2022-05-02

New Effort and Schedule Estimation Models for Agile Processes in the U.S. DoD

Madachy, Raymond; Rosa, Wilson; Clark, Bradford K.

Monterey, California. Naval Postgraduate School

SYM-AM-22-072

Excerpt from the Proceedings of the Nineteenth Annual Acquisition Research Symposium

Acquisition Research: Creating Synergy for Informed Change

May 11–12, 2022

Published: May 2, 2022

Prepared for the Naval Postgraduate School, Monterey, CA 93943.

Acquisition Research Program
Department of Defense Management
Naval Postgraduate School

The research presented in this report was supported by the Acquisition Research Program at the Naval Postgraduate School.

To request defense acquisition research, to become a research sponsor, or to print additional copies of reports, please contact any of the staff listed on the Acquisition Research Program website (www.acquisitionresearch.net).

Acquisition Research Program
Department of Defense Management
Naval Postgraduate School

New Effort and Schedule Estimation Models for Agile Processes in the U.S. DoD

Raymond Madachy—is a Professor in the Systems Engineering Department at the Naval Postgraduate School. His research interests include system and software cost modeling for total ownership cost; affordability and tradespace analysis; modeling and simulation of systems and software engineering processes, and the DoD acquisition process; integrating systems engineering and software engineering disciplines, and empirical-based research with process simulation. His books include Software Process Dynamics, co-authoring Software Cost Estimation with COCOMO II, Software Cost Estimation Metrics Manual for Defense Systems, and What Every Engineer Should Know about Modeling and Simulation. He is currently writing Systems Engineering Principles for Software Engineers. [rjmadach@nps.edu]

Wilson Rosa—serves as Assistant Director at the Department of Homeland Security Cost Analysis Division. He earned a PhD and an MS in Engineering Management from the George Washington University and a BS in Mechanical Engineering from the University of Puerto Rico. Rosa completed the Senior Managers in Government Executive Leadership Program at the Harvard Kennedy School. [wilson.rosa@hq.dhs.gov]

Bradford K. Clark—is Vice-President of Software Metrics Inc., a Virginia based consulting company. His area of expertise is in software cost and schedule data collection, analysis and modeling. Clark received a PhD in Computer Science in 1997 from the University of Southern California. He co-authored a book with Barry Boehm and others on Software Cost Estimation with COCOMO II. Clark is a former U.S. Navy A-6 Intruder pilot. [brad@software-metrics.com]

Abstract

The DoD's new software acquisition pathway prioritizes speed of delivery, advocating agile software processes. Estimating the cost and schedule of agile software projects is critical at an early phase to establish baseline budgets and to select competitive bidders. The challenge is that common agile sizing measures such as story points and user stories are not practical for early estimation as these are often reported after contract award in DoD.

This study provides a set of parametric effort and schedule estimation models for agile projects using a sizing measure that is available before proposal evaluation based on data from 36 DoD agile projects. The results suggest that initial software requirements, defined as the sum of functions and external interfaces, is an effective sizing measure for early estimation of effort and schedule of agile projects. The models' accuracy improves when application domain groups and peak staff are added as inputs.

Keywords— Agile software processes, Cost estimation, Requirements/Specification, Software acquisition, Time estimation

Introductionl

In the United States Department of Defense (DoD), the cost and schedule estimation of agile software development projects is more critical early in the life cycle when limited data is available. These estimates are needed for evaluating contractor cost proposals (Alleman et al., 2003) and to establish initial program budgets and schedules. Accurate estimates (Jorgensen, 2005; Nan & Harter, 2009; Shepperd & Schofield, 1997) help minimize cost overruns and schedule delays (Pendharker et al, 2005; Pillai & Sukumaran, 1997).

Since 2003, more than 1,000 software project data reports (DoD, 2020b) have been collected in the DoD. Of those, less than 5% were identified as agile. The lack of agile software project data has hindered the DoD's ability to implement accurate estimating methods and to articulate whether adopting agile could result in significant savings (Molokken-Ostvold & Jorgensen, 2005). The problem is compounded as agile software processes are increasingly used in the DoD, and acquisition practices must keep pace with these changes.

Studies on agile estimation have either used story points (Bilgaiyan et al., 2016; Choetkiertikul et al., 2018; Choetkiertikul et al. 2019; Owais & Ramakishore, 2016; Nguyen-Cong & Tran-Cao, 2013; Popli & Chauahn, 2015; Usman & Britto, 2016), user stories (Alleman et al., 2003; Saini et al., 2018), function points analysis (Bilgaiyan et al. 2016; Garg & Gupta, 2015; Kang et al., 2010; Nguyen-Cong & Tran-Cao, 2013; Usman & Britto, 2016), use case points (Bilgaiyan et al. 2016; Nguyen-Cong & Tran-Cao, 2013; Usman & Britto, 2016), COSMIC method (Nguyen-Cong & Tran-Cao, 2013), or object-oriented (Alshayeb & Li, 2003) artifacts as primary sizing measures. A second considerably smaller set of studies (Choetkiertikul et al., 2018; Choetkiertikul et al., 2019; Owais & Ramakishore, 2016; Popli & Chauhan, 2014) revealed story points is related to schedule. Although these are widely accepted agile sizing measures, using them at early phases is challenging (Tanveer et al., 2016) as these are typically available later in the life cycle (Choetkiertikul et al., 2019; Jones, 2013; Malik, 2011; Nassif et al, 2013). In the DoD, these sizing measures are provided by developers after contract award (Kaya & Demirors, 2011; Ochodek et al., 2011). Consequently, there is a dire need for early phase cost models (Jorgensten & Grimstad, 2011; Popli & Chauhan, 2013) to help programs get funding before contracting for agile software development projects.

This study contributes to the knowledge base by introducing simple models for estimating effort and duration of agile software development projects at an early phase. An important distinction of this approach is choosing sizing measures (Abraham & Insfran, 2008; DoN, 2010; Malik & Boehm, 2011; Sharma & Kushwaha, 2010, 2012) as model inputs (Ebrahimi, 1999) that are typically available early in the project's life cycle regardless of the development process. The sizing measure in this study is defined as the sum of initial functional requirements plus initial external interfaces. This is pragmatic as these sizing artifacts are the only ones available in the DoD for budget and proposal evaluation.

This study will address the following research questions:

1. Do initial, as opposed to final, software requirements, defined as functional plus external interface requirements, relate to final agile development effort?
2. Do initial software requirements along with super domain, relate to final agile development effort? Super domain is defined as group of application domains with similar software characteristics. For example, vehicle payload and vehicle control are part of the same super domain called real-time embedded.
3. Do initial software requirements along with initial peak staff and super domain relate to final agile development effort?
4. Do initial software requirements relate to the final agile development schedule?
5. Do initial software requirements along with super domain relate to the final agile development schedule?
6. Do initial software requirements along with initial peak staff and super domain relate to final agile software development schedule?

Background

Motivation to Adopt Agile in the DoD

The recent adoption of agile in the DoD has been triggered by the need to move from a capabilities-based to a threat-based acquisition model to counter the rapid growing adversary capabilities. The DoD's traditional development process, based on upfront detailed system requirements for the entire completed system, is inadequate to meet these challenges. Senior officials believe that greater adoption of agile would result in significant improved acquisition performance and the ability to quickly respond to adversary technological advancements and update DoD software systems accordingly.

In 2009, the U.S. Congress enacted the National Defense Authorization Act for Fiscal Year 2010 (2010 NDAA) requiring the DoD to implement a new acquisition process for IT systems (United States House of Representatives , 2018). This new process included principles of agile development such as early and continual involvement of the user, multiple rapidly executed increments or releases, early successive prototyping to support an evolutionary approach, and a modular open-systems approach. Since the enactment of the 2010 NDAA, an increasing number of non-IT DoD acquisition programs have turned to agile software development as a method for delivering new and enhanced capabilities to the warfighters on a rapid and repeatable basis, avoiding delays and cost overruns associated with traditional methods such as waterfall (United States House of Representatives, 2018).

Software Development Processes in the DoD

Waterfall is the traditional software development process in the DoD (United States Department of Defense Science Board, 2018). Waterfall development [66] begins with writing down the full software requirements at the lowest level. Those software requirements are finalized and set by the government before contract award and documented in the software requirements specifications (SRS) and interface requirements specification (IRS). After contract award, the developer will use the government's full software requirements to write the program code as well as the test cases. When the software passes all test cases, it is considered finished and ready for delivery to the government.

Agile Software Development in the DoD

Agile development in the DoD is defined as a life cycle model that employs iterative and incremental development with active user engagement (DoD, 2020a). The main goal is to allow for continuous development throughout the software's life cycle (United States Department of Defense Science Board, 2018). It involves continuous planning, continuous testing, continuous integration, continuous feedback, and continuous evolution of the product. Software is developed in short iterations, called time boxes, which typically last one to four weeks. Each iteration is like a miniature software project of its own and includes all the traditional software activities (planning, requirements analysis, design, coding, testing, and documentation) to release the mini increment of new functionality. At the end of each iteration, the team reevaluates project priorities.

Conversely, **hybrid agile** combines principles of waterfall (United States Department of Defense Science Board, 2018) and agile development. That is, Waterfall for decomposing the software requirements for the entire system upfront; (2) Agile after contract award for segmenting, testing, and delivering the software in short iterations. This hybrid model is suitable for legacy systems (e.g., KC-46A Tanker) that are transitioning to agile, or in fixed-price contracts (e.g., technology demonstration and sustainment) where requirements are set or fully defined before award.

Scrum is the most widely used in the DoD as majority of agile projects are managed by small teams. However, DevSecOps (United States Department of Defense Chief Information Officer, 2019) is gradually becoming the preferred framework as new defense policies (DoD, 2020a) are aiming at applying security throughout the software life cycle in a cloud-based environment.

Agile Requirements Decomposition in the DoD

Most agile software projects in the DoD start by establishing high-level program goals and high-level software requirements (functional, non-functional, interfaces) and considered finished when the program goals have been met. Those high-level software requirements are written by the government (DoN, 2010) and documented in the SRS and IRS. After contract award, the developer will enter those high-level requirements in the product backlog, rewrite and decompose them into user stories, and continuously refine them (add, delete, modify) as small segments of software are developed and presented to the government for feedback.

Research Method

Instrumentation

The data collection questionnaire in the study was obtained from an existing one: *Software Resource Data Report (SRDR)* form (DoD, 2020b; Lipkin, 2011). SRDR is the primary source for actual software industrial data for the DoD and can be obtained by DoD analysts via the Cost Assessment Data Enterprise (CADE) repository (https://cade.osd.mil) owned by the Office of the Secretary of Defense for Cost Assessment and Program Evaluation (OSD CAPE). The SRDR is a regulatory contract reporting requirement (DoD, 2020b) for defense software developers.

The SRDR is used to obtain both the estimated and actual characteristics of new or upgrade software projects. The developer submits an Initial SRDR shortly after contract award and a Final SRDR after contract completion. These constitute a contract data deliverable for contractors that formalizes the reporting of software metrics and resource data. The SRDR questionnaire and data item description and form can be downloaded via the links below.

https://cade.osd.mil/content/cade/files/csdr/dids/current/dd3026-1_2019.XLSX

https://cade.osd.mil/Content/cade/files/csdr/guidance/DI-MGMT-82035A_SRDR%20Report.pdf

In the SRDR questionnaire, developers are required to indicate whether their development process is agile or hybrid agile (e.g., waterfall for Architecture and Requirements, followed by agile for design, code, and unit test).

In the SRDR questionnaire, developers are also required to report the total software requirements by category as shown in Table 2. The Initial SRDR includes the initial software requirements that were baselined at early phase. The Final SRDR includes the final software requirements at contract end. The final includes baselined plus changed requirements (added, modified, deleted) as a result of continuous refinements after award.

Table 1. Software Requirements Reported

Requirement Type	Initial SRDR	Final SRDR
Total Requirements	X	X
Functional Requirements		
Baselined	X	X
Added		X
Modified		X
Deleted		X
External Interface Requirements		
Baselined	X	X
Added		X
Modified		X
Deleted		X
Privacy Requirements	X	X
Security Requirements	X	X
Safety Requirements	X	X

Population and Sample

The sample includes 36 agile projects delivered for the DoD from 2008 to 2019. This study focused on completed agile software projects reported as computer software configuration items. The paragraphs below describe how these projects were characterized in terms of software requirements, operating environment, and super domain.

Operating Environment: The dataset was grouped into operating environments (Clark & Madachy, 2015; DoD, 2020b) as shown In Table 3. Operating environment is the host platform in which the developed software system operates. The dataset did not contain projects developed for space systems.

Table 2. Operating Environment

Type	Definition
Surface Fixed	Software is at a fixed site.
Air Vehicle	Software embedded as part of an aircraft or drone.
Sea System	software is embedded as part of a surface or underwater boat/ship or boat.
Ordnance System	Software embedded as part of a rocket or ordnance.
Missile System	Software embedded as part of a missile system

Super Domain: The dataset was grouped into super domains as shown in Table 4. The raw dataset was initially reported by application domains (Clark & Madachy, 2015; DoD, 2020b; Madachy et al., 2011; Rosa et al., 2014a, 2014b). The dataset was then stratified into four general complexity zones called super domains. This stratification was adopted from our recent work (Rosa et al., 2017). The application domains to super domain mapping are shown in Table 4.

Table 3. Super Domain Taxonomy

Super Domain	Definition	Application Domains included:
Support (SUPP)	Support software assists with operator training and software testing.	Software Tools Training
Automated Information Systems (AIS)	Provides information processing services to humans or other applications. Allows authority to exercise control and access to typical business processes.	Enterprise Resource Planning (ERP) Custom AIS Software Mission Planning
Engineering (ENG)	Take outputs of real-time software and further process them to provide human consumable information or automated control of devices.	Test Equipment Scientific Simulation Process Control System Software
Real-Time (RTE)	Most constrained type of software. These are specific solutions limited by system characteristics such as memory size, performance, or battery life. These take the most time and effort due to very high reliability or hardware constraints.	Communications Real Time Embedded Command & Control Vehicle Control Vehicle Payload Signal Processing Microcode/ Firmware

Variables in the Study

The variables examined are described in Table 5.

Table 4. Variable Names and Definitions

Variable	Type	Definition
Effort (E)	Dependent	Actual labor hours associated to all software activities: requirements analysis, architectural design, coding, Software Integration, Software Qualification Testing, Software Support Processes
Schedule (TDEV)	Dependent	Actual development time (in months) to complete all software activities from Software Requirements Analysis through Software Qualification Test
Initial Software Requirements (REQ)	Independent	Initial functional requirements + initial external interface requirements reported in the Initial SRDR Developer Report
Peak Staff (Staff)	Independent	Initial peak staff measured in terms of full-time equivalents reported in the *Initial SRDR Developer Report.*
Super Domain	Categorical (Dummy)	Treatment of 4 (r) super domains required the addition of 3 (r-1) dummy variables : D1 = 1 if AIS, 0 if SUPP or otherwise D2 = 1 if ENG, 0 if otherwise D3 = 1 if RTE, 0 if otherwise

Model Selection

The model equation forms were chosen based on examining normal probability plots generated using the Cost Analysis Statistical Package (TECOLOTE Inc., 2020). The selection steps for the effort and schedule model forms are summarized below.

Figure 1 shows the residual plot for the linear regression of effort versus initial software requirements. The dataset does not appear to be normally distributed as residuals do not fall on the normality line. Consequently, lognormal regression was chosen for the three effort models.

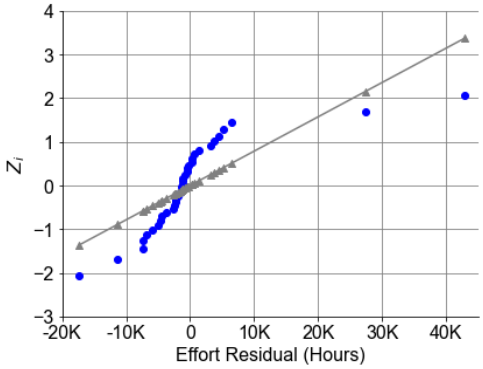

Figure 1. Effort Normal Probability Plot (Linear)

Figure 2 shows the residual plot for the linear regression of schedule versus initial software requirements. The dataset does not appear to be normally distributed as residuals do not fall on the normality line. Consequently, lognormal regression was chosen for the three schedule models.

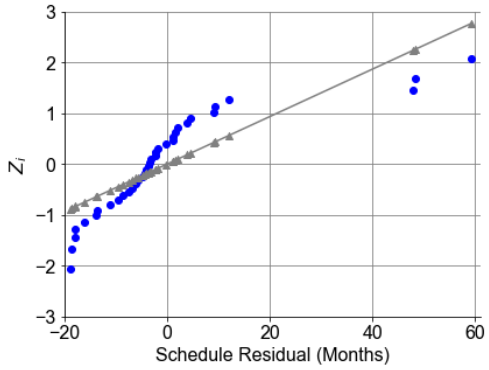

Figure 2. Schedule Normal Probability Plot (Linear)

Model Validation

The regression models were validated using the measures listed in Table 6.

Table 5. Model Validity Measures

Measure	Description
R^2	Coefficient of determination is the percentage of variation in the response explained by the model. (Matson et al., 1994)
R^2 (adj)	Adjusted R2 is the percentage of the variation in the response explained by the model, adjusted for the number of predictors in the model relative to the number of observations.
R^2 (pred)	Predicted R2 is a cross validation method that involves removing each observation from the dataset, estimating the regression equation, determining how well the model predicts the removed observation, and repeats this for all data points.
P-value	Statistical significance through the coefficient alpha (α = 0.05).
VIF	Variance Inflation Factor indicates if multicollinearity is present in a multi-regression analysis; VIF lower than 10, indicates no multicollinearity.
SEE	Standard Error of the Estimate is the difference between observed and the estimated effort. SEE is to linear models as standard deviation is to sample means.
F-test	F test is the square of the equivalent t test; the bigger it is, the smaller the probability that difference could occur by chance.
MMRE	Mean Magnitude of Relative Error is an indicator of model's accuracy: Low MMRE= high accuracy. (Mukhopadhyay & Kekre, 1992)

Dataset Demographics

The sample was identified as 36 agile projects completed from 2008 to 2019 (Figure 3), involving six operating environments (Figure 4), and four super domains (Figure 5). The majority of the projects were completed in the last six years and most of the software projects were hosted at a surface fixed site (e.g., mission operations center, data center).

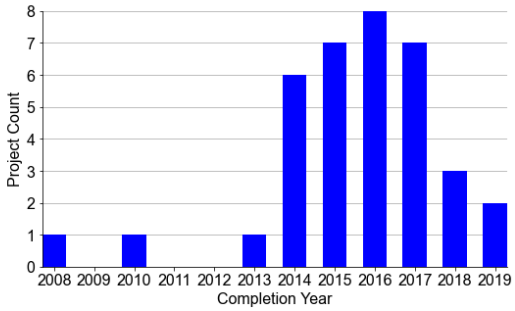

Figure 3. Agile Project Completion Year

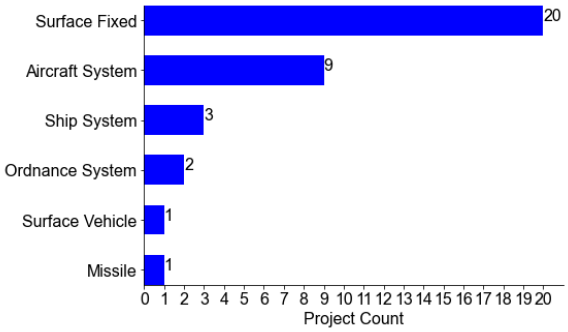

Figure 4. Operating Environment

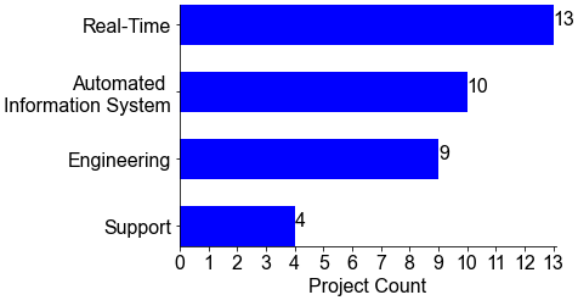

Figure 5. Super Domain

Figure 6 displays the agile process for the 36 projects. Projects were characterized as agile or hybrid agile. The projects identified as **hybrid agile** used waterfall process for requirements analysis, and agile process for design, code, unit test, and integration. This information was obtained from developer's data item descriptions provided as a supplement to their SRDR submission. We also contacted the developers to validate and verify their responses.

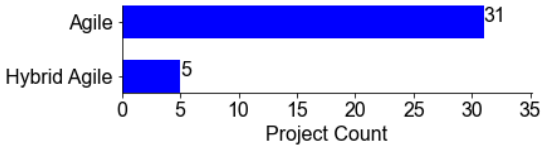

Figure 6. Agile Process

Figure 7 displays the agile framework for the dataset. The majority of the projects used Scrum. This information was obtained from developer's SRDR questionnaire, data item description, and proprietary documents describing their processes. We also contacted the developers to validate and verify their responses.

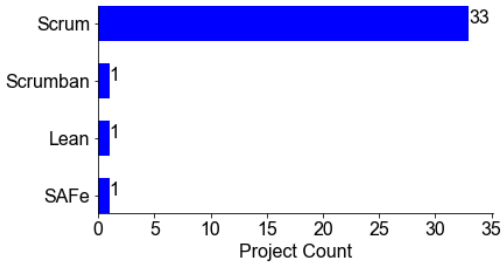

Figure 7. Agile Framework

Descriptive Statistics

Figure 8 is a histogram of the actual software development effort for the agile projects. The average software development effort for the sample was 99,959 hours and standard deviation of 134,641. The distribution appears to be right skewed, confirming the non-normal distribution of effort data as previously shown in Figure 1.

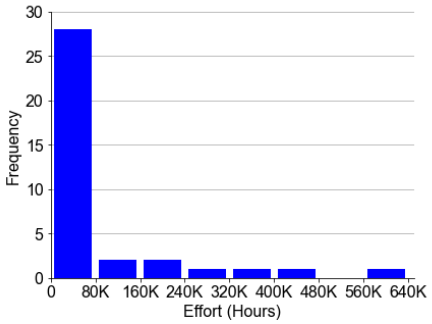

Figure 8. Effort Histogram

Figure 9 is a histogram of the actual development time for the agile projects. The average software development completion time for the sample was 26.95 months and standard deviation of 19.73. The distribution again confirms the non-normal distribution of the data as previous shown in Figure 2.

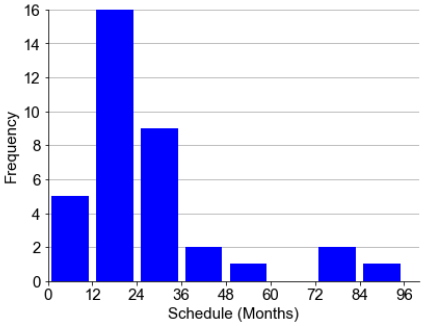

Figure 9. Schedule Histogram

Figure 10 provides a histogram of initial software requirements. The average total software requirements for the sample was 798. The distribution is lognormal.

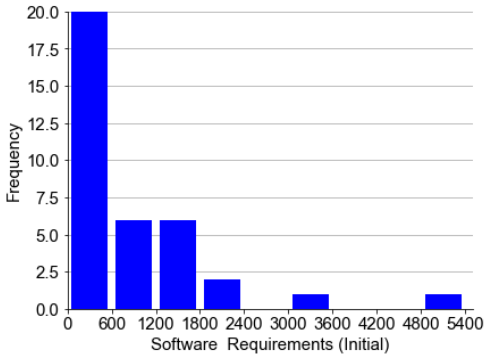

Figure. 10. Software Size Histogram

Figure 11 shows a histogram of peak staff for the agile project dataset showing Full-Time Equivalent (FTE) staff. The average peak staff for the sample was 31. The project with largest peak staff was developed using SAFe. This data has a lognormal distribution.

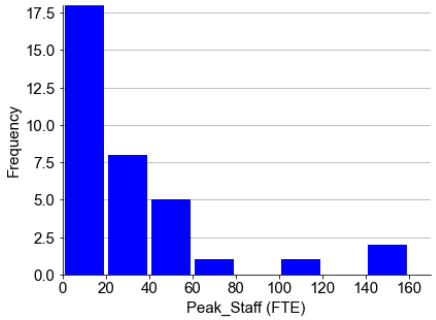

Figure 11. Peak Staff Histogram

Figure 12 provides a histogram of the requirements volatility (RVOL) for the agile project dataset. RVOL is the sum of requirements changes (added, modified, and deleted) divided by the total number of baselined requirements. Baselined requirements are those initial software requirements used in this study. The average RVOL for the sample was 19%. More than half experienced RVOL ≥ 12%. This confirms the notion that software requirements for agile projects in DoD are not fixed and evolve over time.

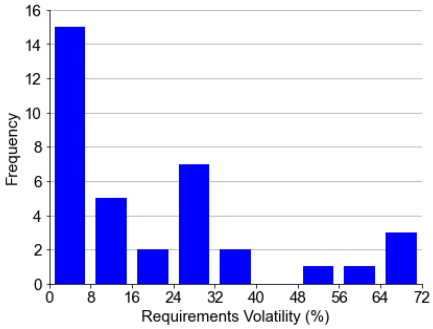

Figure 11. Requirements Volatility Histogram

Results

This section describes the resulting effort and schedule models associated with Research Questions (RQ) 1 through 6. Loglinear regression was performed for all models using the Cost Analysis Statistical Package (TECOLOTE Inc., 2020). Log-normal ordinary least squares (OLS) regression was used to create the multiplicative models in this study. The data is transformed into log-space and the coefficients are derived using OLS regression. The derived coefficients for each predictor variable are treated as exponents and the regression intercept is transformed back into unit-space with an anti-log. Alpha (α = 0.05) is used in evaluating the p-values for each model. These models are applicable for DoD agile software project size ranging between 10 and 5000 initial software requirements, and a peak staff between 1 and 150 full-time equivalents. The sample size, however, may impact models' ability to detect statistical effects with any greater power.

Effort Model 1

RQ 1: Do initial, as opposed to final, software require-ments, defined as functional plus external interface requirements, relate to final agile development effort?

Equation (1) predicts effort for agile software development projects as a function of total initial software requirements.

$$E = 1006 x REQ^{0.65} \quad (1)$$

Where:

E = Final software development hours

REQ = FUNC + EIF

And:

FUNC = Initial Functional Requirements

EIF = Initial External Interface Requirements

Figure 13 shows the residual plot for equation (1). The residuals approximate a straight line. This suggests that loglinear regression is valid for modeling.

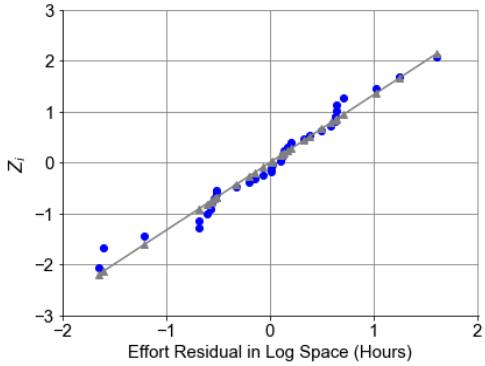

Figure 12. Normal Probability Plot for Equation (1)

Table 7 reports the coefficient statistics, goodness-of-fit test, and analysis of variance for Equation (1). The high t-statistics and low p-value suggest that initial software requirements are

strongly correlated to effort. The result also adds insight to the fact that initial functional requirements plus initial external interface requirements are effective in estimating effort for agile projects. The small difference between adjusted and predicted R^2 suggest that the model predicts new observations as well as it fits the existing data. However, the low adjusted R^2 of 63% suggests adding additional variables for a better model fit.

Table 6. Regression Analysis Results for Equation (1)

Coefficient Statistics Summary

Term	Coef	T-Statistic	P-value	VIF
Intercept	6.9138	13.7	0.0000	***
REQ	0.6500	7.8	0.0000	***

Goodness-of-Fit Test

SE	R^2	R^2 (adj)	R^2 (pred)	MMRE
0.7274	64.11%	63.06%	60.29%	68.16%

Analysis of Variance

Source	DF	Sum of Sq.	Mean Sq.	F-stat
Regression	1	32.14	32.14	60.74
Residual	34	17.99	0.52	
Total	35	50.13		

Effort Model 2

RQ 2: Do initial software requirements along with super domain, relate to final agile development effort?

Equation (2) predicts effort for agile software development projects using total initial software requirements as predictor and super domain as categorical variables (D1, D2, D3).

$$E = 200.7 x REQ^{0.7182}(3.0^{D1})(3.6^{D2})(5.1^{D3}) \qquad (2)$$

Where:

E = Final development labor hours

REQ = FUNC + EIF

And:

FUNC = Initial Functional Requirements

EIF = Initial External Interface Requirements

D1 = 1 if AIS, 0 if otherwise

D2 = 1 if ENG, 0 if otherwise

D3 = 1 if RTE, 0 if otherwise

If D1, D2 and D3 are zero, Equation (2) predicts effort for the SUPP super domain

Figure 14 shows the normal residual plot for Equation (2). Loglinear regression is valid for this model as its residuals approximate a normal distribution.

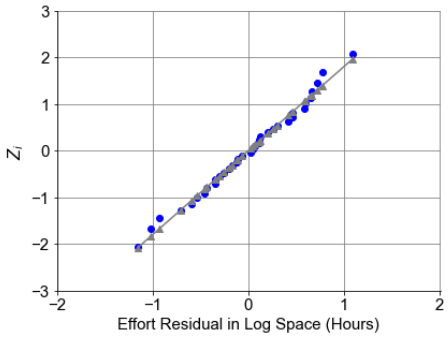

Figure 13. Normal Probability Plot for Equation (2)

Table 8 shows the analysis results for Equation (2). The t-statistics and p-values suggest that super domain is strongly correlated to effort, and low VIF values indicates no sign of multi-collinearity. The small difference between adjusted and predicted R^2 also suggest the model predicts new observations as well as it fits the existing data. This model shows higher adjusted R^2 and lower MMRE than Equation (1); signifying that adding super domain categorical variables to Equation (1) improves accuracy and fit.

Table 7. Regression Analysis Results for Equation (2)

Coefficient Statistics Summary

Term	Coef	T-Statistic	P-value	VIF
Intercept	5.3019	9.7	0.0000	
REQ	0.7182	10.2	0.0000	1.2
D1	1.0929	3.2	0.0028	2.5
D2	1.2728	3.5	0.0013	2.7
D3	1.6332	4.9	0.0000	2.8

Goodness-of-Fit Test

SE	R^2	R^2 (adj)	R^2 (pred)	MMRE
0.5676	80.08%	77.51%	73.60%	47.52%

Analysis of Variance

Source	DF	Sum of Sq.	Mean Sq.	F-stat
Regression	4	40.14	10.0366	31.14
Residual	31	9.98	0.3222	
Total	35	50.13		

Effort Model 3

RQ3: Do initial software requirements along with initial peak staff and super domain relate to final agile de-velopment effort?

Equation (3) predicts software development effort for agile projects using peak staff and total initial software requirements as predictors, while super domain as categorical variables (D1, D2, D3).

$$E = 173.2 x REQ^{0.539} Staff^{0.463} (2.3^{D1})(3.7^{D2})(3.9^{D3}) \quad (3)$$

Where:

E = Final development labor hours

REQ = FUNC + EIF

Staff = Initial peak staff in full-time equivalent

And:

FUNC = Initial Functional Requirements

EIF = Initial External Interface Requirements

D1 = 1 if AIS, 0 if otherwise

D2 = 1 if ENG, 0 if otherwise

D3 = 1 if RTE, 0 if otherwise

If D1, D2 and D3 are zero, Equation (3) predicts effort for the SUPP super domain

Figure 15 shows the transformed normal residual plot for Equation (3). Loglinear regression is valid for this model as its residuals approximate a straight line.

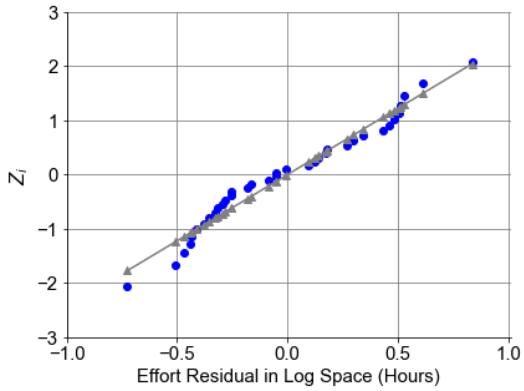

Figure 14. Normal Probability Plot for Equation (3)

Table 9 shows the statistical analysis for Equation (3). The t-statistics suggests that all three variables are strongly correlated to effort; with no signs of multicollinearity. The small difference between adjusted and predicted R^2 suggest the model predicts new observations as well as it fits the existing data. This model shows higher adjusted R^2 and lower MMRE than Equations (1) and (2) signifying that this model achieves the highest accuracy and best fit when all three variables are added to the regression.

Table 8. Regression Analysis Results for Equation (3)

Coefficient Statistics Summary

Term	Coef	T-Statistic	P-value	VIF
Intercept	5.1543	12.7	0.0000	
REQ	0.5390	8.6	0.0000	1.7
Staff	0.4631	5.2	0.0000	1.8
D1	0.8362	3.3	0.0025	2.6
D2	1.3025	4.9	0.0000	2.7
D3	1.3696	5.5	0.0000	2.9

Goodness-of-Fit Test

SE	R^2	R^2 (adj)	R^2 (pred)	MMRE
0.4198	89.46%	87.70%	84.74%	33.85%

Analysis of Variance

Source	DF	Sum of Sq.	Mean Sq.	F-stat
Regression	5	44.8489	8.9698	50.90
Residual	30	5.2861	0.1762	
Total	35	50.1350		

Schedule Model 4

RQ4: Do initial software requirements relate to the final agile development schedule?

Equation (4) predicts software development time (in months) for agile projects as a function of total initial software requirements.

$$TDEV = 6.84xREQ^{0.202} \quad (4)$$

Where:

TDEV = Final development time (in months)

REQ = FUNC + EIF

And:

FUNC = Initial Functional Requirements

EIF = Initial External Interface Requirements

Figure 16 shows the transformed normal residual plot for Equation (4). Loglinear regression is valid for this model as its residuals follow approximate a normal distribution.

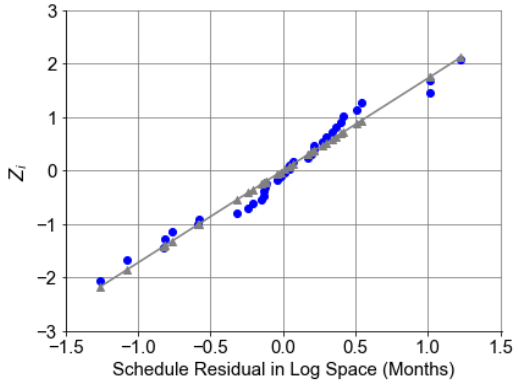

Figure 15. Normal Probability Plot for Equation (4)

Table 10 reports the coefficient statistics, goodness-of-fit test, and analysis of variance for Equation (4). The high t-statistics and low p-value suggests that initial software requirements are strongly correlated to development time (TDEV). However, the low R^2 is an indication that a schedule model only based on initial software requirements does not fit the data well. Adding additional predictors to the model may increase the R^2.

Table 9. Regression Analysis Results for Equation (4)

Coefficient Statistics Summary

Term	Coef	T-Statistic	P-value	VIF
Intercept	1.8998	4.89	0.0000	***
REQ	0.2029	3.16	0.0033	***

Goodness-of-Fit Test

SE	R^2	R^2 (adj)	R^2 (pred)	MMRE
0.5598	22.71%	20.44%	14.53%	46.66%

Analysis of Variance

Source	DF	Sum of Sq.	Mean Sq.	F-stat
Regression	1	3.1306	3.1306	9.99
Residual	34	10.6543	0.3134	
Total	35	13.7849		

Schedule Model 5

RQ 5: Do initial software requirements along with super domain relate to final agile development schedule?

Equation (5) predicts software development time (TDEV) for agile projects using total initial software requirements as predictor and super domain as categorical variable (D1, D2, D3).

$$TDEV = 1.64xREQ^{0.272}(2.1^{D1})(2.9^{D2})(4.0^{D3}) \quad (5)$$

Where:

TDEV = Final development time (in months)

REQ = FUNC + EIF

And:

FUNC = Initial Functional Requirements

EIF = Initial External Interface Requirements

D1 = 1 if AIS, 0 if otherwise

D2 = 1 if ENG, 0 if otherwise

D3 = 1 if RTE, 0 if otherwise

If D1, D2 and D3 are zero, Equation (5) predicts effort for the SUPP super domain

Figure 17 shows the transformed normal residual plot for Equation (5). Loglinear regression is valid for this model as its residuals approximate a normal distribution.

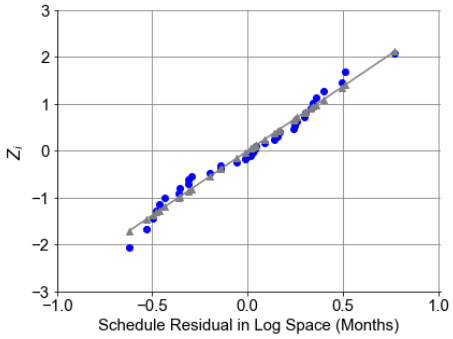

Figure 16. Normal Probability Plot for Equation (5)

Table 11 shows the analysis results for Equation (5). The t-statistics and p-values suggest that super domain is strongly correlated to effort, and low VIF values indicate no multicollinearity. The small difference between adjusted and predicted R^2 indicates the model may predict new observations as well as it fits the existing data. This model shows higher adjusted R^2 (20% → 65%) and lower MMRE (46%→ 30%) compared to Equation (4); signifying that adding super domain categorical variables improves its accuracy and fit.

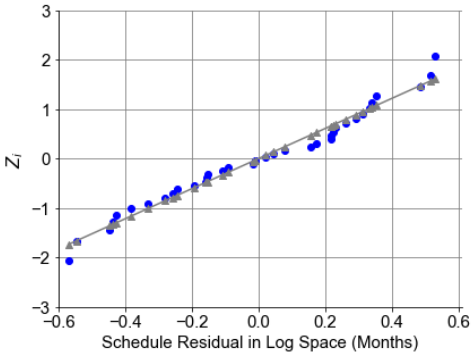

Figure 17. Normal Probability Plot for Equation (6)

Table 10. Regression Analysis Results for Equation (5)

Coefficient Statistics Summary

Term	Coef	T-Statistic	P-value	VIF
Intercept	0.4980	1.40	0.1724	
REQ	0.2722	5.97	0.0000	1.2
D1	0.7639	3.49	0.0015	2.5
D2	1.0972	4.69	0.0001	2.7
D3	1.4072	6.56	0.0000	2.8

Goodness-of-Fit Test

SE	R^2	R^2 (adj)	R^2 (pred)	MMRE
0.3691	69.37%	65.42%	59.14%	30.07%

Analysis of Variance

Source	DF	Sum of Sq.	Mean Sq.	F-stat
Regression	4	9.5625	2.3906	17.5514
Residual	31	4.2224	0.1362	
Total	35	13.7849		

Schedule Model 6

RQ6: Do initial software requirements along with initial peak staff and super domain relate to final agile software development schedule?

Equation (6) predicts software development time (TDEV) for agile projects using peak staff and total initial software requirements as predictors, while super domain as categorical variables (D1, D2, D3).

$$TDEV = 1.74xREQ^{0.345}Staff^{-0.189}(2.3^{D1})(3.0^{D2})(4.5^{D3}) \quad (6)$$

Where:

TDEV = Final development time (in months)
REQ = FUNC + EIF
Staff = Initial peak staff in full-time equivalent

And:

FUNC = Initial Functional Requirements
EIF = Initial External Interface Requirements
D1 = 1 if AIS, 0 if otherwise
D2 = 1 if ENG, 0 if otherwise
D3 = 1 if RTE, 0 if otherwise

If D1, D2 and D3 are zero, Equation (5) predicts effort for the SUPP super domain.

Figure 18 shows the transformed normal residual plot for Equation (6). Loglinear regression is valid for this model as its residuals approximate a normal distribution.

Table 12 shows the statistical analysis for Equation (6). The t-statistics shows all three variables are strongly correlated to TDEV; with no signs of multicollinearity. The small difference between adjusted and predicted R^2 suggests the model predicts new observations as well as it fits the existing data. This model shows higher adjusted R^2 and lower MMRE than Equations (4) and (5); suggesting that this model achieves highest accuracy and best fit.

Table 11. Regression Analysis Results for Equation (6)

Coefficient Statistics Summary

Term	Coef	T-Statistic	P-value	VIF
Intercept	0.5585	1.7	0.0986	
REQ	0.3456	6.9	0.0000	1.7
Staff	-0.1896	-2.6	0.0135	1.8
D1	0.8690	4.2	0.0002	2.6
D2	1.0850	5.1	0.0000	2.7
D3	1.5151	7.5	0.0000	2.9

Goodness-of-Fit Test

SE	R^2	R^2 (adj)	R^2 (pred)	MMRE
0.3383	75.09%	70.94%	63.24%	27.50%

Analysis of Variance

Source	DF	Sum of Sq.	Mean Sq.	F-stat
Regression	5	10.3508	2.0702	18.08
Residual	30	3.4340	0.1145	
Total	35	13.7849		

Discussion of Results

The resulting statistics add insight to the notion that initial, as opposed to final, software requirements, when defined as functional and external interface requirements in the SRS and IRS, is good at predicting effort and time for DoD agile projects.

The multi-variable models ((3) and (6)) based on requirements, peak staff and super domain as inputs, perform better than single-variable models ((1) and (4)) using requirements alone to predict effort or schedule.

Model Usefulness and Limitations

The models in this study are useful for effort and schedule estimates at proposal evaluation or before. Since these models were derived using OLS in log-space (using the natural log), the output represents an estimate at the 50% confidence level in log-space. To understand the uncertainty in the different models, the model results should be displayed as a 95% confidence interval rather than a single value. The confidence interval is derived in log-space using two times the model's standard error (SEE). For example, the 95% confidence interval for the output from Equation (3) and the RTE super domain is expressed as follows:

$$\mathrm{E\ (Low)} = 173.2 x REQ^{0.539} Staff^{0.463} x(3.9) - (2 \times .4198)$$

$$\mathrm{E\ (High)} = 173.2 x REQ^{0.539} Staff^{0.463}\ x(3.9) + (2 \times .4198)$$

The result is transformed into unit-space by taking the natural anti-log. The confidence intervals derived from these models can help program managers independently assess whether their software development contract is in breach status. For example, if the contract's latest revised schedule or cost estimate exceeds the prediction interval's upper bound, the acquisition decision authority may declare a contract breach or restructure the program.

Threats to Validity

Possible threats to validity are summarized next:

Internal Validity Threats:

- The dataset timeframe (2008–2019) raises potential issues as the earlier projects (2008, 2010, 2013) were developed using agile processes tailored to fit the developer's need. It is likely that agile processes evolved during the 11-year timeframe.
- This study is on frameworks commonly considered "agile" and a focus on only one of the frameworks may produce different results.
- The developers "self-reported" the data in the SRDR questionnaire. We verified 80% (29 out of 36) of the project data by contacting the developers and following up for additional information. Whether the unverified projects (20%) are accurate remains unanswered.

External Validity Threats:

- The data from this study came from DoD contracts that exceeded $50 million in value for the total contract. The performance of these large companies may not be generalizable to other organizations.
- The initial software requirement counts came from SRS and IRS, a common artifact in DoD acquisition. Non-DoD organizations may not use an SRS to state their requirements at early phase.
- These models proved to be effective in estimating total development hours and duration for agile projects reported at the release level in the DoD. However, we cannot generalize beyond

this group for several reasons. First, majority of the projects were developed using Scrum and none in XP. Second, the initial software requirements were developed at a high-level and only included functions and interfaces. Third, models may not be suitable for projects using DevOps as the reported effort in the dataset does not capture sustainment activities.

Construct Validity Threats:

- A sample size of 36 agile projects poses a threat to statistical conclusion as it does not allow for detecting effects with greater power. A larger sample size is needed for confirmatory hypothesis testing.
- Projects in this study reported their initial software requirements using the traditional SRS and IRS templates. After contract award, those initial requirements were rewritten into stories and continuously refined using agile processes. Whether the projects should be classified as agile or hybrid agile remains debatable.

Example Applications

The effort and schedule models are used here for estimating two examples: a Radar Display Manager and a Testing Tool. The examples show how to take information on requirements, peak staffing, and the application super domain and use it in the models. Of particular interest is how super domains are represented in each model as there are four super domains but only three super domain variables in the effort and schedule models.

Radar Display Manager

In the first example, there is a need to estimate the software development effort and schedule for Radar Display Manager software. There are 207 Initial Requirements and 23 External Interface Requirements for a total of 230 requirements. The estimated Peak Staffing is 16 people.

The Radar Display Manager is in the Real Time Embedded (RTE) super domain. Effort Model (3) takes requirements, peak staffing, and super domain as inputs. Super domains are represented by the D1, D2, and D3 variables for the AIS, ENG, and RTE super domains respectively. The super domain variables have a value of either zero (0) or one (1). In this example, the Radar Display Manager is in the RTE super domain, variable D3 is set to 1 and the other two variables are set to 0.

Therefore, the effort estimation model for the Radar Display Manager is:

$$E = 173.2\ x\ 230^{0.539}\ x\ 16^{0.463}\ x\ (2.3^{0})\ x\ (3.7^{0})\ x\ (3.9^{1})$$

$$E = 173.2\ x\ 18.75\ x\ 3.6\ x\ 1\ x\ 1\ x\ 3.9$$

$$E = 45{,}595\ hours$$

For the Radar Display Manager software, the time to develop (TDEV) schedule estimation model is:

$$TDEV = 1.74\ x\ 230^{0.345}\ x\ 16^{-0.189}\ x\ (2.3^{0})\ x\ (3.0^{0})\ x\ (4.5^{1})$$

$$TDEV = 1.74\ x\ 6.53\ x\ 0.59\ x\ 1\ x\ 1\ x\ 4.5$$

$$TDEV = 30\ months$$

The estimate for the Radar Display Manager is 45,595 hours and 30 months. That is an average of 1,520 hours per month.

Testing Tool

In the next example, the effort and schedule need to be estimated for the software development of a Testing Tool. There are 31 Initial Requirements and five External Interface Requirements for a total of 36 requirements. The estimated Peak Staffing is four people.

The Testing Tool is in the Support (SUPP) super domain. There are variables in the effort and schedule models for the AIS (D1), ENG (D2), and RTE (D3) super domains but no variable for the SUPP super domain. This is because the base models for effort and schedule are for the SUPP super domain when the D1, D2, and D3 variables are given the value of zero (0).

The Testing Tool effort estimation model is represented as:

$$E = 173.2\ x\ 36^{0.539}\ x\ 4^{0.463}\ x\ (2.3^0)\ x\ (3.7^0)\ x\ (3.9^0)$$

$$E = 173.2\ x\ 6.9\ x\ 1.9\ x\ 1\ x\ 1\ x\ 1$$

$$E = 2{,}271$$

The TDEV schedule estimation model for the Testing Tool is:

$$TDEV = 1.74\ x\ 35^{0.345}\ x\ 4^{-0.189}\ x\ (2.3^0)\ x\ (3.0^0)\ x\ (4.5^0)$$

$$TDEV = 1.74\ x\ 3.44\ x\ 0.77\ x\ 1\ x\ 1\ x\ 1$$

$$TDEV = 4.6\ months$$

The estimate for the Testing Tool software is 2,271 hours, 4.6 months, and 494 hours per month.

Conclusion

The results add insight to the notion that initial, ***as opposed to final***, functional plus external interface requirements, when treated as sizing input along with a super domain categorical variable, are effective in predicting software development effort and schedule of DoD agile projects early in the life cycle; at the time when mainstream agile sizing metrics are not available for estimation in DoD and budget/schedule baselines are being established. These models can be used for building independent government cost estimates to crosscheck request for proposals.

The results also suggest that DoD contractors should consider adding peak staff along with initial software requirements (as defined in this study) and super domain as inputs when building effort and schedule models for their agile project cost proposal. These three inputs are generally obtained from: contract proposals (i.e., peak staff), request for proposals (i.e., super domain), and government provided requirements (i.e., initial software requirements).

References

Abraham, S. & Insfran, E. (2008, September 3–5). A metamodeling approach to estimate software size from requirements specifications. *Software Engineering and Advanced Applications, 2008. SEAA '08. 34th Euromicro Conference*, 465–475.

Alleman, G. B., Henderson, M. & Seggelke, R. (2003). Making Agile development work in a government contracting environment-measuring velocity with earned value. *Proceedings of the Agile Development Conference, 2003. ADC 2003*, 114–119.

Alshayeb, M. & Li, W. (2003, November). An empirical validation of object-oriented metrics in two different iterative software processes. *IEEE Transactions on Software Engineering*, *29*(11), 1043–1049.

Bilgaiyan, S., Mishra, S. & Das, M. (2016). A review of software cost estimation in Agile software development using soft computing techniques. *2016 2nd International Conference on Computational Intelligence and Networks (CINE)*, 112–117.

Choetkiertikul, M., Dam, H. K., Tran, T., Ghose, A. & Grundy, J. (2018, June 1). Predicting delivery capability in iterative software development. *IEEE Transactions on Software Engineering, 44*(6), 551–573.
Choetkiertikul, M., Dam, H. K., Tran, T., Pham, T., Ghose, A. & Menzies, T. (2019, July 1). A deep learning model for estimating story points. *IEEE Transactions on Software Engineering*, *45*(7), 637–656.
Clark, B. & Madachy, R. (Eds.). (2015, April). *Software cost estimation metrics manual for defense systems*. Software Metrics Inc. http://www.sercuarc.org/wp-content/uploads/2014/05/Software-Cost-Estimation-Metrics-Manual-for-Defense-Systems.pdf
Department of Defense. (2016, March). *Agile and earned value management: A program managers desk guide*. http://www.acq.osd.mil/evm/docs/PARCA
Department of Defense. (2020b, October). *DoD instruction 5000.87, operation of the software acquisition pathway*. https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500087p.PDF?ver=virAfQj4v_LgN1JxpB_dpA%3d%3d
Department of Defense. (2020b, November). *Software resource data report*. https://cade.osd.mil/content/cade/files/csdr/dids/current/dd3026-1_2019.XLSX
Department of Navy. (2010, September). *Software criteria and guidance for systems engineering technical reviews (SETR) supplement to guidebook for acquisition of Naval software intensive systems*. http://www.secnav.navy.mil/rda/OneSource/Documents/Program%20Assistance%20and%20Tools/Handbooks,%20Guides%20and%20Reports/Page%205/supplementtoguidebook.pdf
Ebrahimi, N. B. (1999, January–February). How to improve the calibration of cost models. *IEEE Transactions on Software Engineering, 25*(1), 136–140.
GAO. (2012, July). *Software development: Effective practices and federal challenges in applying Agile methods*. http://www.gao.gov/assets/600/593091.pdf
GAO. (2020, September). *Agile assessment guide: Best practices for Agile adoption and implementation* (GAO-20-590C). https://www.gao.gov/assets/710/709711.pdf
Garg, S., & Gupta, D. (2015). PCA based cost estimation model for agile software development projects. *2015 International Conference on Industrial Engineering and Operations Management (IEOM)*, 1–7.
Hastings, T. E. & Sajeev, A. S. M. (2001, April). A vector-based approach to software size measurement and effort estimation. *IEEE Transactions on Software Engineering, 27*(4), 337–350.
Heričko, M., & Živkovič, A. (2008, June). The size and effort estimates in iterative development. *Information and Software Technology, 50*(7–8), 772–781. http://www.sciencedirect.com/science/article/pii/S0950584907000870
Jones, C. (2013). Function points as a universal software metric. *SIGSOFT Software Engineering Notes 38, 4*(July 2013), 1–27.
Jorgensen, M. (2005, November). Evidence-based guidelines for assessment of software development cost uncertainty. *IEEE Transactions on Software Engineering, 31*(11), 942–954.
Jorgensen, M. & Grimstad, S. (2011, September–October). The impact of irrelevant and misleading information on software development effort estimates: A randomized controlled field experiment. *IEEE Transactions on Software Engineering, 37*(5), 695–707.
Kang, S., Choi, O. & Baik, J. (2010). Model-based dynamic cost estimation and tracking method for Agile software development. *2010 IEEE/ACIS 9th International Conference on Computer and Information Science*, 743–748.
Kaya, M., & Demirors, O. (2011). E-Cosmic: A business process model based functional size estimation approach. *Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on, Inf. Inst.*, 404–410.
Lipkin, I. (2011). *Test software development project productivity model* [Doctoral dissertation, University of Toledo].
Madachy, R., Boehm, B., Clark, B., Tan, T., & Rosa, W. (2011). U.S. DoD application domain empirical software cost analysis. *2011 International Symposium on Empirical Software Engineering and Measurement,* 392–395.
Malik, A. & Boehm, B. (2011, July). Quantifying requirements elaboration to improve early software cost estimation. *Information Sciences, 181(*13) 2747–2760. http://www.sciencedirect.com/science/article/pii/S002002550900526X

Malik, A. (2011). *Quantitative and qualitative analyses of requirements elaboration for early software size estimation* [Doctoral dissertation, University of Southern California].
Matson, J. E., Barrett, B. E., & Mellichamp, J. M. (1994, April). Software development cost estimation using function points. *IEEE Transactions on Software Engineering, 20*(4), 275–287.
Molokken-Ostvold, K. & Jorgensen, M. (2005, September). A comparison of software project overruns—flexible versus sequential development models. *IEEE Transactions on Software Engineering, 31*(9), 754–766.
Mukhopadhyay, T. & Kekre, S. (1992, October). Software effort models for early estimation of process control applications. *IEEE Transactions on Software Engineering, 18*(10), 915–924.
Nan, N. & Harter, D. E. (2009, September–October). Impact of budget and schedule pressure on software development cycle time and effort. *IEEE Transactions on Software Engineering, 35*(5), 624–637.
Nassif, A., Ho, D., & Capretz, L. (2013, January). Towards an early software estimation using log-linear regression and a multilayer perceptron model. *Journal of Systems and Software, 86*(1), 144–160. http://www.sciencedirect.com/science/article/pii/S0164121212002221
Nguyen-Cong, D. & Tran-Cao, D. (2013). A review of effort estimation studies in Agile, iterative and incremental software development. *The 2013 RIVF International Conference on Computing & Communication Technologies—Research, Innovation, and Vision for Future (RIVF)*, 27–30.
Ochodek, M., Nawrocki, J., & Kwarciak, K. (2011, March). Simplifying effort estimation based on use case points. *Information and Software Technology, 53(*3), 200–213. http://www.sciencedirect.com/science/article/pii/S095058491000176X
Owais, M. & Ramakishore, R. (2016). Effort, duration and cost estimation in agile software development. *2016 Ninth International Conference on Contemporary Computing (IC3), Noida*, 1–5.
Pendharkar, P. C., Subramanian, G. H., & Rodger, J. A. (2005, July). A probabilistic model for predicting software development effort. In *IEEE Transactions on Software Engineering, 31*(7), 615–624.
Pillai, K. & Sukumaran Nair, V. S. (1997, August). A model for software development effort and cost estimation. In *IEEE Transactions on Software Engineering, 23*(8), 485–497.
Popli, R. & Chauhan, N. (2013). A sprint-point based estimation technique in Scrum. *2013 International Conference on Information Systems and Computer Networks, Mathura*, 98–103.
Popli, R. & Chauhan, N. (2014). Agile estimation using people and project related factors. *2014 International Conference on Computing for Sustainable Global Development (INDIACom)*, 564–569.
Popli, R. & Chauahn, N. (2015). Managing uncertainty of story-points in Agile software. *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, 1357–1361.
Rosa, W. , Madachy, R., Boehm, B., & Clark, B. (2014a). 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.
Rosa, W., Clark, B., Boehm, B., & Madachy, R. (2014b, October). Simple-empirical software cost estimation. *29th International Forum on COCOMO and Systems/Software Cost Modeling [Online].* http://csse.usc.edu/new/wp-content/uploads/2014/10/Simple-Empirical-Software-Cost-Estimation_v2.pdf
Rosa, W., Madachy, R., Clark, B., & Boehm, B. (2017). Early phase cost models for Agile software processes in the U.S. DoD. *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM),* 30–37. doi:10.1109/ESEM.2017.10 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8170082&isnumber=8169971
Saini, A., Ahuja, L., & Khatri, S. K. (2018). Effort estimation of Agile development using fuzzy logic. *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO),* 779–783.
Sharma, A., & Kushwaha, D. (2010, October 5). Early estimation of software complexity using requirement engineering document. *SIGSOFT Software Engineering Notes,* 35.
Sharma, A., & Kushwaha, D. (2012). Estimation of software development effort from requirements based complexity. *Procedia Technology, 4*, 716–722.
Shepperd, M. & Schofield, C. (1997, November). Estimating software project effort using analogies. *IEEE Transactions on Software Engineering, 23*(11), 736–743.
Tanveer, B., Guzmán, L., & Engel, U. M. (2016). Understanding and improving effort estimation in agile software development—An industrial case study. *2016 IEEE/ACM International Conference on Software and System Processes (ICSSP),* 41–50.

TECOLOTE Inc. (2020, November). Automated cost estimating integrated tool: CO$TAT. https://www.aceit.com/aceit-suite-home/product-info/costat

United States Department of Defense Chief Information Officer. (2019, September). *DoD enterprise DevSecOps reference design*. https://dodcio.defense.gov/Portals/0/Documents/DoD%20Enterprise%20DevSecOps%20Reference%20Design%20v1.0_Public%20Release.pdf?ver=2019-09-26-115824-583

United States Department of Defense Science Board. (2018, February). *Design and acquisition of software for defense systems.* https://apps.dtic.mil/dtic/tr/fulltext/u2/1048883.pdf

United States House of Representatives. (2018, June). *Department of Defense appropriations bill, 2019*. https://slidelegend.com/queue/department-of-defense-appropriations-bill-2019-pdf-_5b6a213c097c47ac1b8b45bc.html#

Usman, M., & Britto, R. (2016). Effort estimation in co-located and globally distributed Agile software development: A comparative study. *2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA)*, 219–224.

Usman, M., Mendes, E., Weidt, R., & Britto, R. (2014). 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), 82–91. ACM, New York, NY, USA.

Usman, M., Mendes, E., & Börstler, J. (2015). 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.

Valerdi, R. (2005). *The constructive systems engineering cost model (COSYSMO)* [Doctoral dissertation, University of Southern California, Los Angeles].

Acquisition Research Program
Naval Postgraduate School
555 Dyer Road, Ingersoll Hall
Monterey, CA 93943

www.acquisitionresearch.net