Are Agile/DevOps Programs Doing Enough Systems Engineering?

Anandi Hira

Abstract: Agile and DevOps methodologies offer efficient processes to deliver high quality products and deploy them to the users quickly. Many commercial organizations have reported large savings in cost and increased productivity from implementing Agile and DevOps methodologies. MITRE completed a qualitative study of the cost impacts as a result of applying Agile methodologies and expected the Systems Engineering, Integration and Test, and Program Management (SEITPM) costs would either remain the same or slightly increase for Agile programs compared to Waterfall programs (Manring, 2016). However, this paper later demonstrates that data from Space Ground systems suggest that the SEITPM costs (as an entity) are approximately 30% lower for Agile/DevOps programs compared to Waterfall programs. In this research study, I analyze whether the difference in SEITPM costs between Agile/DevOps and Waterfall programs is statistically significant by comparing the means and evaluating the statistical significance of including a categorical variable in a regression. The results indicate that the decrease in SEITPM costs for Agile/DevOps programs is statistically significant. Reduced systems engineering could potentially lead to troubles while implementing the architecture/design or in the product quality of the completed system. Some examples of possible troubles are missing requirements, interface, and integration issues with other software and/or hardware modules/components, latent defects in the code, and high defect rates. To understand whether the reduced SEITPM costs has any adverse effects, I also conduct a survey with major industry prime contractors to determine if their observations reflect Space Ground systems data, what caused the reduction in SEITPM costs, and if they noticed any positive or negative changes in product quality as an effect. In general, organizations have experienced changes in SEITPM activities but have not experienced adverse effects in product quality as a result. Fortunately, Agile and DevOps methodologies provide a way to reduce costs without negative effects on the product's quality.

The main tenets of the Agile methodology include: incrementally gathering requirements, designing the system, developing and testing the code, demonstrating to users to get feedback, and incorporating changes to the requirements and working software. The DevOps methodology encourages faster software development and release to users by putting the development and operations related activities in parallel with each other and automating as much of the process as possible. Traditionally, software was built with sequential steps, using what is called the Waterfall model: first, the requirements were gathered, then the system was designed, after which the developers implemented the system, testers then tested it, and the system was delivered to users and customers upon completion. Following the Agile and DevOps methodologies allow the development team to provide working software quickly by continually demonstrating working features, as well as get guidance on how much to do or when to stop if schedule and budget constraints are reached. Theoretically, the biggest savings were expected in software development and sustainment efforts. MITRE presented the expected cost impacts of applying Agile methodologies, which states that in the best-case scenario, some savings are

expected in software development effort and significant savings expected in sustainment effort (see Figure 1) (Manring, 2016).

Life Cycle Cost Element	Cost Impact Range	
	Best Case	Worst Case
Program Management/System Engineering	=	+
Software Development	-	=
Integration and Test	=	+
Fielding/Deployment	=	++
Training	+	++
Sustainment	--	-

++ significance increase, + increase, = no impact, - decrease, -- significant decrease

Figure 1. Recreation of MITRE's image demonstrating cost impacts of Agile methodology on various Cost Elements (Manring, 2016)

SEITPM is an abbreviation that Department of Defense (DoD) programs use to signify the effort and costs spent in Systems Engineering (requirements gathering, architecting and designing of the program), Integration and Test, and Program Management. In the Waterfall software development lifecycle model, the steps of developing a software project (requirements gathering, architecting and designing, coding, testing, and deploying) are followed sequentially. Due to this, there is typically a high level of SEITPM effort and costs that occur at the beginning of a program (primarily due to systems engineering and program management), which quickly drops and levels until the end of the program (for program management), ending in an increase for integration and testing efforts. For Agile/DevOps programs, on the other hand, these SEITPM-type activities (as well software development) are expected to occur at a more constant rate throughout the software development lifecycle. See Figure 2 to visually see the difference of how SEITPM costs are expected to behave differently through a software development lifecycle for Waterfall and Agile/DevOps programs/projects.

Traditionally, different teams were responsible for Systems Engineering, Program Management, and Integration and Test activities. These labor categories were typically considered to be separate from the development activities, and therefore, tracked separately from the development activities. The Agile and DevOps methodologies, however, increase the speed at which requirements can change and those changes can be made in the resulting code by tightly knitting all the activities with the software development efforts (Seaver, 2018).

The definitions of the Waterfall, Agile, and DevOps lifecycle models describe how SEITPM costs theoretically are distributed across the lifecycle. The MITRE study (see Figure 1) suggests that the total SEITPM costs will be the same or higher for Agile programs, but that hypothesis is not based on an empirical analysis (Manring, 2016). This research study will determine whether total SEITPM costs differ between Agile/DevOps and Waterfall programs as the MITRE study suggests. Additionally, I survey several Agile/DevOps teams in industry to understand whether they noticed a change in the

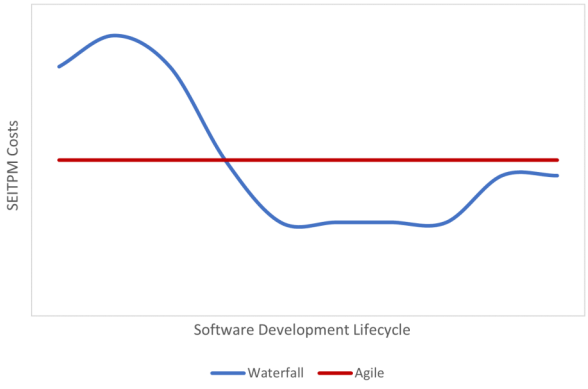

Figure 2. Visual representation of how Systems Engineering, Program Management, and Integration and Test (SEITPM) costs behave through a software development lifecycle for Waterfall and Agile/DevOps programs. This graph is created to visually depict how the costs theoretically differ and is not based on real data.

SEITPM effort/costs, as well as the causes and effects of the changes they observe within the development environments. After a brief introduction to the different software development lifecycle models (Waterfall, Agile, and DevOps), this paper has 2 parts:

1. Empirical comparison of SEITPM costs between Agile/DevOps and Waterfall programs
2. Completed surveys and discussions with Agile/DevOps teams in industry.

Software Development Lifecycle Models

Waterfall

Traditionally, software was developed in sequential steps, as demonstrated in Figure 3. First, the team needs to understand and gather the requirements of what the software system needs to do, then design the system so that the requirements can be satisfactorily met. Taking the completed design and architecture, developers implement the system, followed by testing to ensure that the software works as intended. Finally, the software system is deployed to the users, and maintained as required. The main concept is that each of the steps must be done sequentially in order to fully understand and implement the system correctly.

Software systems had a reputation for high failure rates, budget, and schedule overruns, and not meeting the users' needs. The source of these problems was that working software is only produced at the very end of the waterfall development lifecycle. This caused high amounts of risk and uncertainty in understanding whether the requirements could successfully be met, as well as whether the users would be satisfied with system (Ben-Zahia & Jaluta, 2014). Additionally, it was difficult to assess progress, and testing efforts would often be cut short due to schedule and budget overruns (Davis, 2000). As technology began to change quickly, the completed systems would either no longer be applicable to the current needs or compatible with updated or changed platforms (Sinha & Das, 2021).

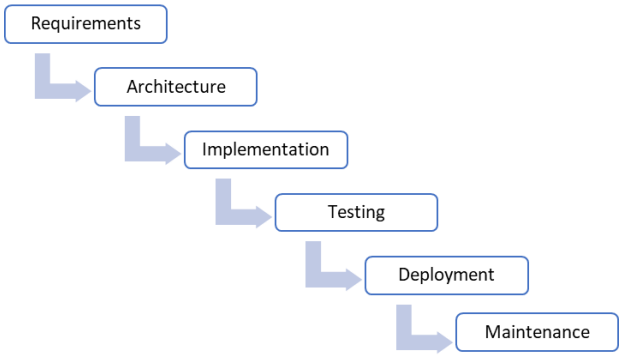

Figure 3. Waterfall software development lifecycle model

Agile

To react to the increasing changes in technology and users' needs, a group of software developers came up with a way to speed up software development and deploy more quickly to market/field. The group developed a manifesto and 12 principles to define the goal and main tenets build software successfully (Beedle, et al., 2001). The main tenets are to shorten the time it takes to get working software to users, and continuously and quickly get feedback from users. The lifecycle model constructed to fulfill the manifesto and the 12 principles are visually described in Figure 4. Instead of performing the steps needed to develop

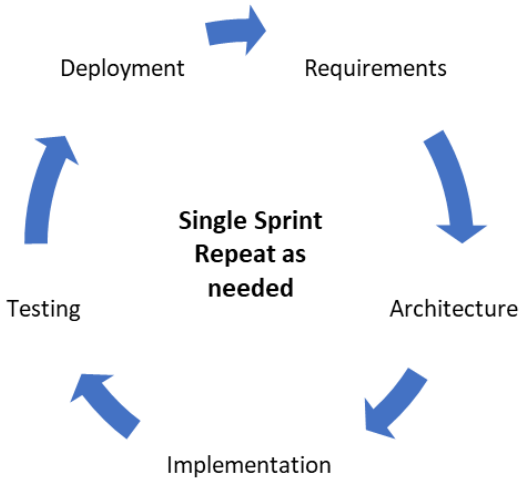

Figure 4. Agile software development lifecycle model

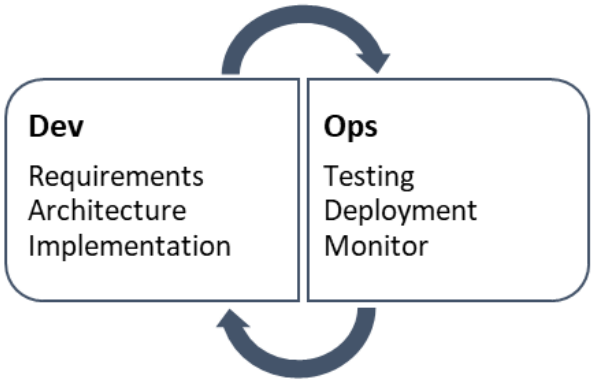

Figure 5. DevOps software development lifecycle model

software sequentially as in the Waterfall model, they are performed iteratively in short "sprints" or iterations throughout the lifecycle. This allows the developers to get feedback from users on a regular basis, demonstrate progress by demonstrating working software, and incorporate changes to the requirements or needs. Many commercial organizations and teams reported being able to deploy software to the market/field earlier, higher development productivity, cost savings, and better customer experience and satisfaction as a result of implementing Agile practices and methodologies (Russo, 2021).

DevOps

While Agile made developing, testing, and deploying software rapidly a common phenomenon, many organizations had separate development and testing teams in order for the testing and verification to be independent from the development efforts. Additionally, many tools to automate various activities (such as developing, testing, and deploying software) became more widely available and highly utilized in development environments. The use of parallel teams and increased use of automation coined the term DevOps to further shorten the development cycle and get operational software out to the users at a faster pace (see Figure 5). Generally, people have been using Agile and DevOps methodologies in conjunction. In some ways,

Dataset	Program Level	Data Description	Data attributes	Data Filters
Dataset A	Total or by Increment	· Targeted Ground systems and software-intensive programs across the Air Force and Space Force. · Data comes from Earned Value reports from contractors, which includes all costs to-date by WBS element. Also includes an Estimate At Completion (EAC) for incomplete programs.	Costs by major program elements (SEITPM, Software, Hardware, and Space segment) as well as software development hours, ESLOC (Equivalent Source Lines of Code), Requirements, Agile-like development process, % Complete, data sources, period of performance in months.	At least 85% complete, to ensure confidence in actual and estimated costs. Also, removed programs included in below dataset.
Dataset B	Annual – summed for Total or Total to Date	· Targeted Ground systems and software-intensive programs across the Air Force and Space Force. · Data comes from the Government's budgeting tool called CcaRs. Based on Contract Line Item Numbers (CLINs).	Costs by major program elements (SEITPM, Software development, and Platform development), as well as ESLOC, Requirements, User Stories, or Story Points.	Programs for which costs could be retrieved to be consistent with the above dataset.

Table 1. Brief description of datasets used

DevOps can be considered as an extension or special case of Agile.

Part 1: Data Analysis

Research Methodology

Datasets

This study uses two sets of data collected from government-funded software development programs collected by the Air Force Cost Analysis Agency (AFCAA), further described in Table 1.

Dataset A focused on identifying whether programs were Agile-like, while Dataset B collected data that followed DevOps processes. Three programs were in both datasets. To avoid double counting these programs, the versions from Dataset A were removed from this analysis. Note, though previous research found that 92.5% complete is equivalent to a complete program (Tracy & White, 2011), this study uses a 85% completion as the threshold to balance between accuracy and retaining data points. Most data points represent large, in-progress programs.

As mentioned in the Software Development Lifecycle Models section above, many teams and organizations utilize both Agile and DevOps processes in conjunction. Therefore, the Agile-like and DevOps programs are grouped together.

Group	Data Sources	# of data points
Waterfall	· Dataset A	30
Agile/DevOps	· Dataset A · Dataset B	27

Table 2 Software Size Metrics

Table 2 shows that there are a comparable number of data points in the 2 groups used in this study.

Base Year Normalization

As mentioned in Table 1, both datasets used in this study provide the costs of major program elements, and these costs are in terms of Then Year dollars (the cost at the time of spending). To ensure that the data and costs are comparable, the costs were normalized to Base Year 2020 (BY20) dollars. The steps to perform the conversions (explained in Table 3) differ by dataset because of how differently the data was collected for both datasets.

SEITPM Estimation Methodologies

Typically, SEITPM effort and costs are estimated in comparison to the Prime Mission Product (PMP), which is the actual software development and infrastructure costs (costs needed to support the development and/or operations environment,

Dataset	Data Source Description	BY20 Conversion Method
Dataset A	Data comes from Earned Value reports, which means the dollars are a cumulative sum of Then Year dollars (dollars' value at time of spending).	***Mid-Point Method*** The mid-point or middle year of a program is used (start and end years are provided in the data) as the original Constant Year (CY), which is then converted to BY20 by applying appropriate escalation indices.
Dataset B	Data comes from budget tool that stores costs on annual basis (in Then Year dollars).	***Sum of Annual Escalations*** Since costs are provided on annual basis, each year's costs are escalated to BY20 dollars. All the converted years' costs of a program are summed up for the total cost.

Table 3. Ways to group and estimate SE, IT, and PM efforts and costs

such as software licenses and supporting hardware). SEITPM is compared to PMP in 2 ways, typically (Markman, Ritschel, & White, 2021):

1. SEITPM is estimated as a factor of, or in proportion to, the PMP costs (SEITPM/PMP)
2. Using a regression where SEITPM costs is a dependent variable and PMP costs is the independent variable. The resulting regression is also called a Cost Estimating Relationship (CER)

As explained previously, SEITPM consists of 3 types of labor/activities: Systems Engineering, Integration and Test, and Program Management. Depending on how teams actually track and bucket their costs and efforts across these 3 activities, it is very common for these 3 activities to be grouped or separated in the 3 ways demonstrated in Table 4.

Analysis Method

The primary objective of this research study is to determine whether there is a significant difference in SEITPM costs between Agile/DevOps and Waterfall programs. As mentioned in the previous subsection, SEITPM is estimated in 2 ways: as a factor of PMP costs or using a regression against PMP costs. Therefore, this study analyzes if there is a difference in SEITPM costs across the Agile/DevOps and Waterfall groups by looking at the data in both ways. A high-level description of the analysis method by type is explained in Table 5.

Numerator or Dependent Variable	**Denominator or Independent Variable**	**Total Costs**
SE + IT + PM	PMP	SEITPM + PMP
SE + PM	PMP	SEPM + PMP **+ IT***
SE + PM	PMP + IT	SEPM + (PMP + IT)

* Note, IT costs need to be added separately to get the total program's cost in the 2nd option/row

Table 4. Ways to group and estimate SE, IT, and PM efforts and costs

Estimation Method	Analysis Method
SEITPM/ PMP Proportion	Compare the means of the SEITPM/PMP proportions, as well as the individual activities' proportions (Systems Engineering (SE), Program Management (PM), and Integration and Test (IT)), between the 2 groups using t-test. The t-test should return a p-value of less than 0.05 for difference to be considered statistically significant. The variables used as inputs are log-transformed and tested for normal distribution using the Shapiro-Wilk test (need a p-value of at least 0.05). If the variables are not normally distributed, the Mann-Whitney test is run, which also requires a p-value
SEITPM vs PMP Regression /CER	Include a categorical/dummy variable for Agile/DevOps and evaluate the p-value of the coefficient, as well as goodness of fit and prediction accuracy statistics. The p-value of the coefficient should be less than 0.05 for statistical significance.

Table 5. Summary of analysis methods by the 2 SEITPM estimation methods

Also explained in the previous subsection are the 3 variants of the SEITPM and PMP costs, and all 3 variants are used in the comparison between Agile/DevOps and Waterfall programs.

Results

SEITPM Proportion Comparison

The t-test is a parametric test, which means that the test assumes the variables used as inputs are normally distributed. Table 6 has the Shapiro-Wilk test p-values for the log-transformed variables (most variables were not normally distributed before the transformation) across the 2 groups (Waterfall and Agile/DevOps), and p-values larger than 0.05 imply the variable cannot reject the null hypothesis of not being normally distributed.

For the variables that returned p-values of less than 0.05 (dark red text in Table 6), the non-parametric Mann-Whitney test is run instead of

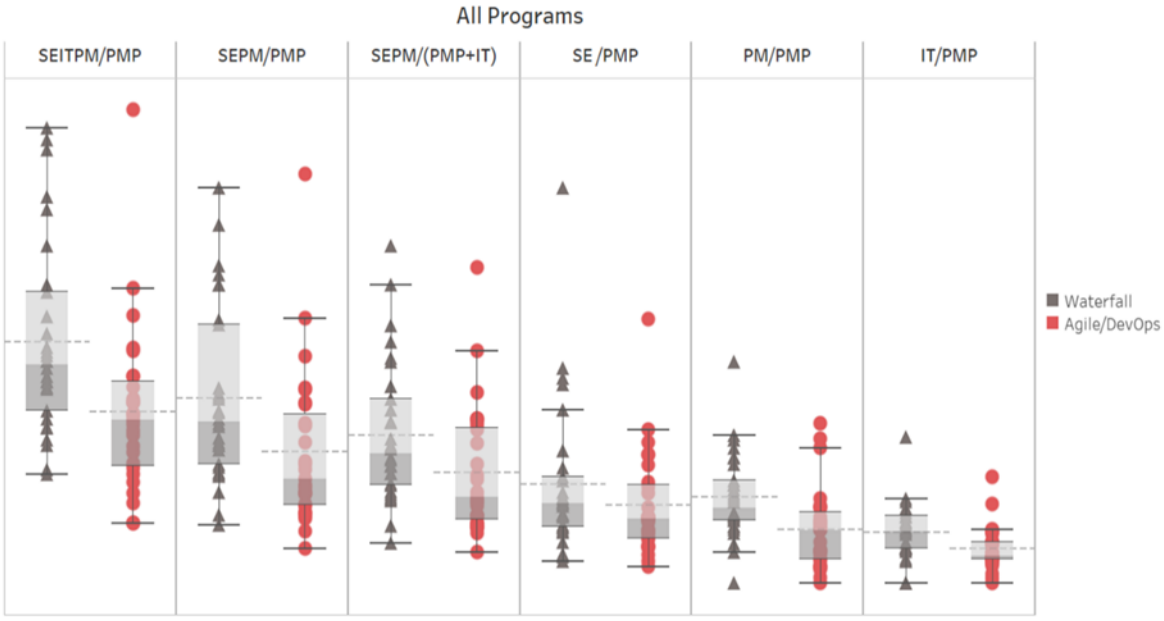

Figure 6. Box plots of the 3 SEITPM variants' and individual activities' (SE, PM, and IT) proportions to PMP costs across Waterfall and Agile/DevOps groups

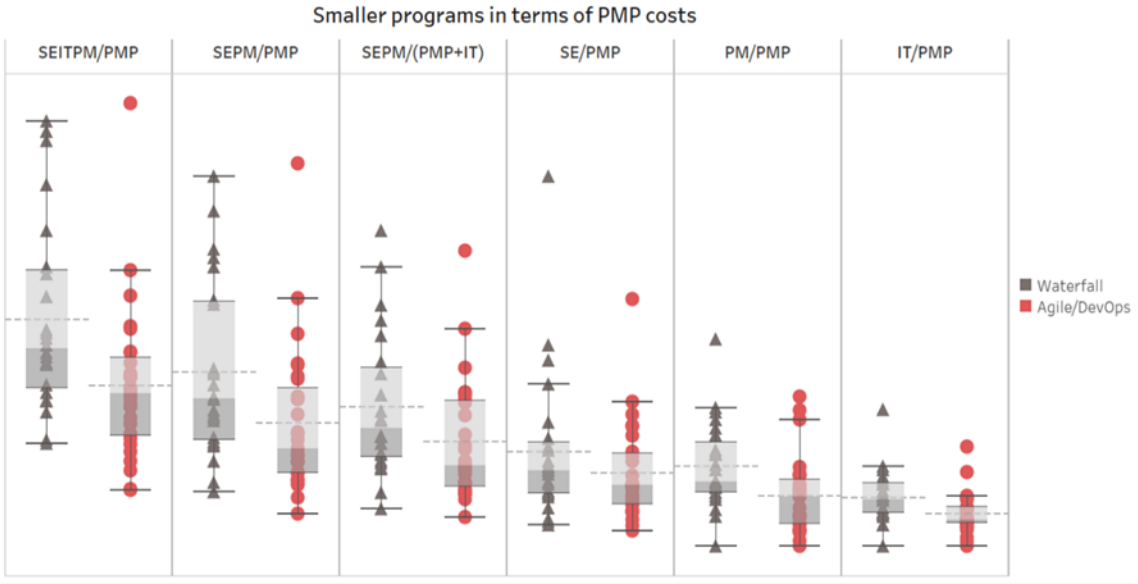

Figure 7. Box plots of the 3 SEITPM variants' and individual activities' (SE, PM, and IT) proportions to PMP costs across Waterfall and Agile/DevOps groups using the subset of smaller programs

	Shapiro-Wilk p-values	
	Waterfall	**Agile/ DevOps**
log(SEITPM/PMP)	0.45	0.81
log(SEPM/PMP)	0.97	0.98
log(SEPM/(PMP + IT))	0.92	0.64
log(SE/PMP)	0.0006	0.02
log(PM/PMP)	0.44	0.06
log(IT/PMP)	0.19	0.04

Table 6. Shapiro-Wilk test for normality p-values on log-transformed variables

the t-test. The Mann-Whitney test also needs to return p-values of less than 0.05 for the difference between the groups to be considered statistically significant.

Along with the p-value, the t-test reports a t-value which represents the ratio of the difference between the two groups' means. Therefore, t-values larger than 1 and p-values of less than 0.05 indicate there is a statistically significant difference between the Waterfall and Agile/ DevOps means for the variable being tested. The Mann-Whitney also produces a W-value, but it is the sum of the ranks of the first sample and does not indicate a difference between the 2 samples. The W-value does not provide a sense of difference or proportions between the 2 samples and, therefore, is not reported in this paper.

Table 7 shows the tests' results comparing Waterfall and Agile/DevOps groups and Figure 6 visually demonstrates the differences between the groups using box plots (the proportions on the y-axis are not shown to maintain confidentiality). Both show that SEITPM, PM, and IT proportions of Agile/ DevOps programs are significantly lower than Waterfall programs.

The largest Agile/DevOps program is significantly

	t-test/Mann-Whitney test	
	t-values	**p-values**
log(SEITPM/PMP)	3.295	0.0009
log(SEPM/PMP)	2.84	0.003
log(SEPM/(PMP + IT))	2.13	0.02
log(SE/PMP)		0.08
log(PM/PMP)	3.09	0.002
log(IT/PMP)		0.004

Table 7. T-test and Mann-Whitney test results on log-transformed variables

smaller than several programs in the Waterfall group (in terms of PMP BY$M). To compare the means of the SEITPM proportions of PMP costs across similarly-sized programs, the dataset is trimmed at programs with PMP costs that are no larger than 5% more than the largest Agile/ DevOps program.

Re-running the above-explained analyses for the smaller programs subset of the data led to the same conclusions: SEITPM, PM, and IT proportions for Agile/DevOps programs are significantly lower than Waterfall programs. SE is the only activity whose difference between the Agile/DevOps and Waterfall groups is not statistically significant. Table 8 and Figure 7 show the statistical test results and the visual representation of the groups' behaviors across the SEITPM variants and individual activities, respectively. As before, the dark red text in Table 8 represents tests with p-values that suggest the

	Shapiro-Wilk p-values		***t-test/Mann-Whitney test***	
	Waterfall	**Agile/ DevOps**	**t-values**	**p-values**
log(SEITPM/PMP)	0.43	0.81	3.06	0.002
log(SEPM/PMP)	0.72	0.98	2.54	0.007
log(SEPM/(PMP + IT))	0.92	0.64	2.13	0.02
log(SE/PMP)	0.004	0.02		0.11
log(PM/PMP)	0.61	0.06	2.69	0.005
log(IT/PMP)	0.12	0.04		0.004

Table 8. Shapiro Wilk and either t-test or Mann-Whitney test results on log-transformed variables across the subset of smaller programs

data cannot be considered normally distributed or that the difference between the groups is not considered statistically significant. The proportions on the y-axis in Figure 7 are not shown to maintain confidentiality, but the SEITPM, PM, and IT proportions are about 30% lower for the Agile/DevOps programs.

SEITPM CER (Cost Estimation Relationship)

In order to rigorously compare and evaluate the regressions' and goodness-of-fit statistics, as well as use a curve that fits the actual trend of how SEITPM costs grow, I log-transformed the variables and ran linear regressions. The 2 regressions I compare are:

1. SEITPM vs PMP without any other variables
2. SEITPM vs PMP with Agile/DevOps categorical variable (set to 1 if the program is an Agile/DevOps program or 0 otherwise)

In both cases, the SEITPM and PMP variables are log-transformed. The Agile/DevOps variable is not log-transformed, and Equation 1 displays how the linear regression is run and how it converts back to unit-space. Therefore, all regression statistics displayed in this section are in log-space, not unit-space. To reduce bias in the regression, I used the Minimum-Unbiased-Percentage Error (MUPE) with Modified Marquardt method, which weighs the data points such that the average error percentage is 0 (Hu, 2001).

$$\log(\text{SEITPM}) = a + b \times \log(\text{PMP}) + \text{Agile/DevOps} \times c$$

$$\text{SEITPM} = a \times \text{PMP}^{b} \times (10^{c})^{\text{Agile/DevOps}}$$

Equation 1 Log-transformed linear regression and conversion to unit-space with Agile/DevOps categorical variable

Figure 8 displays that the trendlines of SEITPM costs against PMP costs for Agile/DevOps programs are, with a few exceptions, consistently and proportionately lower than Waterfall programs. Similar trends are visible when SEPM is graphed against PMP and PMP+IT.

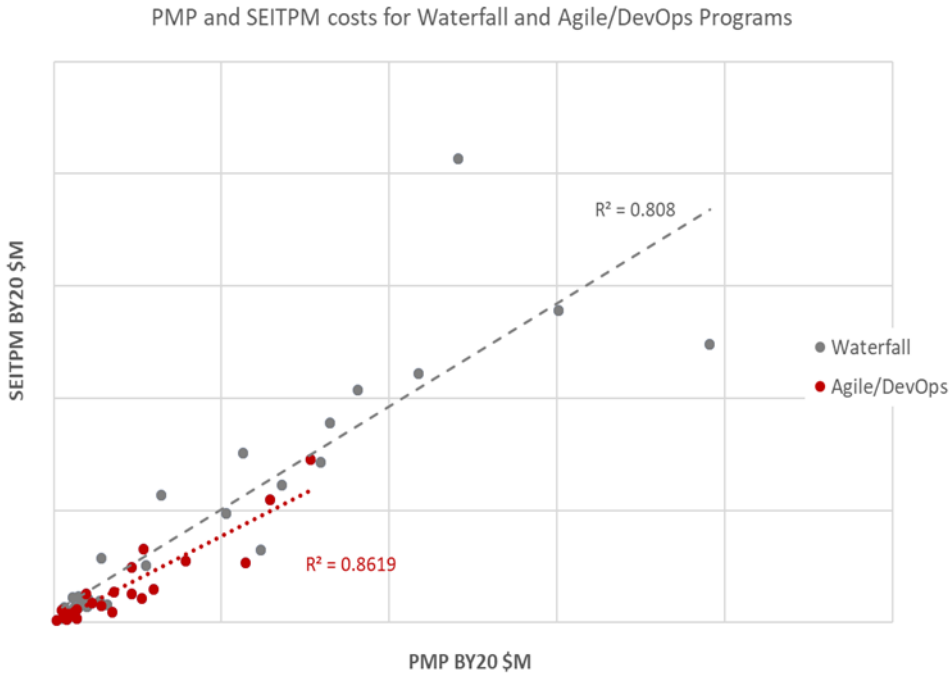

Figure 8. SEITPM costs against PMP (Prime Mission Product) costs trendlines, grouped by development type (Waterfall and Agile/DevOps). Actual data points are removed to preserve the confidentiality of the programs

	Without Agile/DevOps variable			With Agile/DevOps variable		
	SEITPM vs PMP	**SEPM vs PMP**	**SEPM vs PMP+IT**	**SEITPM vs PMP**	**SEPM vs PMP**	**SEPM vs PMP+IT**
Intercept p-value	0.5642	0.6672	0.2724	0.0399	0.3485	0.8554
Agile/DevOps p-value				**0.0028**	**0.0123**	**0.0293**
Adj R^2 for MUPE	85.16%	80.19%	80.62%	87.01%	81.83%	81.79%
Standard Error	0.2026	0.2388	0.2352	0.1849	0.225	0.2261
Average Error %	39.09%	48.78%	47.94%	34.41%	43.95%	44.77%
% of Predictions within 25% of actuals	50.88%	31.58%	31.58%	49.12%	36.84%	33.33%
% of Predictions within 30% of actuals	54.39%	40.35%	42.11%	56.14%	42.11%	43.86%

Table 9. Goodness-of-fit and prediction accuracy statistics for SEITPM/SEPM Regressions/CERs (Equation 2)

$$SETIPM = 1.6734 \times \mathbf{0.6864}^{Agile/DevOps} \times PMP^{0.9005}$$

$$SEPM = 1.3201 \times \mathbf{0.6863}^{Agile/DevOps} \times PMP^{0.8858}$$

$$SEPM = 1.0578 \times \mathbf{0.7206}^{Agile/DevOps} \times (PMP + IT)^{0.8908}$$

Equation 2 SEITPM/SEPM Variants' Regressions/CERs

The p-values on the intercept variable (in log-space) and on the Agile/DevOps variable, and a couple goodness-of-fit statistics on the regressions are listed in Table 9. The 6 regressions are for the 3 variants of SEITPM with and without the Agile/DevOps categorical variable. The results in Table 9 show that Agile/DevOps categorical variable is statistically significant (the p-values are well below 0.05 for all 3 variants of SEITPM) and the goodness-of-fit statistics are better than the regressions without the categorical variable. Additionally, the base/coefficient values for the Agile/DevOps variables (in Equation 2) suggest that SEITPM costs are about 30% lower for Agile/DevOps programs compared to Waterfall programs (similar to the results found when comparing the means in the SEITPM Proportion Comparison subsection above). Note, the resulting regressions/CERs in Equation 2 should not be used without understanding the underlying data and its ranges or for application types or domains not represented in the datasets used in this study.

Conclusion

Analyzing the data available on the Space Ground systems concludes that the SEITPM costs are about 30% lower for Agile/DevOps programs compared to Waterfall programs. Looking at each of the activities separately (SE, PM, and IT), Program Management (PM) and Integration and Test (IT) costs are also significantly lower for the Agile/DevOps programs compared to Waterfall programs. While there is a slight reduction in Systems Engineering (SE) for Agile/DevOps programs, the difference is not considered statistically significant.

These differences can be caused by the differences in the Agile and DevOps methodologies compared to Waterfall, such as:

- Systems Engineering (SE) and Integration and Test (IT) activities should be more incremental and level-loaded, along with software development activities (Seaver, 2018).
- The Agile principles encourages teams to be self-organizing and be part of the task management and decision-making process. Therefore, moving some of the Program Management and Systems Engineering

activities down to the software development team (Beedle, et al., 2001).

- Agile teams are cross-functional and breaking out the effort and costs for specific activities becomes difficult, if not impossible (for example, software development and integration activities) (Beedle, et al., 2001).
- The Agile principles suggest maximizing the amount of work that is not done or streamlining the processes to focus on doing just enough work (Beedle, et al., 2001).

A concern of the reduced activities (and as a result, cost) is whether there would be adverse effects on the product's quality, such as not being able to meet scalability or level of service requirements. To understand whether applying the Agile and DevOps methodologies lead to a reduction in SEITPM costs and whether this reduction leads to lower product quality, the next step of this research was to survey and have discussions with industry partners asking for insights, causes, and effects of the phenomenon.

Part 2: Survey Industry Research Methodology

Survey Questions

The goals of surveying industry were to understand whether or not the software development teams were actively noticing that the Agile/DevOps programs required less SEITPM activities, as well as the causes and effects of this phenomenon. The questions formulated to meet these goals, along with Agile principles or beliefs that support the questions are in Table 10.

Survey Participants

I worked with Space Systems Command (SSC) Financial Management Cost Research (FMCR) department to set up meetings with their industry partners to brief the data analytics results and get their answers on the questions listed in the previous subsection. These industry partners are also represented in the dataset used in the first part of this research study. The suggestions I made for the participants to attend the meeting and respond to the questions were Program Managers, cost analysts, and/or team members that have:

- An understanding of the SEITPM efforts, staffing, and/or costs
- Worked on an Agile/DevOps program that is at least 75% complete
- And also worked on a Waterfall program to be able to comment on the differences between Waterfall and Agile/DevOps programs (or members from both types of programs could also join for real-time comparisons)

The participants were given 2 options for how to order the briefing of the results and answering the questions:

1. Participants could provide responses before the briefing. I would then review the responses and ask follow-up questions after briefing the results.
2. Participants can first view the briefing of the results and dynamically answer the questions during the meeting. This option allowed for participants to get necessary context and background for the questions, which may help participants get clarification and figure out who can answer the questions.

I received responses and held meetings with 5 organizations, using a combination of the two methods above with a combination of Program Managers, cost analysts, and software developers. The organizations and respondents are not mentioned in this paper to maintain confidentiality.

Results

In many cases, the industry partners provided very extensive responses to the questions. In this paper, I provide a summary of the responses that sufficiently answer the questions.

Question 1: Include SEITPM in Scrum/Development Teams?

***Agile Principles** [4]*	***Q#***	***Questions***
Teams should be highly collaborative, self-organizing, and cross-functional.	1	On Agile/DevOps programs, do you include SEITPM FTEs in the Scrum/ development teams? Does the role of the SEITPM FTEs in the Scrum/ Development teams focus only on the Scrum team product? Are any overarching system-level Systems Engineering or system architecture efforts included?
Incrementally gather requirements, develop and test software, and deliver to users.	2	Are SEITPM hours/cost level-loaded across the lifecycle versus high in the beginning for Agile/DevOps programs?
		The data we have suggests that the overarching SEITPM is about 20% lower for Agile/DevOps programs compared to Traditional programs. By looking at each of the activities (Systems Engineering, Program Management, and Integration & Test) separately:
The best architectures, requirements, and designs emerge from self-organizing teams.	3	Systems Engineering may have reduced slightly, but not significantly. Are you noticing if the overarching system-level Systems Engineering is about the same across Waterfall and Agile/DevOps programs? If different, how so and why?
Teams should be highly collaborative, self-organizing, and cross-functional.	4	Program Management is significantly less for Agile/DevOps compared to Waterfall programs. Are you noticing the same behavior? What is causing that (examples: reduced deliverables, management activities being moved into development teams)?
Incrementally gather requirements, develop and test software, and deliver to users.	5	Integration & Test is significantly less for Agile/DevOps compared to Waterfall programs. Are you noticing the same behavior? What is causing that (example: integration and testing efforts being captured within development efforts, as they moved into Scrum/development teams)?
Teams should be highly collaborative, self-organizing, and cross-functional.	6	On 2 different datasets, Causal Inference algorithms found a causal link between analyst and programmer capability. From my previous experiences, I found that teams that had good analytical skills also had the tendency to be better programmers. Have you noticed if the analytical and/or programming skills of the developers improved with SE and PM FTEs being involved in the sprints/iterations?
The best architectures, requirements, and designs emerge from self-organizing teams.	7	Has including SEITPM FTEs in the Scrum/development teams led to improved requirements gathering and accuracy, architectures, and designs?
Incrementally gather requirements, develop and test software, and deliver to users. Incremental deliveries, feedback loops, and frequently tested software lead to better working software and higher customer satisfaction.		Since requirements are gathered and the design/architecture is built incrementally:
	8	Have you noticed positive or negative changes in the quality of products?
	9	Have you noticed any trouble with meeting level of service requirements later in the development lifecycle, compared to when using the Waterfall lifecycle mode?
	10	Has the maintainability of the product improved/decreased for Agile/ DevOps programs compared to Waterfall?
	11	Have you noticed reduction/increase in rework, scrapped code, and defects?

Table 10. Industry Survey Questions

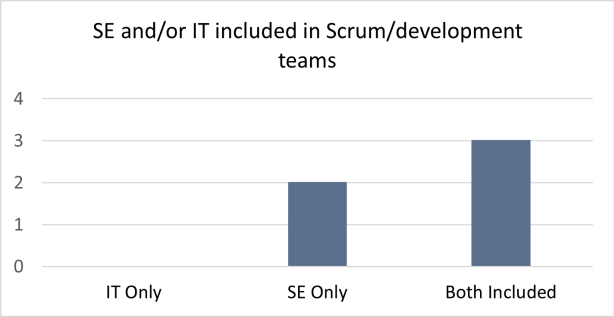

Figure 9. Quantitative Summary of Survey Question 1 Responses

Question: On Agile/DevOps programs, do you include SEITPM FTEs in the Scrum/development teams? Does the role of the SEITPM FTEs in the Scrum/Development teams focus only on the Scrum team product? Are any overarching system -level Systems Engineering or system architecture efforts included?

The goal of the first question is to see if organizations are creating cross-functional teams in practice, and whether systems engineers perform any overarching system-level functions within that role. While systems engineers are needed to ensure that a single component works as expected, Systems Engineering (SE) at the overarching system-level ensures that components are able to integrate and that the system as a whole works as expected. Summaries of responses received are represented in Figure 9.

In general, the industry is creating cross-functional teams that include software developers, systems engineers, and in some cases, testers. However, the SEs typically only serve to provide support in the development of the team's tasks. Hence, no Systems Engineering (SE) that could be attributed to the systems-level is being done within the development/Scrum teams.

Question 2: Is SEITPM level-loaded?

Question: Are SEITPM hours/cost level-loaded across the lifecycle versus high in the beginning for Agile/DevOps programs?

Since Agile and DevOps methodologies promote performing all activities in an iterative fashion, the SEITPM activities and efforts should be mostly level-loaded across the lifecycle in comparison to the Waterfall programs. All industry partners confirmed noticing the same phenomenon.

Question 3: Reduction in Systems Engineering?

Question: Systems Engineering may have reduced slightly, but not significantly. Are you noticing if the overarching system-level Systems Engineering is about the same across Waterfall and Agile/ DevOps programs? If different, how so and why?

In the first part of this research study, the Mann-Whitney test suggested the means of SE/PMP were not significantly different between Agile/ DevOps and Waterfall programs. With this question, the industry partners let us know whether they noticed any significant reductions in the amount of SE used or needed for Agile/ DevOps programs compared to Waterfall ones.

Organization 4 worked on a program where they initially thought they were realizing a 65%

Industry Partner	Summarized Answer
Organization 1	Similar amount of SE activities. Maybe some more upfront activities, but balances with savings by including SE FTEs with the development team.
Organization 2	Don't have data, but probably similar between Agile and Waterfall
Organization 3	Slight reduction, but similar. Developers tend to pick up some of the functionality along the way.
Organization 4	Not sure.
Organization 5	One program noticed higher SE activities and costs compared to a typical Waterfall program, but noted that the nature of the program warrants this. On another program, the team is noticing significantly lower SE costs because the activities are being pushed down to the software development teams.

Table 11. Survey Question 3 Response Summaries

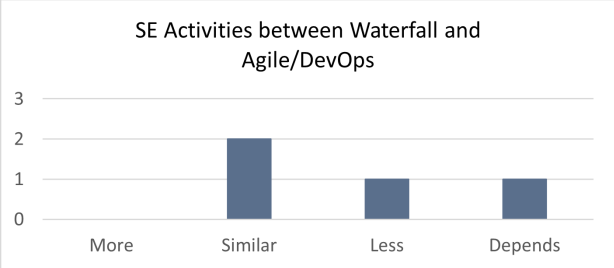

Figure 10. Quantitative Summary of Survey Question 3 Responses

savings in SE costs. Later, they realized they did not do enough SE activities upfront, which led to increased costs later in the lifecycle. Therefore, they are not sure if the SE costs would be lower for Agile/DevOps programs in an ideal scenario. This experience demonstrates a concern that insufficient systems engineering can lead to adverse effects on the program.

In general, the industry partners did not have or analyze their data for whether or not the SE costs were different between Agile/DevOps and Waterfall programs. However, most responses indicate that the team did not notice significant changes in SE activities between Agile/DevOps and Waterfall programs. This may indicate that the industry partners are also being cautious with ensuring that enough systems engineering activities are being done on programs.

Question 4: Reduction in Program Management?

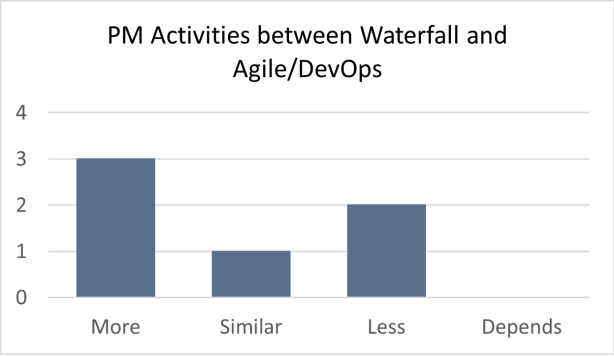

Figure 11. Quantitative Summary of Survey Question 4 Responses

Question: Program Management (PM) is significantly less for Agile/DevOps compared to Waterfall programs. Are you noticing the same behavior? What is causing that (example: reduced deliverables, management activities being moved into development teams)?

This question received mixed answers across the organizations. While the data suggests that PM costs are lower for Agile/DevOps programs compared to Waterfall, the industry partners had different experiences. Three organizations noted that the development team took over some of the PM responsibilities and activities, which leads to a reduction in the PM costs. One organization further noted that the reduction is caused by the developers directly interacting with the Government side of the program, versus going through the PM. Yet, the first two organizations state that the PM activities may have actually increased for Agile/DevOps programs in order to change existing processes and engage the stakeholders regularly.

Question 5: Reduction in Integration and Test?

Question: Integration & Test (IT) is significantly less for Agile/DevOps compared to Waterfall programs. Are you noticing the same behavior? What is causing that (example: integration and testing efforts being captured within development efforts, as they moved into Scrum/development teams)?

Generally, all industry partners are seeing a reduction in IT costs because the activities are either being bucketed with development or because of savings from automated and continuous testing.

Question 6: Improvements in analytical and/or programming skills?

Question: On 2 different datasets, Causal Inference algorithms found a causal link between analyst and programmer capability. From my previous experiences, I found that teams that had good analytical skills also had

Industry Partner	Summarized Answer
Organization 1	Improvements in peer review and test case development. However, not sure analytical/coding skills improved because of Agile or including SE personnel with the software development teams.
Organization 2	Noticed cross-training between the SE and development personnel, and improvements in the knowledge base.
Organization 3	The Agile methodology provides opportunities for developers to demonstrate their skills more compared to Waterfall.
Organization 4	Noticed an increased in productivity with including a SE with the development team.
Organization 5	Noticed an increase in productivity because SE and IT personnel being part of the Scrum team allows issues to be troubleshooted faster.

Table 12. Survey Question 6 Response Summaries

the tendency to be better programmers. Have you noticed if the analytical and/or programming skills of the developers improved with SE and PM FTEs being involved in the sprints/iterations?

Causal Inference algorithms attempt to discover causal relationships from observational data. I applied these algorithms on 2 software development datasets, and the algorithms returned a link between analyst and programmer capabilities in both datasets (though, I did not emphasize or report this result in the studies, as the focus was on causal relationships with effort and schedule) (Hira, Boehm, Stoddard, & Konrad, Preliminary Causal Discovery Results with Software Effort Estimation Data, 2018) (Hira, Boehm, Stoddard, & Konrad, Further Causal Search Analyses With UCC's Effort Estimation Data, 2018) (Alstad, Hira, Brown, & Konrad, 2021). In general, industry agrees that including a system engineer with the Scrum/development teams improves productivity, and that the Agile methodology allows developers to demonstrate and improve their analytical and programming skills.

Question 7: Requirements, Architectures, and Designs Improving?

Question: Has including SEITPM FTEs in the Scrum/development teams led to improved requirements gathering and accuracy, architectures, and designs?

Industry Partner	Summarized Answer
Organization 1	Improvements in requirements gathering, architectures, and designs do not come free with Agile/DevOps. Need a higher-level architecture team.
Organization 2	Really see improvements when stakeholders participate in planning meetings. They are able to clarify and see the requirements.
Organization 3	Noticed less rework, which implies better accuracy. Architecture can depend on external systems and other dependencies, but easier to incorporate changes with Agile/DevOps model.
Organization 4	Not sure (don't have sufficient experience to comment on this)
Organization 5	One program did not start to adopt Agile methodologies until a bit later, but the developers found some of the requirements are not as testable as they could and should have been. Therefore, they are having to rewrite them. Another program started with Agile/DevOps methodologies and found the design is better as a result.

Table 13. Survey Question 7 Response Summaries

One of the Agile Principles states that "the best architectures, requirements, and designs emerge from self-organizing teams" (Beedle, et al., 2001). From the responses received in Table 13, industry generally notices improvements in requirements gathering, designs/architectures, and rework as a result of adopting Agile/DevOps methodologies. However, as the first organization stated, this does "not come free." The improvements depend on having a good architecture team, the team engaging with the stakeholders, and working together to write the requirements.

Question 8: Change in Quality of Products?

Question: Since requirements are gathered and the design/architecture is built incrementally, have you noticed positive or negative changes in the quality of products?

While 2 organizations have experienced both positive and negative changes to product quality, 3 organizations have noticed improvements in product quality as a result of adopting Agile/ DevOps methodologies. While product quality can improve, teams must ensure to not lose focus of the bigger picture and not think of their development environment as a playground.

Question 9: Trouble with Meeting Level of Service Requirements?

Question: Since requirements are gathered and the design/architecture is built incrementally, have you noticed any trouble with meeting level of service requirements later in the development lifecycle, compared to when using the Waterfall lifecycle mode?

"Level of service" requirements refer to requirements that affect the usage of the software systems, such as meeting availability, reliability, scalability, etc. needs. One concern with the design/architecture being built incrementally is whether the architecture/design can and will scale to the needs of the users, especially if these requirements are pushed towards the end of the lifecycle.

While 2 organizations could not comment on this question, the remaining 3 noticed that there is no issue in meeting level of service requirements as long as the discussions, implementing, and testing of these requirements are being done early.

Question 10: Change in Maintainability?

Question: Since requirements are gathered and the design/architecture is built incrementally, has the maintainability of the product improved/ decreased for Agile programs compared to Waterfall?

For this question, maintainability refers to how easily existing software can be modified and maintained. Specific metrics were not required for this question, but just the teams' intuition on how easily they were able to make changes to their existing code.

From the responses, it seems the maintainability of software depends on the system itself and decisions made by the team. This question received varied responses across the participants.

Industry Partner	**Summarized Answer**
Organization 1	Stable, upfront requirements needed for less rework. But Agile can lead to rework.
Organization 2	Fewer defects, because seeing and fixing earlier.
Organization 3	Decrease in rework and less defects. Comes down to overall design, complexity of programs, and maturity of teams.
Organization 4	No answer
Organization 5	Same, but earlier in the lifecycle.

Table 14. Survey Question 11 Response Summaries

Question 11: Change in Rework and Defects?

Question: Since requirements are gathered and the design/architecture is built incrementally, have you noticed reduction/increase in rework, scrapped code, and defects?

Agile/DevOps teams are noticing fewer defects at the end of the lifecycle, because defects are being noticed and fixed earlier. Only 1 organization provided insight on rework, which seems to depend on the stability of requirements.

Conclusion

From the survey responses received from and follow-up discussions with industry, the phenomena and insights that are mostly common across the 5 organizations are:

- Software development/Scrum teams are cross-functional: SE and IT full-time equivalents (FTEs) are generally included.
- SEITPM activities/effort/costs are level-loaded across the lifecycle.
- IT costs are lower due to the activities being bucketed with development, and due to savings from automated and continuous testing.
- Including SE FTEs with development/Scrum teams leads to higher productivity.
- Organizations have noticed an improvement in requirements gathering, architectures and designs from adopting Agile/DevOps methodologies.
- There is an improvement in the product quality, though a couple organizations mentioned that they have also had scenarios where there was a negative impact.
- The organizations have not faced challenges in meeting level of service requirements as long as the discussions, implementation, and testing of these requirements are being done early.
- There are fewer defects at the end of the lifecycle because they are found and fixed earlier. The amount of rework required depends on the stability of requirements, however.

However, industry, as a whole, did not have unified or strong insights for the remaining 3 questions in the survey.

The goal of the survey questions was to ask industry if they noticed the reduced SEITPM costs in Agile/DevOps environments and whether that led to positive or negative effects in the final products. In general, organizations and software development teams noticed reductions in Integration and Test (IT) costs most significantly. Though the data suggests Program management (PM) costs are also lower for Agile/DevOps programs compared to Waterfall programs, industry did not necessarily notice a decrease in the PM activities. The organizations also noticed mostly positive effects in product quality, defects, and meeting level of service requirements. While improvements were not necessarily noticed for rework and maintainability, they also did not necessarily worsen compared to Waterfall programs.

Threats to Validity

This research study is based specifically on Ground software systems from the Space Systems Command (SSC) and Air Force. The programs range from new development to modifications to existing systems and vary in terms of functionality provided and sizes. Given the nature of the data used in this study, there are 2 threats to validity:

1. Since the data and survey participants come from Ground systems, the findings in this study might not apply to other application domains (particularly the SEITPM costs estimating regression (Equation 2)). As mentioned in the Future Work section (below), a good future step would be to analyze data across different application domains/types to evaluate how generalizable the findings are.

2. The Agile/DevOps programs in the datasets used in this study have a considerably small total cost/size range compared to the Waterfall programs. Therefore, the results in this paper might not hold for larger, more complex programs using the Agile/DevOps methodologies. Also mentioned in the Future Work section (below) is the suggestion to update this study when larger Agile/DevOps data is collected to observe whether the SEITPM costs are still lower than for the Waterfall programs.

Comprehensive Conclusions

This research study consists of 2 parts:

1. Analyze the SEITPM costs between Agile/ DevOps and Waterfall programs

2. Survey industry to get their insights on the SEITPM cost differences between Agile/ DevOps and Waterfall programs.

The first part of the study showed that the SEITPM costs are about 30% less for Agile/ DevOps programs compared to Waterfall programs and that this difference is statistically significant. By looking at the individual activities separately, the reduction in PM and IT costs Agile/DevOps and Waterfall programs are statistically significant, while the reduction in SE costs is not.

Reduced SEITPM costs can imply insufficient systems engineering and planning activities, which can lead to the program's inability to scale to requirements, increased defects, reduced maintainability of the code, and overall decline in products' quality. The second part of the research study, surveying and having discussions with industry, was designed to understand whether the development teams are noticing a decline in product quality as a side-effect to adopting Agile and/or DevOps methodologies.

Discussions with industry concluded that the software development teams usually did not notice a major reduction in SEITPM costs and activities for Agile/DevOps programs – particularly for SE and PM. One thing to note here is that the industry partners did not study their own data prior to these discussions and were asked to answer based on their intuition. This suggests that there is not an active attempt to reduce SEITPM activities because maintaining product quality is essential. However, they did note that the responsibilities, activities, and cost reporting between software development and SEITPM activities had blurred and overlapped more than on Waterfall programs. In general, the industry noticed either an improvement or similarity in the product's quality, number of defects, rework, and maintainability compared to Waterfall programs.

In answer to the question posed in the title of this paper (are Agile/DevOps programs doing enough systems engineering?), this research study found that software development teams are able to and have been doing enough engineering to produce high quality products while utilizing Agile/ DevOps methodologies and reducing costs.

Future Work

There are several future steps that could enhance this analysis further:

1. Perform a similar analysis on a dataset that contains data points across the various application domains to evaluate whether the findings in this study are generalizable.

2. Reach out to more industry teams, not just SSC's industry partners, to get their responses on the survey questions. With more responses, we may be able to understand if there are patterns that are more common than others as well as all the unique ways Agile/DevOps teams are formed.

3. Collect annual SEITPM costs across multiple Waterfall and Agile/DevOps programs to be able to generalize how the SEITPM activities' levels behave throughout the lifecycle and how they differ between the 2 groups.
4. Update this analysis when more data on Agile/DevOps programs is collected, especially on larger programs. Since the Agile/DevOps programs are significantly smaller than many of the Waterfall programs in the data used, it is unclear if the behavior identified (that SEITPM costs are significantly lower for Agile/DevOps programs compared to Waterfall programs) will continue as the Agile/DevOps programs grow in size and difficulty.

Acknowledgements:

The author thanks the Air Force Cost Analysis Agency (AFCAA) for sharing the data used in this research study, and Raj Palejwala, Natasha Edwards, Adriana Contreras, and Ernest Rangel of the Space Systems Command (SSC) for supporting and reviewing this study. The author also thanks Matt Murdough, Miguel Aceves, and Ben Kwok of Tecolote Research Inc. for reviewing, providing suggestions, and guiding this study from initiation to submission. This study was funded by the Space Systems Commands (SSC) under contract FA8802-19-F-0005.

References:

Alstad, J., Hira, A., Brown, A. W., & Konrad, M. (2021). Investigating Causal Effects of Software and Systems Engineering Effort. *International Cost Estimating and Analysis Association Professional Development and Training Workshop.*

Bassil, Y. (2012). A Simulation Model for the Waterfall Software Development Life Cycle. *International Journal of Engineering and Technology, 2*(5).

Beedle, M., van Benekum, A., Cockburn, A., Cunningham, W., Fowler, M., Highsmith, J., . . . Thomas, D. (2001). *Principles behind the Agile Manifesto*. (Agile Manifesto) Retrieved November 5, 2021, from https:// agilemanifesto.org/principles.html

Ben-Zahia, M. A., & Jaluta, I. (2014). Criteria for selecting software development models. *2014 Global Summit on Computer \& Information Technology (GSCIT)* (pp. 1-6). IEEE.

Davis, G. (2000). Managing the test process [software testing]. *Proceedings International Conference on Software Methods and Tools. SMT 2000* (pp. 119-126). IEEE.

Hira, A., Boehm, B., Stoddard, R., & Konrad, M. (2018). Further Causal Search Analyses With UCC's Effort Estimation Data. *Acquisition Research Program.*

Hira, A., Boehm, B., Stoddard, R., & Konrad, M. (2018). Preliminary Causal Discovery Results with Software Effort Estimation Data. *Proceedings of the 11th Innovations in Software Engineering Conference.*

Hu, S.-P. (2001). Minimum-Unbiased-Percentage Error (MUPE) Method in CER Development. *Third Joint Annual ISPA/SCEA International Conference.*

Manring, J. (2016). Maturing the Economic Aspects of Agile Development in the Federal Government. *International Cost Estimating and Analysis Association Professional Development and Training Workshop.*

Markman, M. R., Ritschel, J. D., & White, E. D. (2021). USE OF FACTORS IN DEVELOPMENT ESTIMATES: IMPROVING THE COST ANALYST TOOLKIT. *Defense Acquisition Research Journal: A Publication of the Defense Acquisition University, 28*(1).

Russo, D. (2021). The Agile Success Model: A Mixed-methods Study of a Large-scale Agile Transformation. *ACM Transactions on Software Engineering and Methodology (TOSEM), 30*(4), 1-46.

Seaver, D. (2018). Agile to DevOPS and its Impact on Estimation and Measurement. *Joint IT and Software Cost Forum.*

Sinha, A., & Das, P. (2021). Agile Methodology Vs. Traditional Waterfall SDLC: A case study on Quality Assurance process in Software Industry. *2021 5th International Conference on Electronics, Materials Engineering \& Nano-Technology (IEMENTech)* (pp. 1-4). IEEE.

Tracy, S. P., & White, E. D. (2011). Estimating the final cost of a DoD acquisition contract. *Journal of Public Procurement*.

Anandi Hira *is currently a Data Scientist/Researcher at the Carnegie Mellon University Software Engineering Institute (CMU SEI). Previously, Anandi performed several Agile and software cost estimation research projects as a cost analyst at Tecolote Research Inc. She received her PhD in software cost estimation under Dr. Barry Boehm at University of Southern California (USC), where she collected data and calibrated the COCOMO® II model to include functional size metrics. Her research interests include software metrics and its application to project management, software cost estimation, and software process improvement.*

The International Cost Estimating and Analysis Association is a 501(c)(6) international non-profit organization dedicated to advancing, encouraging, promoting and enhancing the profession of cost estimating and analysis, through the use of parametrics and other data-driven techniques.

www.iceaaonline.com

Submissions:

Prior to writing or sending your manuscripts to us, please reference the JCAP submission guidelines found at

www.iceaaonline.com/publications/jcap-submission

Kindly send your submissions and/or any correspondence to JCAP.Editor@gmail.com

International Cost Estimating & Analysis Association

4115 Annandale Road, Suite 306 | Annandale, VA 22003

703-642-3090 | iceaa@iceaaonline.org