./PM)
1-100	0.37	0.33
101-500	0.96	0.80
501-5000	1.97	1.16
Composite	0.80	0.66

VI. CONCLUSIONS

A. Primary Findings

The regression analysis indicates initial software requirements is a valid size measure for predicting 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 incrementally added to the model.

The models may be used for validating contract cost proposals for agile software projects for independent government cost estimates, as the input variables used in the study are often available during the bidding phase.

Since the data was collected at the CSCI level, the resulting models may not be appropriate for projects reported at the aggregate level due to the excluded cost of subsystem integration. The models have also not been validated outside of the regression models dataset size range.

A related result of interest from this first batch of CADE data containing agile projects is a productivity comparison. The dataset indicates that agile software projects appear to be more productive than non-agile projects, and this will be updated in the future with larger samples.

B. Threats to Validity

This study only examined the impact of software requirements, peak staff, and super domain on development effort. A future investigation should analyze the impact of other cost drivers such as percent requirements reuse, volatility, process maturity, and personnel experience.

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. A future study will mitigate this shortcoming by asking each organization to apply discrete weights (easy, nominal, and difficult) to the estimated requirements similar to the one used in the Constructive Systems Engineering Model [35].

In principle, the analysis framework may apply to commercial sector systems, but this study did not have the data to test this hypothesis.

A larger dataset beyond 20 projects is preferable to increase model validity and accuracy, and this data collection is ongoing.

C. 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.

Given the process trends, DoD acquisition practices must keep pace with the changing processes. Our research will support this on technical and regulatory levels. Results will be transitioned at DoD venues. New CERs for agile processes across the domains will go in the next edition of the Software Cost Estimation Metrics Manual for Defense Systems. Recommended improvements to the SRDR to cover agile processes and size measures will also be submitted to the OSD CAPE for incorporation.

ACKNOWLEDGMENT

Thanks to Joseph M Diaz (Naval Center for Cost Analysis) and to the many interviewed data submitters for their outstanding support of this effort.

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, PrenticeHall, 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, PrenticeHall, 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

[13] S. Furey. (1997, Apr.). Why We Should Use Function Points [software metrics]. Software, IEEE, 14(2), pp. 2830. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=582971&isnumber=12658

[14] M. Heriko, and A. ivkovi. (2008, Jun.). The size and effort estimates in iterative development, Information and Software Technology. Volume 50, Issues 78, 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. Notes38, 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. Rodrguez, M.A. Sicilia, E. Garca, 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. Brstler, 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.

[36] United States Department of Defense (2016, Mar.). Agile and Earned Value management: A Program Managers Desk Guide. [Online]. Available http://www.acq.osd.mil/evm/docs/PARCA

[37] United States Government Accountability Office (2012, Jul.). SOFTWARE DEVELOPMENT: Effective Practices and Federal Challenges in Applying Agile Methods. [Online]. Available http://www.gao.gov/assets/600/593091.pdf

[38] M. Poppendieck, and T. Poppendieck (2003). Lean Software Development: An Agile Toolkit, Addison-Wesley Professional