Agile Software Development Cost Estimating

Jim Golden, Unison Cost Engineering

International Cost Estimating and Analysis Association (ICEAA) Workshop, 2024

Abstract

This paper will discuss agile software development cost estimating within the acquisition multi-year planning cycle. 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.

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; and 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 (DOD Agile Software Acquisition Guidebook, Feb 27, 2020).

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. Therefore, focusing on capabilities provides the right level of definition for program planning and development priority. These capabilities should be displayed on the product roadmap (DOD Agile Software Acquisition Guidebook, Feb 27, 2020).

In an Agile program, capabilities are documented by the product owner (with support from the end-user/warfighter) as epics. An epic communicates a high-level need. For example, an epic may be the ability for a customer service representative to create a new customer profile. This capability / epic can be managed on a roadmap in order to help convey where the priority for this capability falls in relation to other capabilities. The epic provides a high-level description of why the end-user/warfighter needs the ability to create a new customer profile, clarifying the "why, where and how" without going deep into implementation detail (DOD Agile Software Acquisition Guidebook, Feb 27, 2020).

Epic *- 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.*

User Story *– The smallest unit of requirements written from a user's perspective of how they will use the software. User stories are defined and prioritized by the Product Owner via backlogs. User stories that cannot be completed within a single sprint should be divided into smaller elements. Each user story should have clear acceptance criteria.*

When it is time for the epic to be worked, For example, the customer profile can have a user story that the customer representative, while on the phone with a new customer, needs the ability to input basic customer definition details which includes capturing the customer's first name, middle initial and last name. The middle initial can be further clarified as an optional field since not all customers have a middle name (DOD Agile Software Acquisition Guidebook, Feb 27, 2020).

The user story also includes capturing 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." The acceptance criteria are applied during test and verification of the user stories. User stories identify the product owner who owns the user story and is responsible for reviewing and accepting (or rejecting) the work (DOD Agile Software Acquisition Guidebook, Feb 27, 2020).

The acceptance criteria are used during testing activities that take place during the development sprint cycles. The same criteria are used when end-user / warfighter (user acceptance) testing is conducted. The acceptance criteria are also reviewed during the sprint and/or iteration demonstrations, during which the product owner and end-user/warfighter see all the work accomplished and have the responsibility to communicate whether it is approved and should be noted as "complete" (DOD Agile Software Acquisition Guidebook, Feb 27, 2020).

Note that during sprint and/or iteration demonstration ceremonies, the product owner may define new requirements and/or changes to priorities. They also see and/or have used the solution allowing them to clarify design preferences and identify options they might not have realized before (DOD Agile Software Acquisition Guidebook, Feb 27, 2020).

Product Backlog

The product owner is responsible for the product backlog. The product backlog identifies what work / capability is in progress and what is expected to be worked. The sprint backlog identifies the user stories that are currently being

worked, as well as user stories that are available for the next one to three sprints. The development team manages the sprint backlog (DOD Agile Software Acquisition Guidebook, Feb 27, 2020).

The user stories identified in the current sprint backlog are noted as planned/in-progress. Only once the work is complete, demonstrated, and has received product owner approval, can the user story be updated/noted as complete. Once all the user stories associated to a product backlog item have been noted as complete, the product capability can also be noted as complete. Updates to the product roadmap can be made based on the product backlog status (DOD Agile Software Acquisition Guidebook, Feb 27, 2020).

Data Gathering (Agile Scrum) for Rough Order of Magnitude (ROM) Cost Estimate

1. 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.
2. Assume that software Size Units in TruePlanning 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 (e.g., planning poker) like the Fibonacci Sequence (1,2,3,5,8,13,20,40) or variations thereof.
3. How many Epics (Releases) are planned for the software development effort?
4. How many User Stories are planned to be completed in each Epic?
5. How many Story Points are assigned to each User Story for the software development effort?
6. How many Sprints are planned for each Epic?
7. 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.
8. 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.
    a. Since agile teams differ across projects, one agile team's velocity may not be the same as another agile team's velocity.

b. An agile team's velocity may be determined after at least three (3) completed sprints on the project. Given the following information pertaining to an agile team's performance on the first three (3) sprints for a notional project:

   i. Sprint 1 = 16 story points completed

   ii. Sprint 2 = 15 story points completed

   iii. 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.

9. 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
10. How many Agile Software Development Team members/developers are there, besides the Product Owner and Scrum Master?
11. What programming language(s) will be used in the effort? (Note: In the TruePlanning application, a different Agile Software Component cost object will be used for each different language).
12. Note: for a ROM cost estimate in TruePlanning, assume 100% New Code.

Example: Software Application - ROM

Assumptions (for this project example in the TruePlanning application)

1. A Sprint is a time box of two weeks (80 hours) during which a done, usable, and potentially releasable product increment is created.
2. Set Agile Software Development Team members/developers, besides the Product Owner and Scrum Master, to seven (7).
3. Velocity is defined as the number of story points an agile team can complete in a sprint.
   a. Since agile teams differ across projects, one agile team's velocity may not be the same as another agile team's velocity.

b. An agile team's velocity may be determined after at least three (3) completed sprints on the project. Given the following information pertaining to the agile team's performance on the first three (3) sprints for of project:

   i. Sprint 1 = 16 story points completed

   ii. Sprint 2 = 15 story points completed

   iii. 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.

4. 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
5. 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. 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
   b. Velocities must be determined for user stories that will be coded with specific programming languages.
6. Assume that the team velocity is 16 story points/sprint, regardless of which programming languages will be used.
7. The programming language for all user stories will be "C".
8. The sprint length is 2 weeks (80 hours). Assume 100% New Code.
9. 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

10. This results in a total of 48 story points that the agile team must complete.
11. The Product Owner determines that the highest value stories will be completed as soon as possible given the velocity (capability) of the agile team.
12. Sprint 1 will have the agile team complete:

    User Story A – 5 SP User Story D – 5 SP User Story E – 6 SP

13. Sprint 2 will have the agile team complete:

    User Story B – 6 SP User Story F – 5 SP User Story G – 5 SP

14. Sprint 3 will have the agile team complete:

    User Story C – 6 SP User Story H – 5 SP User Story I – 5 SP

15. Reference item 3 above. 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. In TruePlanning, for all Agile Software Component cost objects, we must change the Size Units in the input sheet (L11) to ”Custom Size Units”; and using the Custom Size Unit Name calculator (L12), input the New Size Unit Name as “Story Points”. We must also choose a “Relative Size Unit”, so choose SLOC. Enter the “Size Conversion Factor” as 9.90.
    a. The inputs essentially tell TruePlanning that we will be using story points as a sizing measure, and that 1 story point will be equal to 9.9 source lines of code. This will result in TruePlanning estimating 5 hours of labor effort for one story point.
    b. Use one Agile Software Component cost object in each sprint (which are “excluded” in the example PBS) to do this “calibration” of a story point to number of hours of effort.
16. Next, populate the PBS with Sprints and associated User Stories.
17. Populate the New Size in L13 of the input sheets 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!
18. Set Length of Iteration or Sprint (L47 of detailed input sheets) to 2 weeks.

19. Initial setup for a ROM is complete, notwithstanding the changes needed for operating spec, labor rate burdens, life cycle dates, escalation, etc.

20. Results:

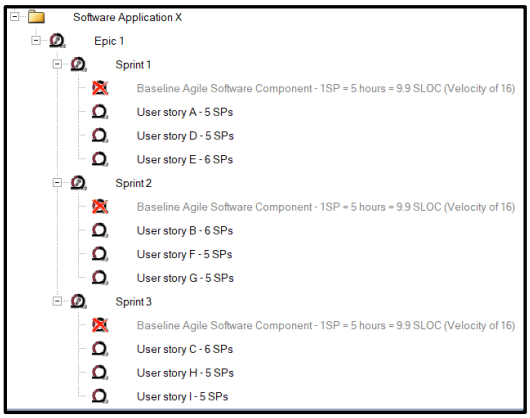

Figure 1. Example Agile Software Development Product Breakdown Structure (PBS)

While there are many Agile frameworks, taking the time to understand the process used to develop the software using agile processes, and gathering the appropriate data, is critical to set the pace (velocity) of the work and ultimately determine the cost of the development effort. Saving actuals from current projects will inevitably lead to enhanced future cost estimates.

References

1. Agile Scrum Crash Course: A Guide to Agile Project Management and Scrum Master Certification PSM 1. Umer Waqar, 2020.

2. Agile Software Acquisition Guidebook; Best practices & lessons learned from the FY18 NDAA; Section 873/874 Agile Pilot Program. https://www.dau.edu/sites/default/files/Migrated/CopDocuments/AgilePilotsGuidebook V1.0 27Feb20.pdf