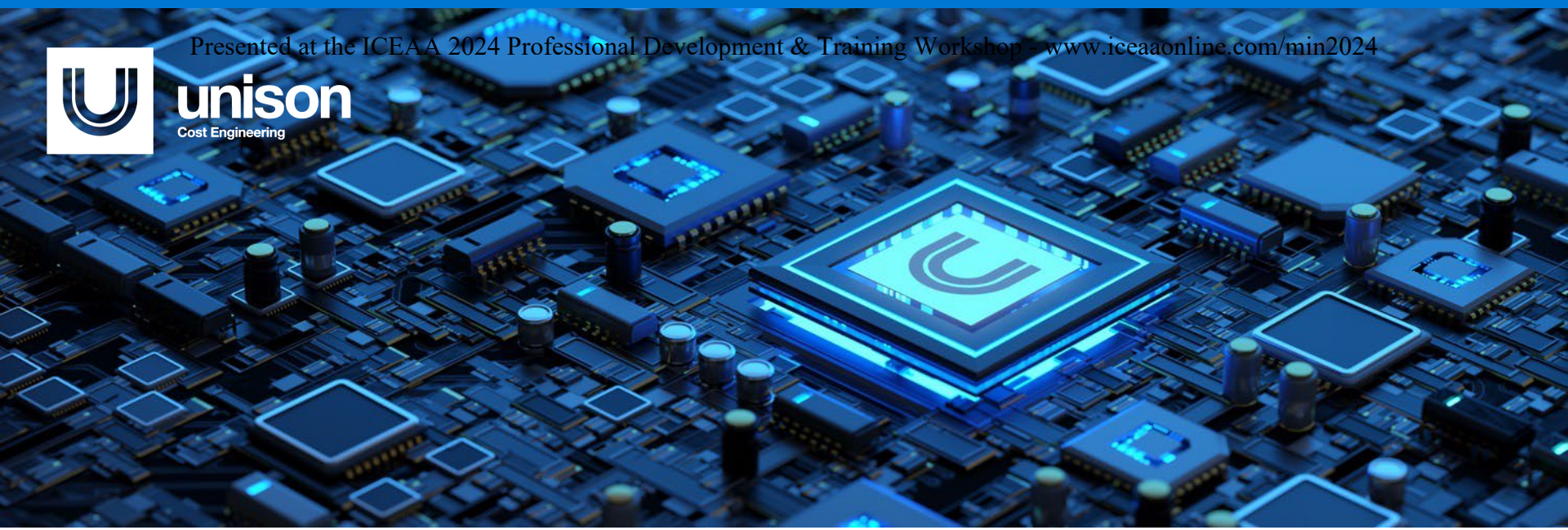
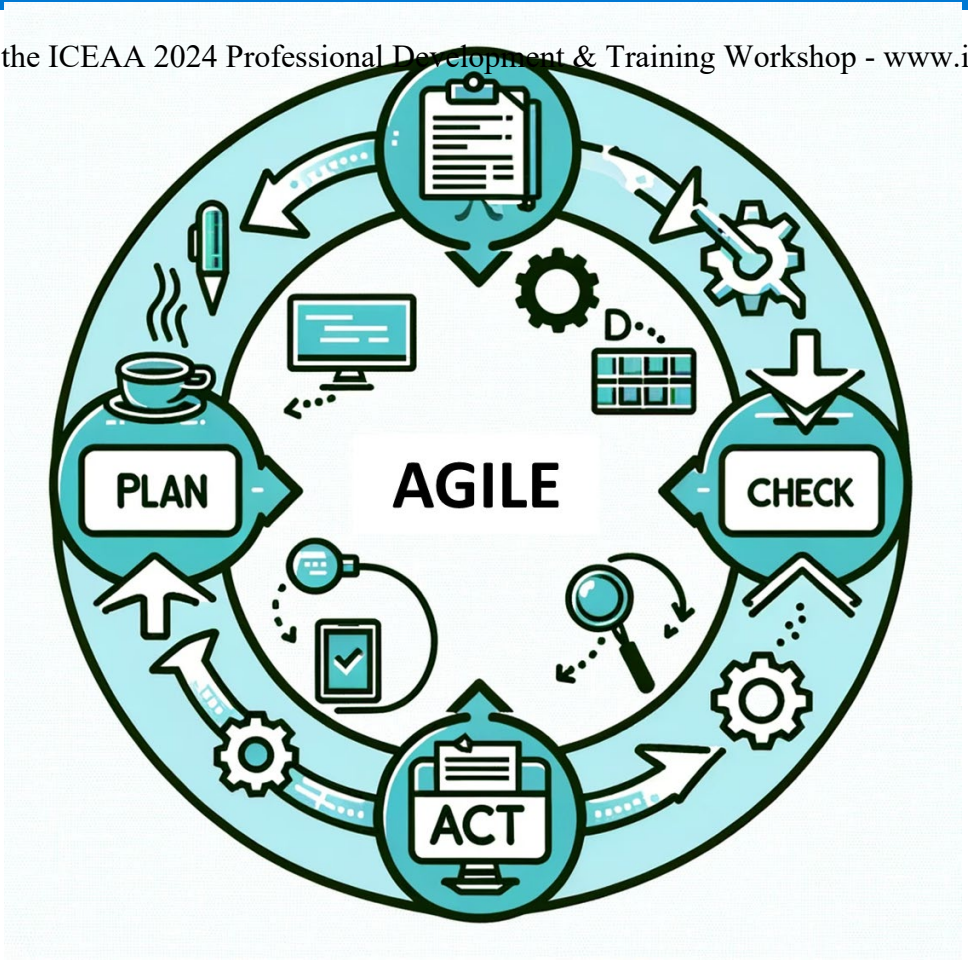
Presented at the ICEAA 2024 Professional Development & Training Workshop - www.iceaaonline.com/min2024

unison
Cost Engineering



Agile Software Development Cost Estimating

Jim Golden, Unison Cost Engineering



Source: Glenn C. Butts, NASA, 2024 .

Introduction

- Agile software development programs focus on near term workload and activities with only limited future planning cycles identified
- Future cycles are only activated as their start date nears
- Any model-based cost estimate or predictive analysis for the budget needs to be flexible, responsive, and adaptive to the daily dynamics of Agile software development program planning and execution
- As a cost estimator, integrating with the IPT for a particular program or project has always been a critical factor in understanding requirements, gathering data, and producing a quality estimate
- With agile processes being adopted more frequently across software development organizations, cost estimators and program offices are challenged even further to work closely with developers to continuously update cost estimates
- Agile sprint results reveal progress of development, and subsequently could affect the cost estimate and budget requests

Agile Framework

- There are many Agile frameworks, each offering methodologies that apply variations of iterative development and continuous feedback
- One of the most popular Agile frameworks is Scrum. It is a simple framework for software development teams to collaborate incrementally and iteratively on delivering value to the customer
- Agile approaches to software avoid the need for very detailed upfront, predictive requirements capture
- These approaches begin with a high-level capture of business and technical needs that provides enough information to define the software solution space, while also considering associated quality needs (such as security)
- This activity aims to define the value proposition of the capabilities to be developed, and the expected impact to the operational mission

Source: (DOD Agile Software Acquisition Guidebook, Feb 27, 2020).

Agile Principles

- Agile principles acknowledge the “cone of uncertainty” and stress collaboration and communication between product owners and developers throughout the program life
- Agile practices are open to requirement changes in support of customer satisfaction and needs
- The approach acknowledges that identifying all requirements (i.e. tasks, features, user stories) upfront isn’t realistic and it also restricts the team from building the best solution
- Focusing on capabilities provides the right level of definition for program planning and development priority and are displayed on the product roadmap
- Capabilities are documented by the product owner (with support from the end-user/warfighter) as epics

Source: (DOD Agile Software Acquisition Guidebook, Feb 27, 2020).

Agile - Epic

- **Epic** - Communicates a high-level need and provides a high-level description of why the end-user/warfighter needs the capability, clarifying the “why, where and how” without going deep into implementation detail
  - *A large body of work to be completed during development*
  - *Depending on the Agile framework, the Epic can be too large to complete within a sprint*
  - *Epics are further decomposed into smaller features and user stories*
  - *Epics may express business functionality or identify constraints placed on the product or system*
- This epic (capability) can be managed on a roadmap to help convey where the priority for this capability falls in relation to other capabilities

Source: (DOD Agile Software Acquisition Guidebook, Feb 27, 2020).

Agile – User Story

- **User Story** - the development team works with the product owner to decompose the epic into smaller units (user stories) to be developed
  - *The user story includes the acceptance criteria that will be used during testing*
  - *Acceptance criteria answer the question of when the user story has been successfully implemented and can be considered "done."*
  - *User stories identify the product owner who owns the user story and is responsible for reviewing and accepting (or rejecting) the work*
- A user story is a concise, written description of a specific user interaction or functionality from the perspective of an end user or stakeholders
  - capture the requirements or desired outcomes and are typically written in a simple format, such as "As a [user role], I want [action] so that [benefit]."
  - User stories are small, independent units of work that can be completed within a single sprint.

Source: (DOD Agile Software Acquisition Guidebook, Feb 27, 2020).

Agile – User Story Sizing

- A few user stories are selected from the product backlog that represent different levels of effort and complexity
- These '**Reference User Stories**' will act as reference points for estimating the rest of the backlog (relative sizing)
- An estimation session is conducted with the development team, including developers, testers, and other relevant stakeholders
  - The selected reference user stories' complexity and effort are discussed
  - The team the collaboratively estimates the story points for the remaining product backlog items by comparing them to the reference stories

Source: (Agile Scrum, Umer Waqar, 2020).

Agile – User Story Sizing – Story Points

- Sizing a user story involves taking the following aspects into consideration
  - Amount of work: How much work is required to complete it?
  - Complexity: how difficult and challenging is it?
  - Time & Risk: how much time would it take and how much uncertainty is involved?
- Story points represent the effort required to complete the user story
  - Each user story is assigned story points
  - Story points measure the size and complexity of the user story
- Story points use different scales for sizing. Different scales include:
  - Linear or Fibonacci Scales
  - T-shirts (X-Small, Small, Medium, Large, Extra-Large)
- Story points are assigned to the reference user stories based on their relative complexity (relative sizing)

Source: (Agile Scrum, Umer Waqar, 2020).

Agile – User Story Sizing – Story Points

- To estimate the value of a story point, the development team defines one user story as a baseline (reference user story)
- All user stories are listed and organized from smallest to largest
- A familiar user story is chosen by the development team as the baseline user story
- The team uses the baseline user story to estimate all the other stories
  - The chosen story point ‘scale’ is used to ‘rate’ each user story
  - Scale values correlate to effort, time, complexity, and uncertainty
- The team then picks the next user story and determines whether it is bigger or smaller than the baseline story
- This process continues until the development team has estimated the size of all the user stories in the product backlog

Source: (Agile Scrum, Umer Waqar, 2020).

Agile – Velocity & Sprints

- Velocity is the number of story points the team can complete in a given time period (e.g., per sprint)
  - Measure the team's historical velocity by summing up the story points completed in previous sprints, which will help in forecasting the team's future capacity
- A Sprint is a fixed period of time where a specific amount of work is completed, and can be considered as cycles of work where the product is developed over multiple iterations
- Sprint Planning – based on the team's velocity, a set of user stories from the product backlog is selected by the team to be worked on in the upcoming sprint
- As the project progresses, the team refines the product backlog and estimates the effort and cost for new stories.
- The team's velocity may change over time, so it's important to update and adjust the estimations accordingly.

Source: (Agile Scrum, Umer Waqar, 2020).

Agile – Velocity Tracking

- Agile estimations are not meant to provide precise predictions but rather to establish a shared understanding among the team and stakeholders
- The estimates should be revisited and refined throughout the project as more information becomes available and as the team gains a better understanding of the work involved
- Over time, the team's historical velocity, which represents the number of story points completed in previous sprints, can be used as a reference point for future estimation. It provides insights into the team's capacity and helps in forecasting future work
- Agile teams typically track their velocity (e.g., story points completed per sprint)
  - Tracking velocity over multiple sprints allows the team to observe trends and patterns in their productivity
  - While this doesn't directly align with a traditional learning curve, it provides insights into the team's capacity and potential improvements in efficiency

Agile – Data Gathering for a Rough Order of Magnitude Estimate

- Obtain the Product Backlog for the software development effort. The Product Backlog is the document which contains the overall list of requirements and features to make the product
- Assume that software Size Units will be "Custom Size Units" called "Story Points" based on the idea that User Stories (Features) in a Product Backlog are sized by the agile development team/developers using techniques like the Fibonacci Sequence (1,2,3,5,8,13,20,40) or variations thereof
- For the software development effort, determine how many:
  - Epics (Releases) are planned for the software development effort
  - User Stories are planned to be completed in each Epic
  - Story Points are assigned to each User Story
  - Sprints are planned for each Epic

Source: (Agile Scrum, Umer Waqar, 2020).

Agile – Data Gathering for a Rough Order of Magnitude Estimate

- What is the period of the Sprint? Typically, this is a time box of two weeks (80 hours) during which a done, usable, and potentially releasable product increment is created
- What will be the assumption for the agile development team's Velocity? Velocity is defined as the number of story points an agile team can complete in a sprint
  - Since agile teams differ across projects, one agile team's velocity may not be the same as another agile team's velocity
  - An agile team's velocity may be determined after at least three (3) completed sprints on the project
  - If no historical team velocity data is available, make an assumption as to the velocity for a starting point and revisit

Source: (Agile Scrum, Umer Waqar, 2020).

Agile – Data Gathering for a Rough Order of Magnitude Estimate

- Given the following information pertaining to an agile team's performance on the first three (3) sprints for a notional project:
  - Sprint 1 = 16 story points completed
  - Sprint 2 = 15 story points completed
  - Sprint 3 = 17 story points completed
- therefore: (16 + 15 + 17) / 3 = 48 / 3 = 16 story points/sprint. This means that the velocity of the agile team is 16. The agile team can complete 16 story points per sprint
- If we know the number of hours in a sprint, and we know the velocity, then we can determine the number of hours per story point:
  - 80 hours / 16 story points = 5 hours/story point

Agile – Data Gathering for a Rough Order of Magnitude Estimate

- How many Agile Software Development Team members/developers are there, besides the Product Owner and Scrum Master?
- What programming language(s) will be used in the effort?
  - A different cost estimate line item may be required if there are multiple software development languages used across the development effort
  - Typical programming languages include: Python, C++, C#, HTML
- For a ROM cost estimate assume 100% New Code
  - 100% new code will result in a high ROM in terms of dollars and hours
  - As the analyst gets more data, the ROM should become a ‘detailed’ estimate
  - The ‘detailed’ estimate can then estimate other types of code such as adapted, deleted, reused, auto-generated, or auto-translated code

Agile – Example Software Application ROM – Assumptions

- The Sprint is a time box of two weeks (80 hours) during which a done, usable, and potentially releasable product increment is created
- The Agile Software Development Team size, besides the Product Owner and Scrum Master, is seven (7)
- Velocity is defined as the number of story points an agile team can complete in a sprint
  - Given the following information pertaining to the agile team's performance on the first three (3) sprints of a project:
    - Sprint 1 = 16 story points completed
    - Sprint 2 = 15 story points completed
    - Sprint 3 = 17 story points completed
  - therefore: (16 + 15 + 17) / 3 = 48 / 3 = 16 story points/sprint
  - The velocity of the agile team is 16. The agile team can complete 16 story points per sprint

Agile – Example Software Application ROM – Assumptions

- If we know the number of hours in a sprint, and we know the velocity, then we can determine the number of hours per story point: 80 hours / 16 story points = 5 hours/story point
- An assumption must be made at this point concerning the programming language(s) that is used by the agile team to write the software code
  - There may be different velocities of the agile team given different programming languages used for certain user stories or tasks within user stories
  - A decision must be made as to whether the velocity will represent the capability of the agile team across all languages that will be used on the project, or
  - Velocities must be determined for user stories that will be coded with specific programming languages
- Assume that the team velocity is 16 story points/sprint, regardless of which programming languages will be used

Agile – Example Software Application ROM – Assumptions

- The programming language for all user stories will be “C”
- The sprint length is 2 weeks (80 hours). Assume 100% New Code
- There will be three (3) sprints with three (3) user stories per sprint as determined by the Product Owner (PO)
- The PO has written each user story and, using Planning Poker with the Agile Team, has determined the number of story points assigned to each User Story:

User Story A – 5 SP	User Story B – 6 SP
User Story C – 6 SP	User Story D – 5 SP
User Story E – 6 SP	User Story F – 5 SP
User Story G – 5 SP	User Story H – 5 SP
	User Story I – 5 SP

- This results in a total of 48 story points that the agile team must complete

Agile – Example Software Application ROM – Assumptions

- The Product Owner determines that the highest value stories will be completed as soon as possible given the velocity (capability) of the agile team
  - Sprint 1 will have the agile team complete:

    User Story A – 5 SP User Story D – 5 SP User Story E – 6 SP
  - Sprint 2 will have the agile team complete:

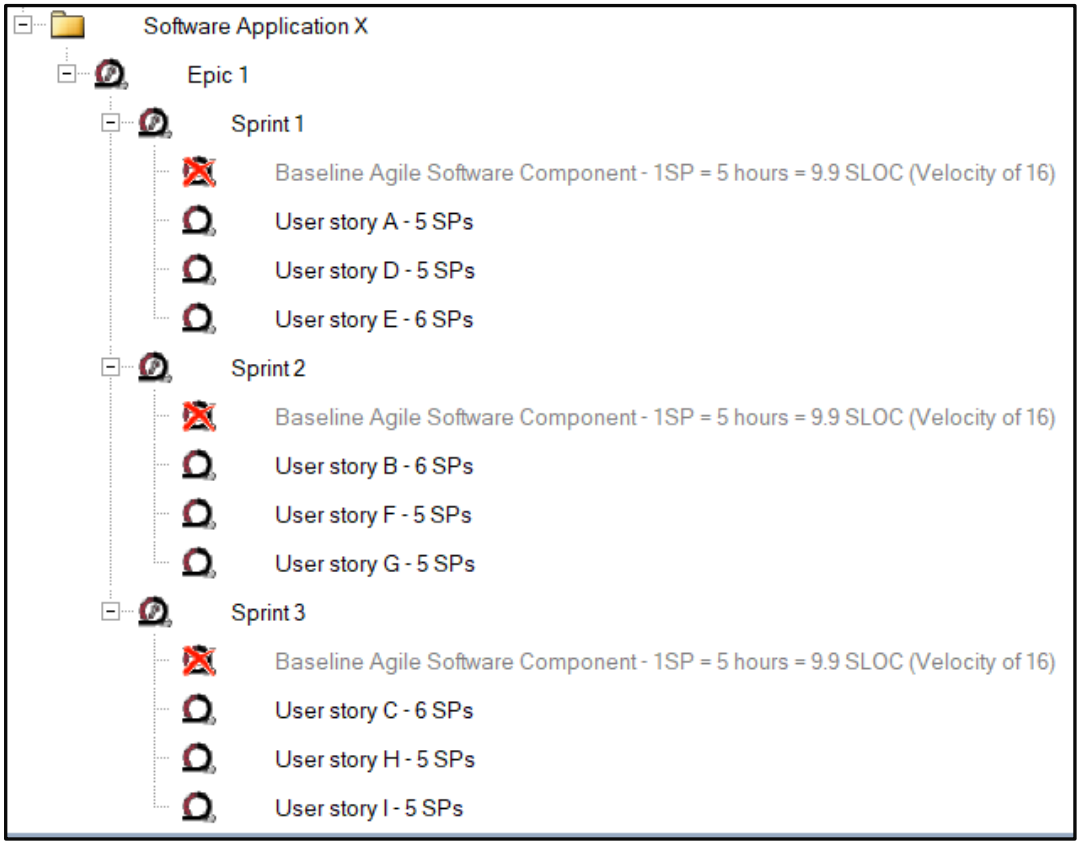
    User Story B – 6 SP User Story F – 5 SP User Story G – 5 SP
  - Sprint 3 will have the agile team complete:

    User Story C – 6 SP User Story H – 5 SP User Story I – 5 SP
- Since we know that the team velocity of 16 story points per sprint (80 hours), we know that 1 story point equals five (5) hours of effort
- You can choose, at this point, to establish a “Relative Size Unit”, such as Source Lines of Code (SLOC). Assuming a “Size Conversion Factor” of 9.90 yields 1 SP = 9.9 SLOC

Agile – Example Software Application ROM – Assumptions

- Populate the Product Breakdown Structure (PBS) or whatever cost estimating structure you are using (WBS, CES, etc.) with Sprints and associated user stories
  - Epic
  - Sprints
  - User Stories
- Populate the New Size with the number of story points designated for each user story (see items 11, 12 & 13 above). Remember…the size is now story points NOT source lines of code, unless you have converted all story point estimates to SLOC
- Set Length of Iteration or Sprint to 2 weeks
- Initial setup for a ROM is complete, notwithstanding the changes needed for operating spec, labor rate burdens, life cycle dates, escalation, etc.

Agile – Example Software Application ROM – PBS



Agile – Example Software Application ROM – Metrics

Metrics : Software Application X - [Folder] Currency in USD ($) (as spent)	Value	Units
Development Cost	53093.27	$
Development Labor Hours	307.73	Hours
Development Duration	5.00	Months
Drivers----------		
Total Software Size	950.40	Source Lines of Code (SLOC)
Project and Productivity Details ----------		
Total Cost Per Software Size Unit	55.86	$/Source Lines of Code (SLOC)
Total Hours Per Software Size Unit	0.32	Hours/Source Lines of Code (SLOC)
Development Cost Per Software Size Unit	55.86	$/Source Lines of Code (SLOC)
Development Hours Per Software Size Unit	0.32	Hours/Source Lines of Code (SLOC)
Software Defects Injected	11.48	Defects
Software Defects Removed	9.80	Defects
Total Software Defects	1.69	Defects

Agile – Example Software Application ROM – Estimated Labor Hours

Labor Requirements (Hours) Software Application X	Total		Labor Requirements (Hours) Software Application X	Total
Release Planning			Develop, Integrate, Test	
Release Train Engineer	4.01		Product Owner	24.78
Product Manager	2.50		Scrum Master	40.72
System Architect/Engineer	6.01		Agile Team	111.53
Product Owner	6.04		Subtotal	177.03
Scrum Master	8.46		Final Release, Test and Certification	
Agile Team	9.66		Product Owner	7.18
Subtotal	36.68		Scrum Master	10.06
Requirements, Design and Integration			Agile Team	21.55
Release Train Engineer	8.65		Subtotal	38.79
Product Manager	7.87			
System Architect/Engineer	22.81		**Total**	**307.73**
Subtotal	39.34			
Release, Test and Certification				
Release Train Engineer	5.72			
Product Manager	3.18			
System Architect/Engineer	6.99			
Subtotal	15.89			

Agile – Example Software Application ROM – Estimated Cost

Cost ($) - Software Application X	Total		Cost ($) - Software Application X	Total
Release Planning			Develop, Integrate, Test	
Release Train Engineer	$604.32		Product Owner	$4,542.01
Product Manager	$500.94		Scrum Master	$5,877.52
System Architect/Engineer	$1,227.40		Agile Team	$19,324.05
Product Owner	$1,106.87		Subtotal	$29,743.58
Scrum Master	$1,220.60		Final Release, Test and Certification	
Agile Team	$1,674.39		Product Owner	$1,316.45
Subtotal	$6,334.52		Scrum Master	$1,451.71
Requirements, Design and Integration			Agile Team	$3,733.92
Release Train Engineer	$1,305.07		Subtotal	$6,502.09
Product Manager	$1,582.25			
System Architect/Engineer	$4,684.50		**Total**	**$53,093.27**
Subtotal	$7,571.81			
Release, Test and Certification				
Release Train Engineer	$862.90			
Product Manager	$640.28			
System Architect/Engineer	$1,438.09			
Subtotal	$2,941.28			

Conclusion

- The transition to agile management approaches for software development has introduced several challenges for cost estimating teams, especially as they support long-term budget development and program execution
  - Required software capabilities are mapped to short release cycles/increments, focusing on the immediate cycle in-work.
  - Software development and testing are accomplished in short term iterations, such as sprints.
  - Daily changes may occur in the order and priority for work to be completed in each release cycle
  - Effort is measured based on qualitative sizing (e.g., story points) relative to the specific software being used and the development cycle in-work.
  - A “story point” does not have a standard, well-defined meaning, and changes from programmer to programmer
- Cost Estimators must adapt to agile programs by using estimating methods that capture the flexibility and dynamics of rapid iterations, increased automation, and frequent deliveries of working software features inherent to agile software development techniques