Why Your Software Cost Estimates Change Over Time and How DevSecOps Data Can Help Reduce Cost Risk

Julie B. Cohen

September 2023

Introduction

Program managers (PMs) must realize that early estimates are likely to be off by over 40 percent and that programs need to continually update estimates as additional information becomes available. It is important to understand how program risks can impact cost estimates. Tracking items—such as the decisions that have not yet been made, stability of the top capabilities needed, and measurements derived from the DevSecOps pipeline—can all help program management offices (PMOs) better understand program uncertainties that can impact estimates and help increase confidence in estimates over time.

DoD Program Estimation Background

Department of Defense (DoD) programs are required to perform cost analyses at various stages of the program lifecycle. For Acquisition Category I (ACAT I) programs, an independent cost estimate is required for each milestone review. These estimates must be approved by the director, Cost Assessment and Program Evaluation (DCAPE), and their fidelity should increase at each successive review as program uncertainties resolve over time. New uncertainties may unfold as the program progresses, but in general, overall uncertainty should progressively be reduced.

When discussing software cost estimates, it is important to remember this quote by Steve McConnell: "*The primary purpose of software estimation is not to predict a project's outcome; it is to determine whether a project's targets are realistic enough to allow the project to be controlled to meet them.*" A project should not be a "random walk;" instead, it should be a walk with a sequence of course corrections. A sound estimate assures that the outcome is achievable with the available time and resources.

The cost estimation process typically starts during an analysis of alternatives (AoA), which is performed during the Materiel Solution Analysis phase leading up to Milestone A. An AoA includes a lifecycle cost baseline for each alternative, the lifecycle cost per unit system, and the lifecycle cost per specified quantity of systems. The AoA is typically performed under contract. The program office typically has a cost section staffed by a combination of military, civilian, and contactors. The program office is ultimately responsible for the program office estimate (POE).

Typically, multiple contractors submit proposals for the Technology Maturation and Risk Reduction phase. Estimates derived from these proposals are often used to further refine the POE.

All the POEs should follow the guidance from the *DoD Cost Estimation Guide* [DoD 2020]. This guide calls for the estimation to account for risk:

> *A risk is a potential future event or condition that may have a negative effect on cost, schedule, and/or performance. An opportunity is a potential future event or condition that may have a positive effect on cost, schedule, and/or performance. Risk/opportunities have three characteristics: a triggering event or condition, the probability that event or condition will occur, and the consequence of the event or condition should it occur. Analysts often use the terms risk and uncertainty interchangeably. In fact, they are distinct from one another. Uncertainty is the indefiniteness of the outcome of a situation. Uncertainty captures the entire range of possible positive and negative outcomes associated with a given value or calculated result. In a cost estimating model, an analyst generally addresses uncertainty first. The analyst then addresses risks/opportunities if and only if the uncertainty assessment has not already captured them.*

Another way to look at risk in cost estimates and the changes in estimates as a program progresses is using a framework called the Cone of Uncertainty. The first use of the actual term *Cone of Uncertainty* for software was by Steve McConnell is his book, *Software Project Survival Guide* [McConnell 1997], but the concept of the changing uncertainty ranges was first used in 1958 by J.M. Gorey in an Association for the Advancement of Cost Engineering (AACE) paper [Gorey 1958] and was later used by Barry Boehm as the “funnel curve.” Chris Adams provides a good summary of this concept on the web [Adams 2023]; we discuss this concept in the next section.

The Cone of Uncertainty Background

The Cone of Uncertainty is a term often used in project management that describes the phenomenon by which project unknowns decrease over time. The Cone of Uncertainty framework is used in software estimation to determine the most likely outcome; Chris Adams describes it as follows [Adams 2023]:

> *As the project proceeds and more research and development is completed the amount of uncertainty decreases, eventually approaching zero. Project unknowns, or uncertainty, largely correlate to variances in project estimates. Plotting these variances over time creates a cone or funnel shape (variance percentages shown are only examples, values may vary).*

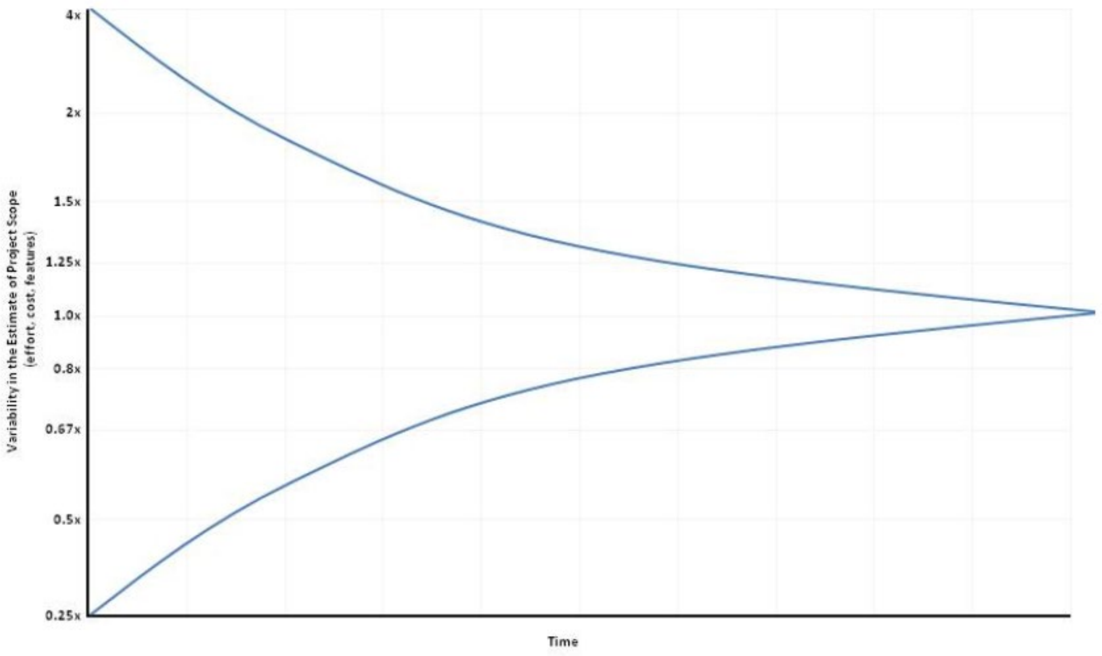

Figure: Cone of Uncertainty (This file is made available under the Creative Commons CC0 1.0 Universal Public Domain Dedication File:Cone of Uncertainty.jpg - Wikimedia Commons)

The Cone of Uncertainty and Iterative Development

Applying the Cone of Uncertainty to iterative projects is somewhat more involved than applying it to sequential projects.

If you are working on a project that completes a full development cycle in each iteration (i.e., requirements definition through release), then you will go through a miniature cone during each iteration. Before you do the requirements work for the iteration, you will be at the Approved Product Definition part of the cone, which is subject to 4x the variability from high to low estimates.

In short iterations (less than a month), you can move from Approved Product Definition to Requirements Complete and User Interface Design Complete in a few days, which reduces your variability from 4x to 1.6x. If your schedule is fixed, the 1.6x variability applies to the specific features you can deliver in the time available rather than to the effort or schedule.

Although there are many uncertainties that affect predictability, requirements flow down to design and implementation decisions. Approaches that delay a full requirements specification until the beginning of each iteration also delay narrowing the cone of uncertainty—with respect to cost, schedule, and feature delivery—several iterations down the road. It is difficult, after all, to know when your project will be done if you do not at least specify what *done* looks like. Your program might highly prioritize flexibility, or it might prefer projects with more predictability.

Many development teams settle on a middle ground between flexibility and predictability in which a majority of requirements are defined at the front end of the project, but design, construction, test, and release are performed in short iterations. In other words, the project moves sequentially through the User Interface Design Complete milestone about 30% of the calendar time into the project, and then shifts to a more iterative approach from that point forward. This approach drives down the variability from the cone to about ±25 percent, which allows for project control that is good enough to hit targets while still tapping into the major benefits of iterative development.

Project teams can leave some amount of planned time for as-yet-to-be-determined requirements at the end of the project. Doing that introduces some minor variability related to the feature set, which, in this case, is positive variability because you will exercise it only if you identify new desirable features to implement. This middle ground supports long-range predictability of cost and schedule as well as a moderate amount of requirements flexibility [Construx 2023]. Even when using this method, unless there is a large user-driven change in the capability needed for the project, the requirements volatility should decrease over time.

Risk and Uncertainty

Glen Alleman stated the following about uncertainty [Alleman 2018]:

> *Uncertainty comes from the lack information to describe a current state or to predict future states, preferred outcomes, or the actions needed to achieve them. This uncertainty can originate from random naturally occurring processes of the program (Aleatory Uncertainty). Or it can originate from the lack of knowledge about the range of future outcomes from the work on the program (Epistemic Uncertainty).*

Aleatory uncertainty can be thought of as common cause variation that is natural to the system. Epistemic uncertainty results from an incomplete understanding or characterization (e.g., a lack of understanding the range of natural variation or incomplete requirements). Finally, ontological uncertainty can be thought of as unknown-unknowns. Whereas epistemic uncertainty represents an incomplete understanding of something we know of, an ontological uncertainty appears as a complete surprise. Ontological uncertainty is often a form of special cause variation that was not anticipated or precedented.

Common cause variation cannot be specifically reduced through management action; instead, it requires technical change. However, in Agile development, several approaches are commonly used to reduce epistemic uncertainty (incomplete knowledge). Frequent increments and product demonstrations allow feedback from both the users and the development process. Feedback with an incomplete product allows unforeseen uses, requirements, or component interactions to be discovered. Development spikes are designed to uncover information (e.g., about performance).

Sequencing work such that important but uncertain features and capabilities start earlier, not only enables using what was learned in refining and developing those capabilities to “buy down” overall uncertainty, but it also reduces the remaining uncertainty later in the program when there is less opportunity to recover. Taking any or all of these actions early in a program can help improve the accuracy of the overall cost estimate and reduce risk exposure.

The level of risk and cost estimation uncertainty can also be viewed from the perspective of the lifecycle phases of a system's development. Even in agile development, it is important to understand the top-level capabilities needed at the start of the program. In the Software Acquisition Pathway, these are described in a capability needs statement [DAU 2023]. These capabilities may change based on operational needs, but requirements changes may increase risk and can lead to increased cost if other capabilities are not swapped out.

Another important aspect to consider for reducing risk is to focus on the architecture early in the program. Ensuring the architecture is suitable for both the functional and non-functional requirements can help ensure that large-scale architecture changes will not be required later in the program. A facet of DevSecOps that can reduce risk is early integration and automated testing. The earlier you start integrating and testing code, the sooner any issues or defects can be found. This approach can also reduce overall risks and increase the confidence in cost estimates.

Some areas to consider when determining your estimation uncertainty include

1. the decisions that have been made (e.g., reuse, computer languages, architecture)
2. what has been discovered (i.e., known unknowns and unknown unknowns)
3. the overall scope and amount of requirements growth
4. what can be measured to help understand how much uncertainty remains

How Metrics from DevSecOps Pipeline Data Can Help Reduce Estimation Risk

Metrics can help the program management team better understand and estimate program status and risks. Although measurement does not by itself reduce risk, measurement informs decisions that can reduce risk. Some metrics can originate from contract data requirements list (CDRL) deliveries, such as an earned value management (EVM) report or a metrics report. CDRLs typically provide data from the last one or two months. In modern software development, data obtained from the DevSecOps environment can provide real-time, helpful information.

A few metrics you can obtain through the pipeline to better understand and reduce estimation uncertainty include the following:

1. **Completion Rate Volatility**: An example of measuring completion rate volatility is measuring changes in sprint velocity to detect uncertainty. If teams properly estimate their sprint velocity and consistently work at the estimated rate, then overall uncertainty can be reduced because you have more confidence that the epistemic uncertainty has been quantified as a common cause variation and special cause (ontological uncertainty) variations can be identified and addressed. Likewise, teams that start with "low hanging fruit" may gain a false sense of confidence unless they recognize that completion rates closely match the actual progress for easier tasks.

2. **Capability Development Progress**: Your pipeline can provide data on how many capabilities are fully developed, how many are in progress, and how many remain in the backlog. When working in program increments, it is possible for capabilities to remain incomplete and for predecessors to create dependencies in other areas. Understanding how capability development compares to the plan can help reprioritize work so that key capabilities reach completion. This approach can also help you better understand what uncertainty may remain in your estimates.
3. **Software Quality**: Your DevSecOps pipeline should include static and dynamic code scanning tools. These tools, along with counts of defects discovered during testing, allow the PMO to better understand aspects of quality that can cause (1) delays in testing, (2) rework, or (3) operational failures. Good quality software not only requires less time to correct errors rather than produce a new product, but it also increases confidence in the estimates' accuracy and precision.
4. **User Acceptance**: One of the main tenets of agile development is user involvement. If users are regularly involved in end-of-sprint and/or end-of-increment demonstrations, then the PMO can gain an early understanding of how users are reacting to the capabilities being developed. If the user reaction is positive and the number of changes requested is in line with the estimated work, then this can also increase confidence in the initial estimate. On the other hand, unplanned rework can result in additional costs and delays, unless other work is removed to allow the new work requested by the user to take priority within the schedule and budget.
5. **Organization and Staffing**: Using data from tools such as Confluence and Jira, a PMO should be able to see the development organizational structure and the number of people on each team to help understand if the project is fully staffed. Changes can also be tracked to understand staffing volatility. Staffing volatility, in turn, leads to a need to recalibrate team velocity estimates, thus increasing uncertainty until new baseline data is available. If the project is fully staffed with an organizational structure that supports it, then this can reduce estimation risks. If the project is slow to staff up or never reaches the full staffing profile, the estimate must be adjusted to reflect this.

More information about using data from the DevSecOps pipeline is available in the white paper *Program Managers—The DevSecOps Pipeline Can Provide Actionable Data* [Cohen 2023].

The Bottom Line

The primary takeaway from this paper is that early estimates are likely to be off by over 40 percent, and programs need to continually update estimates as additional information is available. Tracking items, such as what decisions have not yet been made, the stability of the top capabilities needed, and measurements derived from the DevSecOps pipeline, can all help the PMO better understand program uncertainties that impact estimates and help increase the confidence in estimates over time.

References

URLs are valid as of the publication date of this paper.

[Adams 2023]
Adams, Chris. What Is the Cone of Uncertainty? *Modern Analyst Website*. September 14, 2023 [accessed]. https://www.modernanalyst.com/Careers/InterviewQuestions/tabid/128/ID/2324/What-is-the-Cone-of-Uncertainty.aspx

[Alleman 2018]
Alleman, Glen. What Is Risk? *LinkedIn*. March 10, 2018. https://www.linkedin.com/pulse/what-risk-glen-alleman-mssm/

[Cohen 2023]
Cohen, Julie B. & Nichols, William Richard. *Program Managers—The DevSecOps Pipeline Can Provide Actionable Data.* Software Engineering Institute, Carnegie Mellon University. April 2023. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=890538

[Construx 2023]
Construx Software. The Cone of Uncertainty. *Construx Software Website*. September 14, 2023 [accessed]. https://www.construx.com/books/the-cone-of-uncertainty/

[DAU 2023]
Defense Acquisition University (DAU). Software Acquisition. *DAU Website*. September 2023 [accessed]. https://aaf.dau.edu/aaf/software/

[DoD 2020]
Department of Defense Office of the Secretary of Defense. *DoD Cost Estimating Guide, Version 1.0*. DoD. December 2020.
https://www.cape.osd.mil/files/Reports/DoD_CostEstimatingGuidev1.0_Dec2020.pdf

[Gorey 1958]
Gorey, J. M. Estimate Types. *AACE International website*. April 17, 1958.
https://library.aacei.org/pgd01/pgd01.shtml

[McConnell 1997]
McConnell, Steve. *Software Project Survival Guide*. Microsoft Press. 1997. ISBN: 978-1572316218. https://www.microsoftpressstore.com/store/software-project-survival-guide-9780735691483

Contact Us

Software Engineering Institute
4500 Fifth Avenue, Pittsburgh, PA 15213-2612

Phone: 412/268.5800 | 888.201.4479
Web: www.sei.cmu.edu
Email: info@sei.cmu.edu