A Novel Approach to Early Phase Agile Software Estimating and Sizing

Wilson Rosa and Sara Jardine

Abstract—Since the Agile Manifesto, analysts have struggled to find the appropriate software size measures for cost estimates at the earliest stages of an agile software acquisition and at the time when popular sizing measures are not available. Early program documentation such as the Mission Needs Statement, Concept of Operations, and the Release Roadmap provide three software size measures to predict total software development effort and schedule: Capability Gap, Capability, and Epic. With the cost and schedule models provided in this study, analysts can develop agile software development estimates to inform early program decisions, support Analysis of Alternatives (AoAs), and Rough Order Magnitude (ROM) estimates. The analysis presented in this study is based on data from twenty (20) agile projects implemented between 2014 to 2022 in the Department of Homeland Security and Department of Defense. Analysts can use these estimation models for agile programs following the DHS' Streamlined Software Acquisition Process or DoD's Software Acquisition Pathway.

Index Terms— Agile software process, Cost estimation, Requirements/Specification, Software acquisition, Software process, Time estimation

1 Introduction

HISTORICALLY, the United States (U.S.) Department of Homeland Security (DHS) has struggled with incorporating software development requirements into their program's cost estimate and planned schedule [8].

In 2017, the DHS Under Secretary for Management (USM) tasked the DHS Cost Analysis Division to find ways to improve cost estimates for Agile software development programs. There were two primary objectives [8]:

1. Enhance the credibility and accuracy of a software development estimate, and
2. Decrease the time required to develop the estimate.

Despite these efforts, the lack of agile software development data continues to hinder DHS' ability to implement new methods for developing credible estimates at early phase. Likewise, the Department of Defense (DoD) struggle with similar issues as it pertains to agile software development cost estimating. The problem is compounded as the government cost community continues to struggle with identifying the most appropriate software sizing measures for cost estimation that can be used throughout the acquisition cycle.

1.1 Significance of Proposed Study

This study improves the software cost estimating practice in four ways:

- Breaks new ground by incorporating three new and potential software size measures -- **Capability Gap**, **Capability**, and **Epic**. These sizing measures were chosen since they are the only artifacts that could be collected or counted at the earliest phase of an agile software acquisition lifecycle.
- Provides practical estimation models based on sizing measures available at an earlier phase and has the flexibility to be used for developing cost estimates before or after award.
- Provides a ranking of the accuracy and fit of effort and schedule estimation models using the three early phase sizing measures.
- This study uses a cross-company dataset and captures total contract effort at the release level. The choice for using total effort is driven by the fact that most agile development contracts in DHS are Firm Fixed Price (FFP) or Time & Materials (T&M), and these contracts typically report effort at the total level instead of lower-level elements (software development, systems engineering, training, etc.).

1.2 Research Questions

The following four research questions (RQ) are addressed in this study :

RQ 1: How do each of the three high-level size measures accurately predict to total development contract effort?

RQ 2: How do the three high-level size measures compare and rank as accurate predictors of total development contract effort?

RQ 3: How do each of the three high-level size measures accurately predict total software development schedule?

RQ 4: How do the three high-level size measures compare and rank as accurate predictors of total software development schedule?

2 Background

2.1 Agile in the DHS Context

In the DHS, *Agile* ([1], [3]) is the required development approach for information technology (IT) acquisition programs and projects in accordance [7] with the Federal Information Technology Acquisition Reform Act. Although there is not a specific set of DHS-approved *Agile* methodologies, program managers are encouraged to determine the most appropriate *Agile* approaches for their program ([1], [3]). Finally, DHS programs implementing *Agile* development are subject to the requirements of the DHS Acquisition Lifecycle Framework (ALF) and the Systems Engineering Life Cycle (SELC) [5].

2.2 Agile Software Acquisition Phases in the DHS

In December 2021, the DHS Office of the Undersecretary for Management established [5] the Streamlined Software Acquisition Process (SSAP) for use by software development programs. The SSAP is an Agile approach to the SELC that enables consistent engineering management and supports the effective delivery of capabilities to end users. Figure 1 shows the phases, acquisition decision **events** (**0, 1, 2A, 2B, 2C,** and **3**), **acquisition documentation**, and available software **sizing measures** of the DHS SSAP.

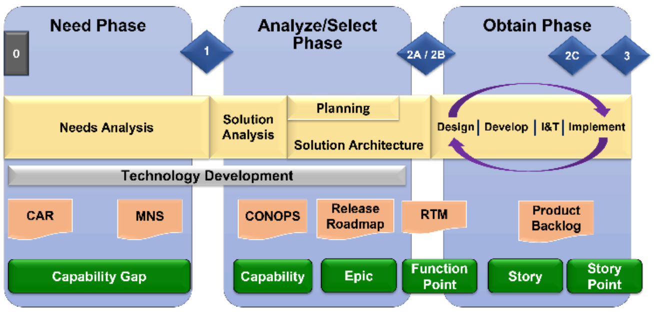

Figure 1 Streamlined Software Acquisition Process (SSAP)

A description of each acquisition phase is provided below:

A. **Needs Phase:** During this phase, a capability analysis is conducted and documented in the Capabilities Analysis Report (CAR). The CAR [4] identifies capability gaps, duplications, and potential high-level materiel or non-materiel solution approaches to mitigate identified capability gaps. When a materiel solution is identified, the Mission Needs Statement (MNS) is developed to define the specific **Capability Gaps** [4] from the CAR that will be improved by the proposed materiel solution.

B. **Analyze/Select Phase:** During this phase, Solution Analysis, Planning, and Solution Architecture tasks are conducted. During Solution Analysis, the materiel solution's **Capabilities** [4] are documented in the Concept of Operations (CONOPS). These Capabilities are in response to the **Capability Gaps** identified in the MNS. During the Planning and Solution Architecture, **Epics** [1] are identified in the Release Roadmap for developers to deliver functionality.

C. **Obtain Phase:** During this phase, Design, Develop, Integrate and Test (I&T), and Implementation are conducted for the materiel solution. Before the Design stage (ADE-2B), **Simple Function Point** [8] is measured from the Requirements Traceability Matrix (RTM). SiFP is the primary input for independent cost assessments in the DHS. After Design, **Story** and **Story Points** are quantified and continuously updated in the Product Backlog. During this time, the criteria to achieve Initial Operational Capability (ADE-2C) and Full Operational Capability (ADE-3) are generally defined.

This study focuses on high-level software sizing measures (Capability Gap, Capability, Epic) available to measure before ADE -2A that can be used to estimate effort and schedule at the early stages of an agile program. To understand the applicablity of these size measures across the software community, the next section will map these measures to mainstream agile sizing measures.

2.3 High-level Requirements Hierarchy in the DHS

Figure 2 maps the high-level size measures from mainstream agile ([13],[15],[16]) to those used in the DHS. In the context of mainstream agile, a Theme is an organization goal that drives the creation of Initiatives and is mapped to a Capability Gap in DHS. A Theme is the highest level in the agile software requirements hierarchy. An Initiative in mainstream agile is a collection of Epics that drive toward a common goal and is mapped to a Capability in DHS [13]. Epics [13] are large bodies of work that can be broken down into smaller tasks (Stories) and mean the same in mainstream agile and DHS. In summary, in the DHS context, a Theme is referred to as Capability Gap [4] and Initiative is referred to as Capability [4].

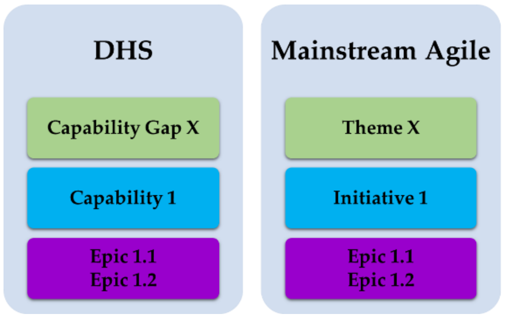

Figure 2 High-Level Requirements Hierarchy

3 Research Method

3.1 Population and Sample

This study captured agile projects categorized as *automated information systems*. The sample dataset includes twenty (20) agile projects across 14 different companies, delivered for the DHS (17) and DoD (3) from years 2014 to 2022.

3.2 Data Collection

The data collected for each agile project included actual effort, schedule, final size, and project characteristics. The data in this study (Figure 3) were extracted from official documents such as monthly contractor invoices, Mission Needs Statement (MNS), Concept of Operatoins (CONOPS), Operational Requirements Document (ORD), Release Roadmap, acquisition documents, and agile core metrics.

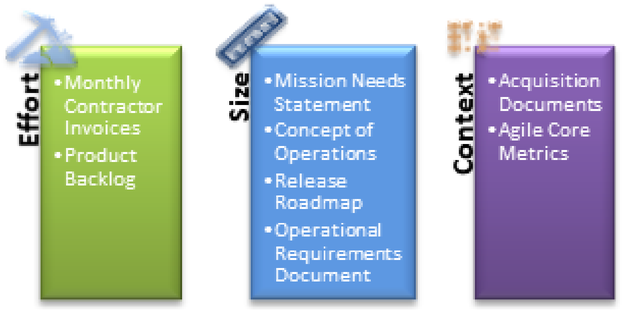

Figure 3 Data Sources

The effort data in this study captures eleven major cost elements incurred by the contractor's agile development teams in accordance with the DHS IT Work Breakdown structure (WBS). Those major WBS cost elements are also applicable to the DoD programs captured in this study and are identified in Table 1. The choice for reporting total effort (as opposed to software development alone), is driven by the fact that most agile project contracts in DHS are FFP or T&M, and therefore, generally do not break out effort by major elements as would traditional cost-plus contracts.

Table 1 Agile Project Labor Activities

ID	DHS IT WBS Element
1.i.1	Program Management
1.i.2	Systems Engineering
1.i.4.2	Software Development
1.i.4.3	Data Development & Transition
1.i.4.5	Training Development
1.i.4.6.1	Development Test & Evaluation
1.i.4.6.1	Cybersecurity Test & Evaluation
1.i.4.7	Logistics Support Development
1.i.7	System Level Integration & Test
1.i.8.6.1	Help Desk/Service Desk (Tier 3)
1.i.8.6.4	Software Maintenance

3.3 Variables in the Study

The variables chosen in this study (Table 2) represented agile measures that could be collected from available early phase documentation sources in DHS. Of note, a categorical variable, characterizing whether a project scope was an enhancement or full development, was also evaluated in the regression analysis. A full development versus enhancement scope is important in estimating schedule since the scope of an enhancement is typically less than a one-year effort while a full development effort is typically more than a one-year. The description of each variable evaluated in the effort and schedule models are defined in Table 2 below.

Table 2 DHS Agile Project Work Breakdown Structure

Variable	Type	Definition
Effort (E)	Dependent	Actual hours associated to the labor activities listed in Table 2
Schedule (S)	Dependent	Actual development time (in months) associated to all activities listed In Table 2. Reported at the release level.
Capability Gap (CAP_GAP)	Independent	The difference between the necessary capabilities and the capabilities which are currently possessed/planned. [4]
Capability (CAP)	Independent	The means to accomplish a mission, function, or objective. A specified course of action supporting users and departmental goals/missions. [4]
EPIC (EPIC)	Independent	Body of work that can be broken down into specific tasks based on the needs of end-users.
Scope (D1)	Categorical	Indicate whether the scope of the project is an enhancement or full development effort.

3.4 Data Normalization

The data normalization process included counting Capability Gaps, Capabilities, and Epics from early requirements documents. The details of the normalization process for each of these measures will be discussed in this section.

3.4.1 Counting Capability Gaps

The team used a repeatable method to count the Capability Gaps for each project in the dataset. Below is an outline of the steps employed to determine the total number of Capability Gaps from an agile project's MNS. Table 3 provides an example of the Capability Gaps output from the MNS that the team used during the counting process.

Step 1: In the MNS, find table titled "Capability Needs and Gaps" under section titled "Mission(s) and Capabilities."

Step 2: In the column titled, *Capability Gap*, count the rows containing a narrative statement. The example in Table 3 below results in a total count of **3 Capability Gaps**.

Table 3 Examples of Capability Gap in the MNS

ID	Capability Gap
1	Unable to provide notifications that are automated, user-relevant, and on-going regarding data changes and events, resulting in a failure to provide timely, accurate, and actionable information.
2	No ability to attach Documents to Policies or Claims
3	Very limited reporting capabilities and no data modeling capabilities.

3.4.2 Counting Capabilities

The team used a repeatable method to count the Capabilities for each project in the dataset. Below is an outline of the steps to determine the total number of Capabilities from an agile project's CONOPS. Table 4 provides an example of the Capabilities output from the CONOPS that the team used during the counting process.

Step 1: In the CONOPS, find table "Capability Requirements" in the section titled "Required Mission(s) and Needs."

Step 2: In the column titled, *Capability*, count the rows containing a narrative statement. The example in Table 4 results in a total count of **3 Capabilities**.

Table 4 Examples of Capability in the CONOPS

ID	Capability	Description
1	[System X] Near Real-Time Financial Data Processing	Validation of financial data being sent from numerous sources.
2	[System X] Reporting Functionality	Extensive reporting features including standard predefined reports, ad-hoc reporting, and export of data into multiple file formats
3	Financial Statement and Report Generation	Capability to generate monthly financial statements and other financial reports

3.4.3 Counting Epics

The team used a repeatable method to count epics for each project in the dataset. The Epics of each project are documented in the Release Roadmap as well as in the Product Backlog. Since the Epics in the Backlog are traceable to the Release Roadmap, the team counted the epics directly from the Product Backlog. Table 5 below provides an example of the backlog the team used during the counting process.

Step 1: Determine 100% Complete Issues. Find the column titled, *Issue Status*, and filter by rows marked as *Done*. By filtering by the issues that were 100% compete, issues that had status of being *in progress* or *deferred*, were omitted.

Step 2: Calculate total unique epics. In the column titled, *Epic Link*, count the total unique Epics. The example in Table 5 results in a total count of **3 Epics**.

Table 5 Examples of Epic in Product Backlog

Issue	Status	Description	Issue Type	Epic Link
0001	Done	As a <user> I need to manually initiate the<outcome>	Story	Plan Subapplication
0002	Done	As a <user> I need to view the <outcome>	Story	Plan Subapplication
0003	Done	As a <system administrator> I need to manage certificates so I can <outcome>	Story	Plan Subapplication
0004	Done	As a <SCRUM Master> I want to review list of Epic(s) so I can <outcome>	Task	Configurable Permissions
0005	Done	The following field name is spelled incorrectly: <Name>	Bug	Address Verification
0006	Done	As a <system user> I need to have the selected tools installed…	Other	Address Verification

3.4.4 Model Selection and Validation

Regression analysis was performed on the full dataset (20) using the Automated Cost Estimating Integrated Tools (ACEIT) Cost Analysis Statistical Package (CO$TAT) [11]. Each regression model was selected based on evaluating the following goodness-of-fit measures (Table 6). These statistics were examined to determine the reliability, accuracy, and fit of each model.

Table 6 Goodness-of-Fit Measures

Metric	Description
R^2	Coefficient of determination is the percentage of total variation in the response variable explained by the model. The higher the R^2, the more variability is explained by the model.
R^2 (adj)	Adjusted R^2 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. The higher the R^2 (adj), the more variability is explained by the model.
R^2 (pred)	Predicted R^2 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. The higher the R^2 (pred), the more variability is explained by the model.
P-value	The probability value of obtaining results at least as extreme as the observed results of a statistical hypothesis test, assuming the null hypothesis is correct. The lower the p-value for each independent variable, the more statistically significant those variables are in predicting the dependent variable.
SEE	Standard Error of the Estimate (SEE) is the difference between observed and estimated effort. SEE is to linear models as standard deviation is to sample means. The lower the SEE, the better the regression model fits to the dataset.
F-test	F-test is the square of the equivalent t-test; the larger

	the value, the smaller the probability that difference could occur by chance.
MMRE	Mean Magnitude of Relative Error (MMRE) is an indicator of a model's accuracy. The lower the MMRE, the higher the accuracy of the model.

4 Data Analysis

The entire dataset in this study represents projects categorized as Automated Information Systems (AIS) delivered from 2014 to 2022. Sixteen (16) of the twenty (20) projects are hosted on the cloud, while the remaining four are hosted on-premises. Of the 16 cloud-hosted, 15 used Amazon Web Services (AWS). All DHS projects were from Surface Fixed operating environments, while the DoD projects were from Sea System environments. Figure 4 shows project counts by agency and operating environment.

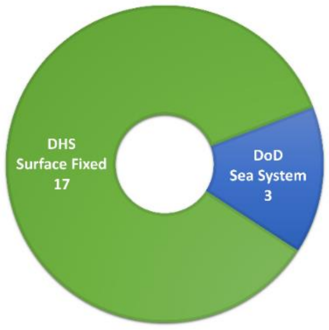

Figure 4 Operating Environment

The project counts by contract type is shown in Figure 5. Majority of the agile development projects utilized Firm-Fixed Price (FFP) and Time and Materials (T&M) contract types.

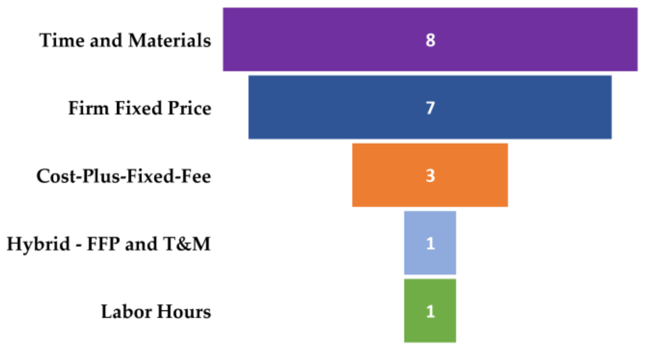

Figure 5 Contract Strategy

The team approach of each agile project was documented to better understand the development practices of the agile teams. Of the twenty (20) agile projects, half (10) followed SecDevOps practices while the other half (10) used the DevOnly or DevOps team approach. Figure 6 presents the agile process and team approach for the dataset.

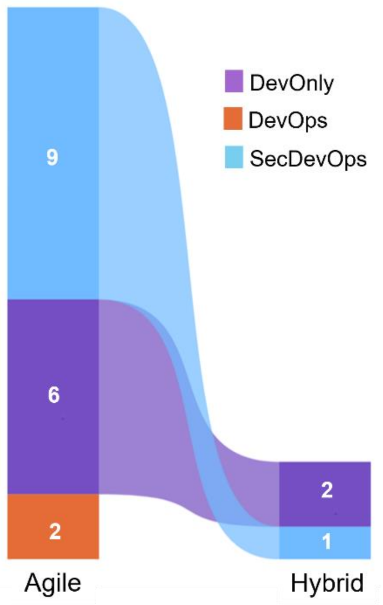

Figure 6 Agile Team Approach

The distribution of the Agile framework is presented in Figure 7. The Hybrid Agile projects all used the Scaled Agile Framework. The majority (14) used Scrum framework.

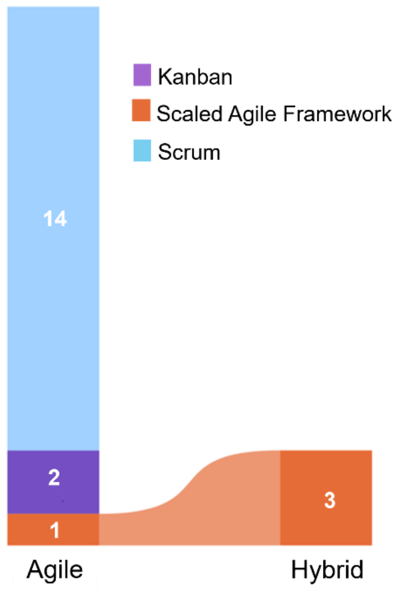

Figure 7 Agile Framework

The two dependent variables used in this study are *Effort* and *Schedule*. Effort is measured in terms of total contract hours whereas schedule is measured in terms of total months. Figure 8 displays a histogram of the descriptive statistics for effort hours. The average total effort hours for the dataset were 122,889 hours.

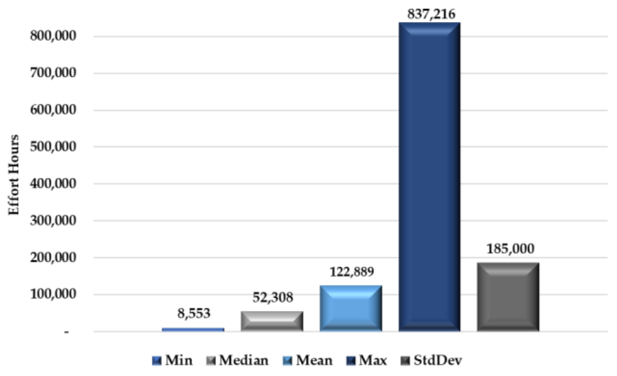

Figure 8 Effort Distribution (Hours)

Figure 9 displays a histogram of the descriptive statistics for schedule months. The average total months for the dataset was 19 months.

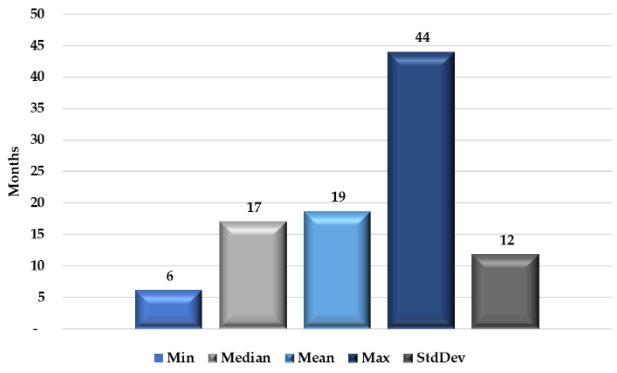

Figure 9 Schedule Distribution (Months)

The descriptive statistics for the independent variables in the agile dataset are shown in Table 7. For each variable, the minimum, median, maximum, and standard deviation (StdDev) values are shown.

Table 7 Independent Variable Descriptive Statistics

Size Measure	Min	Median	Max	StdDev
Capability Gap	1	5	26	6
Capability	4	10	50	11
Epic	13	35	406	89

The effort and schedule benchmarks are shown in Table 8 and Table 9 respectively and can be used to develop quick turnaround estimates. The effort benchmarks include (1) Hours per Capability Gap, (2) Hours per Capability, and (3) Hours per Epic. The schedule benchmarks include (1) Months per Capability Gap; (2) Months per Capability, and (3) Months per Epic. For each benchmark, the first quartile (Q1), **median (Q2)**, third quartile (Q3), and standard deviation (StdDev) values are shown.

Table 8 Effort Productivity Benchmark

Productivity Measure	Q1	Q2	Q3	StdDev
Hours/CAP_GAP	22,210	26,310	30,443	14,890
Hours/CAP	4,490	5,696	9,112	3,925
Hours/Epic	1,180	1,789	2,048	792

CAP_GAP = Capability Gap; CAP = Capability

Table 9 Schedule Productivity Benchmark

Productivity Measure	Q1	Q2	Q3	StdDev
Months/CAP_GAP	4.7	4.9	9.7	3.6
Months/CAP	1.0	1.6	2.0	0.8
Months / Epic	0.3	0.4	0.5	0.2

CAP_GAP = Capability Gap; CAP = Capability

5 RESULTS

This section provides the results organized by research question. For each research question, the key findings are also discussed. The resulting effort and schedule models are applicable to DHS and DoD agile software project ranging approximately 1 to 26 Capability Gaps, 4 to 40 Capabilities, 13 to 406 Epics, and a peak staff between 9 to 200 Full-Time Employee (FTE).

5.1 Results for Research Question 1

RQ 1: How do each of the three high-level size measures accurately predict to total development contract effort?

5.1.1 Effort Model 1

Equation (1) predicts software development effort as a function of Capability Gap.

$$Effort = 26368 \; x \; CAP\_GAP^{0.9518} \qquad (1)$$

Where,

Effort	=	total final development hours
CAP_GAP	=	Capability Gaps obtained from MNS

Table *10* provides the regression analysis report of the coefficient statistics, goodness-of-fit statistics, and analysis of variance for equation (1). The resulting equation is statistically significant and reveals that Capability Gaps is a good predictor of effort of agile software projects at early stage.

Table 10 Regression Analysis Results for Equation (1)

Coefficient Statistics Summary

Term	Coef	T-Statistic	P-value
Intercept	10.18	41.81	0.00
Cap_Gap	0.95	6.24	0.00

Goodness-of-Fit Statistics

SE	R^2	$R^2_{(adj)}$	$R^2_{(pred)}$	MMRE
0.57	74.98%	73.05%	66.09%	46.28%

Analysis of Variance

Source	DF	Sum of Sq.	Mean Sq.	F-stat
Regression	1	12.44	12.44	38.95
Residual	13	4.15	0.32	
Total	14	16.59		

Figure 10 shows the normal probability plot for Equation (1). The residuals are close to the straight line. This suggests that loglinear regression is valid for modeling effort vs Capability Gaps.

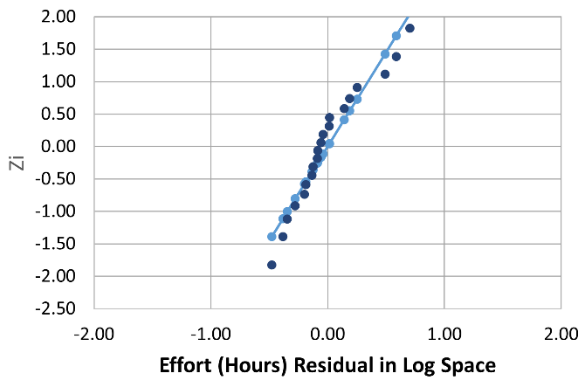

Figure 10 Normal Probability Plot for Equation (1)

5.1.2 Effort Model 2

Equation (2) predicts software development effort as a function of Capability.

$$\textbf{Effort} = \mathbf{1170}\ \boldsymbol{x}\ \boldsymbol{CAP}^{1.712} \qquad (2)$$

Where,

Effort	=	total final development hours
CAP	=	Total number of Capabilities obtained from CONOPS or ORD

Table 11 provides the regression analysis report for equation (2). The high t-statistic and low p-value suggest that Capability is strongly correlated to total effort. The small difference (1%) between adjusted and predicted R² suggest the model predicts new observations as well as it fits the existing data. An adjusted R² greater than 92% and MMRE lower than 25% signify that Capabilities accurately predicts the total effort.

Table 11 Regression Analysis Results for Equation (2)

Coefficient Statistics Summary

Term	Coef	T-Statistic	P-value
Intercept	7.06	25.47	0.00
CAP	1.71	14.90	0.00

Goodness-of-Fit Statistics

SE	R^2	$R^2_{(adj)}$	$R^2_{(pred)}$	MMRE
0.33	92.50%	92.08%	91.16%	23.46%

Analysis of Variance

Source	DF	Sum of Sq.	Mean Sq.	F-stat
Regression	1	23.49	23.49	221.92
Residual	18	1.91	0.22	
Total	19	25.40		

Figure 11 shows normal probability plot for Equation (2). The residuals approximate a straight line, suggesting that loglinear regression is valid for modeling effort vs Capabilities.

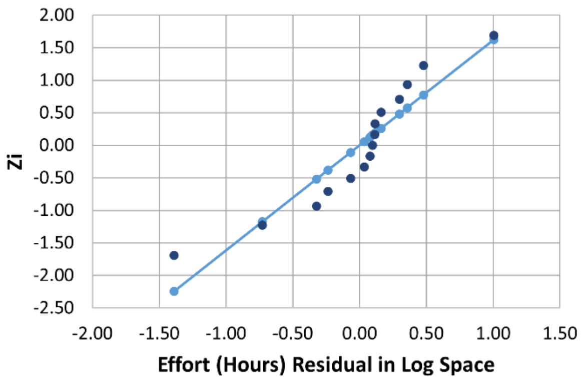

Figure 11 Normal Probability Plot for Equation (2)

5.1.3 Effort Model 3

Equation (3) predicts software development effort as a function of Epics.

$$\textbf{Effort} = \mathbf{710.6}\ \boldsymbol{x}\ \boldsymbol{EPIC}^{1.215} \qquad (3)$$

Where,

Effort	=	total final development hours
EPIC	=	Epics obtained from Release Roadmap

Table 12 provides the regression analysis report of the coefficient statistics, goodness-of-fit statistics, and analysis of variance for equation (3). The high t-statistic and low p-value suggest that Epics is strongly correlated to total effort. The slight difference (2%) between adjusted and predicted R² suggest the model predicts new observations as well as it fits the existing data. An adjusted R² greater than 80% and an MMRE of 44% indicates that Epics is a good predictor of the total effort.

Table 12 Regression Analysis Results for Equation (3)

Coefficient Statistics Summary

Term	Coef	T-Statistic	P-value
Intercept	6.57	12.83	0.00
EPIC	1.22	8.99	0.00

Goodness-of-Fit Statistics

SE	R^2	$R^2_{(adj)}$	$R^2_{(pred)}$	MMRE
0.51	81.77%	80.76%	78.23%	43.76%

Analysis of Variance

Source	DF	Sum of Sq.	Mean Sq.	F-stat

Regression	1	20.77	20.77	80.74
Residual	18	4.63	0.26	
Total	19	25.40		

Figure 12 shows the normal probability plot for Equation (3). The residuals approximate a straight line. This suggests that loglinear regression is valid for modeling effort vs Epics.

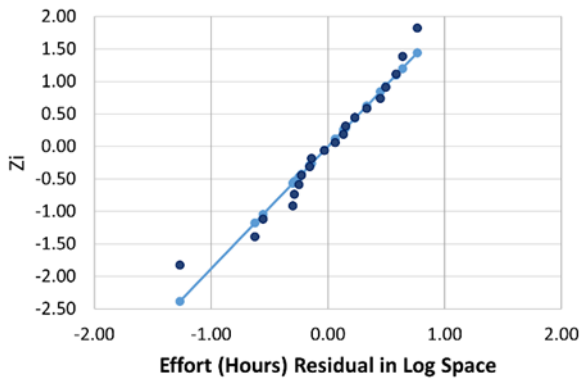

Figure 12 Normal Probability Plot for Equation (3)

5.2 Results for Research Question 2

RQ 2: How do the three high-level size measures compare and rank as accurate predictors of total contract software development effort?

Table 13 compares the statistical significance of the resulting effort equations using the three different software size predictors. The comparative results with a synopsis of the suggested ranking order of the models is summarized below. The models were compared and ranked according to the resulting MMRE, from lowest to highest. The decision to use the MMRE is substantiated by literature ([14], [6], [9]). A low MMRE is indicative of high model predictive power whereas a high MMRE indicates low predictive power.

Table 13 Effort Model Comparison

Equation	Independent Variable	$R^2_{(adj)}$	$R^2_{(pred)}$	MMRE	Rank
1	Capability Gap	73.1%	66.1%	43.8%	3
2	Capability	92.1%	91.2%	23.5%	1
3	Epic	80.8%	78.2%	42.5%	2

All three effort models were determined to be statistically significant in accordance with regression goodness-of-fit statistics (Table 6). The most accurate sizing measure predicters to schedule development effort in ranking order from highest to lowest are Capability, Epic, and Capability Gap. Next, we will evaluate the resulting schedule models.

5.3 Results for Research Question 3

RQ 3: How do each of the three high-level size measures accurately predict total software development schedule?

5.3.1 Schedule Model 1

Equation (4) predicts schedule for agile software development projects as a function of Capability Gaps.

$$\textbf{Schedule} = \mathbf{12.13}\ \boldsymbol{x}\ \boldsymbol{CAP\_GAP}^{\mathbf{0.4272}} \qquad (\mathbf{4})$$

Where,

Schedule	=	total final development months
CAP_GAP	=	Capability Gaps obtained from MNS

Table *14* provides the regression analysis report for equation (4). The resulting equation is statistically significant and demonstrates that Capability Gap is a good predicter of schedule for agile software projects.

Table 14 Regression Analysis Results for Equation (4)

Coefficient Statistics Summary

Term	Coef	T-Statistic	P-value
Intercept	2.50	26.42	0.00
CAP_GAP	0.43	7.22	0.00

Goodness-of-Fit Statistics

SE	R^2	$R^2_{(adj)}$	$R^2_{(pred)}$	MMRE
0.22	80.03%	78.49%	68.80%	16.35%

Analysis of Variance

Source	DF	Sum of Sq.	Mean Sq.	F-stat
Regression	1	2.51	2.51	52.10
Residual	13	0.63	0.05	
Total	14	3.13		

Figure 13 shows the normal probability plot for Equation (4). The residuals approximate a straight line, suggesting that loglinear regression is valid for modeling schedule vs Capability Gaps.

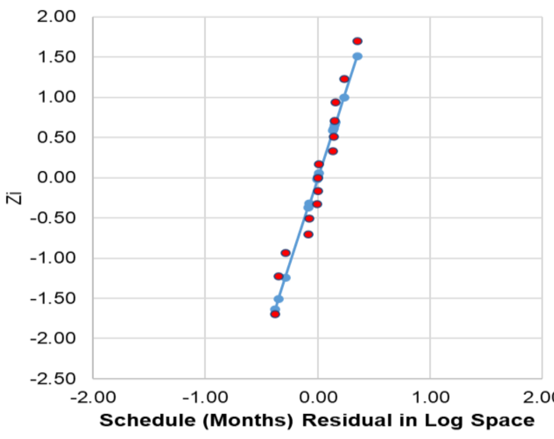

Figure 13 Normal Probability Plot for Equation (4)

5.3.2 Schedule Model 2

Equation (5) predicts schedule for agile software development projects as a function of number of Capabilities and a dummy variable associated with scope type.

$$\textbf{Schedule} = 2.45\ x\ CAP^{0.507}\ x\ 2.4^{D1} \qquad (5)$$

Where,

Schedule	=	total final development months
CAP	=	Capabilities from CONOPS or ORD
D1	=	where full development was assigned a value of 1 and enhancement a value of 0

Table 15 provides the regression analysis report for equation (5). The resulting equation is statistically significant and demonstrates that Capabilities when treated along with Scope are good at predicting schedule.

Table 15 Regression Analysis Results for Equation (5)

Coefficient Statistics Summary

Term	Coef	T-Statistic	P-value
Intercept	0.90	3.48	0.00
CAP	0.51	4.16	0.00
D1	0.88	4.91	0.00

Goodness-of-Fit Statistics

SE	R^2	$R^2_{(adj)}$	$R^2_{(pred)}$	MMRE
0.30	82.43%	80.37%	75.95%	24.45%

Analysis of Variance

Source	DF	Sum of Sq.	Mean Sq.	F-stat
Regression	2	6.27	3.13	44.60
Residual	12	0.84	0.07	
Total	14	7.11		

Figure 14 shows the normal probability plot for Equation (5). The residuals approximate a straight line. This suggests that loglinear regression is valid for modeling schedule vs Capabilities.

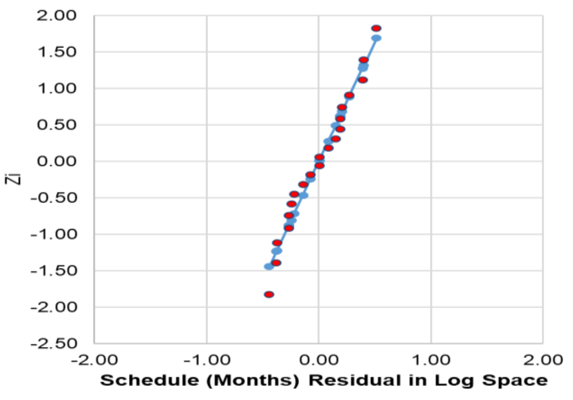

Figure 14 Normal Probability Plot for Equation (5)

5.3.3 Schedule Model 3

Equation (6) predicts schedule for agile software development projects as a function of Epics and a dummy variable associated with scope type.

$$\textbf{Schedule} = 2.069\ x\ EPIC^{0.3634} x 2.43^{D1} \qquad (6)$$

Where,

Schedule	=	total final development months
EPIC	=	Epics obtained from Release Roadmap
D1	=	dummy variable associated with scope where full development was assigned a value of 1 and enhancement a value of 0

Table 16 provides the regression analysis report for equation (6). The resulting equation is statistically significant and demonstrates that Epics when treated along with scope, as categorical variable, are effective in estimating the schedule of agile software projects.

Table 16 Regression Analysis Results for Equation (6)

Coefficient Statistics Summary

Term	Coef	T-Statistic	P-value
Intercept	0.73	2.25	0.04
EPIC	0.36	3.72	0.00
D1	0.89	4.69	0.00

Goodness-of-Fit Statistics

SE	R^2	$R^2_{(adj)}$	$R^2_{(pred)}$	MMRE
0.32	80.47%	78.18%	72.84%	24.29%

Analysis of Variance

Source	DF	Sum of Sq.	Mean Sq.	F-stat
Regression	2	7.11	3.56	35.03
Residual	17	1.73	0.10	
Total	19	8.84		

Figure 15 shows the normal probability plot for Equation (6). The residuals approximate a straight line suggesting that loglinear regression is valid for schedule vs Epics.

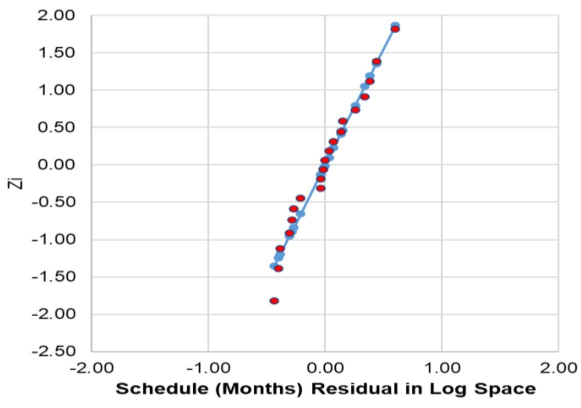

Figure 15 Normal Probability Plot for Equation (6)

5.4 Results for Research Question 4

RQ 4: How do the three high-level size measures compare and rank as accurate predictors of total software development schedule?

Table 17 compares the statistical significance of the schedule model equations using three different size predictors. The comparative results with a synopsis of the suggested ranking order of the models is summarized below.

Table 17 Schedule Model Comparison

Equation	Independent Variable	$R^2_{(adj)}$	$R^2_{(pred)}$	MMRE	Rank
4	Capability Gap	78.5%	68.8%	16.4%	2
5	Capability	80.4%	80.0%	24.5%	1
6	Epic	78.2%	72.8%	24.3%	3

All three schedule models were determined to be statistically significant in accordance with regression goodness-of-fit statistics (Table 6). **Capabilities** is the strongest predictor to development schedule followed by **Capability Gap** and **Epic**.

5.5 Discussion of Results

The results of our study suggest that **Capability Gap** is a good predicter of total software development effort, as indicated by the adjusted R^2 (73%) and statistical significance. However, the resulting MMRE (44%) is an indication of a moderate predictive power. This could be attributed to the fact that the "Capability Gaps" are not written in a standard form, may contain duplicates, or include gaps already captured in other projects. Despite showing a moderate predictive power, practitioners should use Capability Gap (or "Theme") as an effort predicter, especially since it is typically the only software sizing measure available at the inception phase and at the time when other high-level requirements (Initiatives, Epics, Stories) have not been defined yet.

Our results reveal that **Capability** is an effective predicter of total software development effort at the Analyze/Select Phase, as evidenced by the high accuracy and high model fit. This finding is a breakthrough since past research has not examined Capability (or Initiative) as a predicter to software development at an early program phase. In fact, Capability is just as good as a predictor to software development as the more popular Function Point and Story Point sizing measures, which are known later in the lifecycle. The higher predictive power of Capability could have been attributed to the fact that our dataset contained projects of the same application domain, same scope, and normalized by the same team. One subject that remains to be explored is how to adjust the Capability counts based on the relative T-Shirt Size, Complexity, and Volatility.

Our findings suggest that **Epic** is a good predicter of total software development effort, as indicated by the adjusted R^2 (81%) and strong goodness-of-fit statistics. This finding is also breakthrough since past research has not examined Epic as a predicter to software development at an early program phase. Like Capabilities, Epic is just as good as a predictor to software development as the more popular Function Point and Story Point sizing measures, which are known later in the lifecycle. Practitioners can use Epics to predict total effort during proposal evaluation or implementation phase, as it can be measured from the Release Roadmap or from a Product Backlog after contract award.

When considering the following constraints, this study revealed that effective effort estimation models can be built for early phase decision-making. These constraints include:

1. Effort data reported at the total contract level
2. Projects are of the same application domain
3. Effort and size collected at the release level
4. Projects collected from cross-companies
5. Effort and size data analyzed at the release level
6. Size measures counted and validated by same team

5.6 Threats to Validity

Possible threats to the validity of the resulting effort and schedule models include *internal*, *external*, and *constructive*. A discussion of each threat is summarized below.

Threats to *internal* validity include the dataset timeframe from 2014 to 2022, which raises potential issues where earlier projects (2014-2018) were developed using agile processes tailored to fit the developer’s need. It is likely that agile processes have evolved during the 8-year timeframe. The scope of this study covers programs that were classified as *Agile*, perhaps loosely, and a focus on only a single development process. This poses a limitation to programs using a different software development process such as waterfall and may produce different results.

Threats to *external* validity include the availability and quality of early program acquisition documentation such as the MNS, ORD, and CONOPS. To produce a quality count of Capability Gaps from a MNS or Capabilities from an ORD or CONOPS, it is important that the program documentation be in a mature or final state with clearly written gaps and capabilities. If the documentation is provided in a draft, interim state, or if written unclearly then an inaccurate count of Capability Gaps or Capabilities may result. The counting process would ultimately result in a gross estimate of software development effort and schedule when using the model equations.

The models presented in this paper revealed to be effective in estimating total development effort and duration for agile projects reported at the release level for DHS and DoD. However, we cannot generalize beyond this group for several reasons. First, majority of the projects were developed using Scrum and SAFe. Second, the total effort includes other activities above and beyond those captured in mainstream software cost estimating models. Examples of elements captured in the total effort for our agile dataset included program management, systems engineering, training, security, testing, and operations.

Threats to *constructive* validity include the limited number of datapoints in the sample size of 20 agile projects. With a sample size this small, there is a threat to the statistical conclusions as they may be subject to overfitting and does not allow for detecting effects with greater power. To address this threat, a larger sample size is needed for definitive hypothesis testing and statistical conclusions of the regression models.

6 Conclusion

Capability Gap (aka Theme), Capability (aka Initiative), and Epic are high-level software size measures that can be found in early agile program phase documentation. These measures may be used to estimate total software development effort and schedule of an agile project during early program phases of DHS' Streamlined Software Acquisition Process or DoD's Software Acquisition Pathway. The results of our study support the concept that these three measures known to us before the agile project started, are just as good of predicters to software development as the measures (i.e., Story Point, Story) known to us after contract award.

Capability Gap, Capability, and Epic have low volatility throughout the agile lifecycle. As a result, these size measures could prove to be more useful to Practitioners when predicting agile program software development effort or schedule at an early phase and throughout the lifecycle. On the other hand, Functional Requirements and Stories are continuously changing and likley underestimated at the early lifecycle phases, which may prove to be less useful to analysts when estimating agile program software development at an early phase.

Acknowledgement

The contents of this paper reflect the views of the authors and are not necessarily endorsed by the Department of Homeland Security.

References

[1] Chief Technology Officer Directorate, Office of the Chief Technology Officer, "Department of Homeland Security Agile Guidebook," 31 Jul. 2021.

[2] Department of Defense, "DoD Instruction 5000.87, Operation of the Software Acquisition Pathway," Oct. 2020. . Available at: https:// https://www.esd.whs.mil/DD/

[3] Department of Homeland Security, "*Agile Development and delivery for Information Technology*," Instruction 102-01-004, rev 02, 19 Feb. 2020. Available at: https://www.dhs.gov/sites/default/files/publications/agile-development-and-delivery-for-information-technology.pdf

[4] Joint Requirements Council, Department of Homeland Security, "DHS Instruction Manual 107-01-001-01, revision 01, Department of Homeland Security Manual for the Operation of the Joint Integration and Management System," 20 Sep. 2018

[5] DHS Office of Program Accountability & Risk Management, "Systems Engineering Life Cycle Guidebook," May 2021

[6] M. Fernández-Diego, E. R. Méndez, F. González-Ladrón-De-Guevara, S. Abrahão and E. Insfran, "An Update on Effort Estimation in Agile Software Development: A Systematic Literature Review," in IEEE Access, vol. 8, pp. 166768-166800, 2020, doi: 10.1109/ACCESS.2020.3021664.

[7] V. Kundra, U.S. Chief Information Officer, "25 Point Implementation Plan to Reform Federal Information Technology Management," Office of Management and Budget, The White House, 9 Dec. 2010. Available at:https://www.dhs.gov/sites/default/files/publications/digital-strategy/25-point-implementation-plan-to-reform-federal-it.pdf

[8] K. Mann and R. Hoang, "Functional Sizing of Agile Programs at U.S. Department of Homeland Security," MetricViews, Issue 1, IFPUG, Jul 2021. Available at: https://www.ifpug.org/wp-content/uploads/2021/07/IFPUG_July_MetricViews_2021.pdf

[9] W. Rosa, B. K. Clark, R. Madachy and B. Boehm, "Empirical Effort and Schedule Estimation Models for Agile Processes in the US DoD," in IEEE Transactions on Software Engineering, doi: 10.1109/TSE.2021.3080666.

[10] W. Rosa, R. Madachy, B. Clark and B. Boehm, "Early Phase Cost Models for Agile Software Processes in the US DoD," 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Toronto, ON, 2017, pp. 30-37. doi:10.1109/ESEM.2017.10, available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8170082&isnumber=8169971

[11] TECOLOTE Inc. "Automated Cost Estimating Integrated Tool: CO$TAT," November 2020. Available at: https://www.aceit.com/aceit-suite-home/product-info/costat

[12] 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.

[13] M. Rehkopf, "Agile epics: definition, examples, and templates," in Atlassian.com. Available at: https://www.atlassian.com/agile/project-management/epics

[14] P. Abrahamsson, I. Fronza, R. Moser, J. Vlasenko and W. Pedrycz, "Predicting Development Effort from User Stories," 2011 International Symposium on Empirical Software Engineering and Measurement, 2011, pp. 400-403, doi: 10.1109/ESEM.2011.58

[15] M. Choetkiertikul, H. K. Dam, T. Tran, A. Ghose and J. Grundy, "Predicting Delivery Capability in Iterative Software Development," in IEEE Transactions on Software Engineering, vol. 44, no. 6, pp. 551-573, 1 June 2018, doi: 10.1109/TSE.2017.2693989.

[16] Tsilionis and Y. Wautelet, "From Service-Orientation to Agile Development by Conceptually Linking Business IT Services and User Stories: A Meta-Model and a Process Fragment," 2021 IEEE 23rd Conference on Business Informatics (CBI), 2021, pp. 153-162, doi: 10.1109/CBI52690.2021.10066.

[17] Dantas, Emanuel & Perkusich, Mirko & Dilorenzo, Ednaldo & Santos, Danilo & Almeida, Hyggo & Perkusich, Angelo. (2018). Effort Estimation in Agile Software Development: An Updated Review. International Journal of Software Engineering and Knowledge Engineering. 28. 1811-1831.

10.1142/S0218194018400302.

Wilson Rosa. Dr. Wilson Rosa serves as Assistant Director at the Department of Homeland Security Cost Analysis Division. Dr. Rosa has won 6 Best Paper Awards at premier conferences including the IEEE/ACM International Symposium on Empirical Software Engineering and Measurement (1), International Conference on Software Process and Product Measurement (1), and ICEAA Professional Development & Training Workshop (4). He earned a PhD and M.S. in Engineering Management from the George Washington University and a BS in Mechanical Engineering from the University of Puerto Rico. Dr. Rosa completed the Senior Managers in Government Executive Leadership Program at the Harvard Kennedy School.

Sara Jardine. Ms Jardine is a Program Manager and Expert Cost Analyst for Galorath with over 18 years of financial management experience. She has served a broad variety of federal agencies including the Army, Navy, OUSD AT&L, DAU, Veterans Affairs, and the Department of Homeland Security. Ms Jardine earned a MS in Project Management from George Washington University and a BS in Mathematics from the University of Michigan.