Software Cost Estimating – Friend or Foe (to Agilists)

Carol Dekkers, PMP, CFPS (Fellow), CSM, P.Eng.
President, Quality Plus Technologies, Inc.
dekkers@qualityplustech.com

Abstract

Software cost estimating – the mere mention of the term elicits a myriad of polarized opinions, and like most other topics associated with software development today, it has become a maelstrom of self-appointed experts who weigh in with commentary. As a software engineer with years of development, project management and software measurement experience, I had the good fortune to be contracted as the Lead Author for the Software Cost Estimating Body of Knowledge (CEBoK-S) from 2019-2020 and I worked with some of the worlds leading experts in software cost estimation. I anticipated the role to be challenging, however, looking back at the project, it was a "drinking from the firehose" experience that illuminated that I had no idea how involved and complex professional cost estimating really is. I didn't know what I didn't know.

In my role to develop CEBoK-S I was armed with experience and expertise as a world-recognized software measurement expert, and project manager. I thought I knew what software cost estimating was all about, at least the basics, and in the beginning, I held a similar opinion to that of one of the reviewers of my first drafts of this paper: "I do think Estimates are a very basic discipline that you simply can't do without."

I no longer hold that opinion and I realize it was based on a lack of knowledge about professional software cost estimating. If you gain one message from this paper, I hope it is that "software cost estimating is a profession worthy of a seat at the management table and deserving of the same (or better) respect and status of equally experienced software development and project management professionals." In May of 2023, ICEAA released the first version of the official CEBoK-S together with an associated Software Cost Estimating Certification (SCEC.) I am extremely proud of the product and of the role I played in its development. Software Cost Estimating is both necessary and integral at the beginning and throughout software development – not as a single pie-in-the-sky activity to be performed when management initiates a project. Software cost estimating, done right, involves skills in statistical analysis, data mining, business analysis, formal methods, identification, and quantification of areas of risk and uncertainty, developing appropriate cost estimating relationships (CERs) based on normalization and data analysis to piece together often inconsistent and incomplete historical data. To the team of professional cost estimators who patiently educated and work with me to develop CEBoK-S, I am grateful and hold the utmost regard. Hopefully after reading this paper and attending my presentation, you'll gain an appreciation for the professional and necessary role that professional software cost estimators play in our software development industry.

Biography

Ms. Carol Dekkers is a Certified Scrum Master (CSM), a Certified Function Point Specialist (CFPS-Fellow), a Professional Engineer (P.Eng-Canada) and a Certified Software Cost Estimator (SCEC). She is also the lead author of the International Cost Estimating and Analysis Association (ICEAA)'s new Cost Estimating Body of Knowledge for Software (CEBoK-S), and a long-standing member of the U.S. delegation to the International Organization for Standardization (ISO) subcommittee for writing Software and Systems Engineering standards. Ms. Dekkers expertise spans software development, software measurement, quality engineering, project management, and software cost estimating and has shared her insights with technical and non-technical professionals worldwide through keynotes/presentations, textbooks, and articles published in industry journals. Ms. Dekkers has received numerous awards for her industry contributions and thought leadership including Computing Magazine Global Leader in Consulting – Pro Bono (2023), ICEAA Educator of the Year (2022), IFPUG Honorary Fellow (2022), Brazil and Korean metrics associations special awards (circa 2010) and was named one of the 21 New Faces of Quality for the 21st Century by the American Society for Quality, ASQ (2000.)

1. Introduction

Software projects are different from hardware or construction projects in that they are human-centric, creative, and technical endeavors that bring software ideas to life, and for which there are many uncertainties – especially early in the development process. At the time when early estimates are required, software requirements may not yet be articulated or understood. Software, while often considered an intangible product, follows a development path involving stakeholders (many), processes, rework, incomplete requirements, and other changing factors that complicate creating accurate estimates. Dr Barry Boehm, the inventor of the Constructive Cost Model (COCOMO) at the University of Southern California introduced the concept of a "cone of uncertainty" to illustrate how early software estimates can be inaccurate by up to +/-4X depending how early in the development life cycle" they are created.

With agile software development, projects commence with only minimal requirements specificity, and the range of estimating uncertainty could be even greater. Software estimating approaches typically include adjustments to account for anticipated project variations, but agile software developers believe these variations are so marked that cost estimating should be abandoned altogether. Such was the impetus for the "no estimates" movement. Today ,cost estimators and agilists often remain at odds. On one side, estimators are tasked to create software cost estimates from early requirements (documents lack specificity,) while on the other side, agile software developers hesitate to participate in preparing estimates that they feel will become budgets, schedules, and targets.

This paper acknowledges that agile software development introduces added uncertainties and challenges to software cost estimating and suggests recommendations to formalize the role of software cost estimators in creating more realistic estimates. More realism in cost estimates should lead to more realistic budgets and timelines for software development contracts and hopefully position all software development projects for greater success.

In 2020 when I signed the contract as Lead Author for the CEBoK-S for the International Cost Estimating and Analysis Association (ICEAA), I envisaged my task as fairly routine – I'd be starting off with an established curriculum from the 5-day US Department of Defense Acquisition University (DoD DAU) software cost estimating curriculum and "simply" remove DoD references, and update five year old content to current best practices. During the process, I worked with some of the top professional software cost estimators in the US, as well as participants from abroad. I am humbled that my original opinion of software cost estimation actually mirrors the shared an opinion with one of the reviewers I do think Estimates are a very basic discipline that you simply can't do without of this paper who stated: "." I was so wrong!
And so, it is with the majority of software-savvy developers and project managers who share my original opinion.

Whether or not you agree with cost estimates and the myriad of opinions about their relative merit or damage, or that accompany Cost estimates are a necessary first step in the acquisition process for funding large scale software development efforts, regardless of the development approach. Software is pervasive in every aspect of our lives from Smart homes with remote controlled appliances to Self-driving cars to Smart highways. Even on projects where software is not dominant (such as a toll-highway construction project,) delays in the software development can have a major impact. In addition, the volume of software "code" now involved in major programs such as the F-15 military aircraft program has increased so exponentially that software (once a minor component of the product) is now as costly and dominant as hardware and other product components.[1]

In 2001, ***The Agile Manifesto[2],*** written by seventeen prominent software developers, revolutionized the approach to developing software and presented guiding principles to both streamline software development *and* increase the value of delivered software, among others. The change to how software is developed minimize the upfront, protracted requirements definition of the earlier waterfall approach, and allows the software scope to evolve iteratively instead of monolithically. This means that software product development today does not come about using a fixed scope of software requirements, but rather through incremental "deep dives" into the software requirements as prioritized by the product owners (the business.) What has not changed, however, are the needs for advanced, high-level cost and schedule estimates for software development often prepared as much as 2-3 years in advance. As such, software estimates remain a non-negotiable requirement for agile and non-agile projects alike. This creates friction within the

[1] Software Cost Estimating Body of Knowledge (CEBoK-S) Lesson 1: Motivation for Software Cost Estimating, 2020.
[2] **The Agile Manifesto**, aka **The Manifesto for Agile Software Development**, Beck, Kent, Mike Beedle, Aire van Bennekum, Alistair Cockburn, et al, 2001, https://agilemanifesto.org/

agile development community and resistance to the entire concept of estimates by agile software developers who argue that estimates of product development are not possible based on the creative nature of software development, couple with a lack of scope or detailed requirements.

At the same time, the ongoing definition of project success according to the annual Standish Group CHAOS[3] report continues to be those projects delivered "on-time," "on-budget," and meeting the user needs (based on estimates.) The degree of success for agile software development projects has improved over the traditional waterfall (linear, fixed) approach to building software, however, despite the approach, less than 50% of projects meet the criteria to be deemed successful. There are many reasons that software projects exceed their budgets and schedules including management issues, development delays and handoffs to multiple teams, development process inconsistencies, supply chain delays, over-optimism, and others.

In the 25+ years of CHAOS reports, improvements to overcome and improve project success have focused on the software development processes and the management-related reasons for project "failure" assuming that the project budget and schedule were solid. (The reasons most frequently addressed include a lack of customer-centric processes and tools, management support, and user engagement.) The assumption that the budget and schedule were appropriate in the first place assumes that the underlying project estimates were sound from the get-go, yet in this author's opinion, poor software cost estimates may be a fundamental cause of under-budgeting and overly optimistic schedules.

Focusing on the appropriateness of the contracted budget and schedule implies that the underlying estimates were sound, yet during this author's research as lead author for the international software cost estimating body of knowledge highlight that overly optimistic estimates and immature estimating processes may be at least partially at fault for at least some project failures.

This paper and the accompanying presentation answer such questions as:

- Why do software projects never seem to finish on time and on budget? (And why it is not the agile developer's fault.)
- How can developers help to create better cost estimates?
- Why do cost and schedule estimates take so long to prepare when all the data are right there in their tools?
- Why are formal cost estimating approaches beneficial in today's development environment?

For software development to succeed in meeting project goals of on-time and on-budget delivery, the software cost estimates on which they were based should be reliable, realistic, and data-founded. An exploration of how to professionalize and formalize software cost estimating follows, with simple steps for agile software development teams to productively engage with cost estimating teams. The time is now for software developers and customers to embrace software cost estimating as a critically important profession and engage to guarantee project success for the future.

2. Background

Software development is young compared to other product development industries such as construction. In the 1950's software was designed to automate manual business practices and to automate the processing of large volumes of data using written computer "code" or programs, but without standardized software development life cycle processes. Over the decades, technological advancements, and the application of automation to more industries added complexity to software development and a variety of sequential, formal approaches were used.

In 2001, a group of developers who realized that software requirements often changed after the design phase, wrote the Agile Manifesto which espoused a dozen customer-centric concepts such as valuing working software over documentation, and people over processes. Thus, the Agile Software Development movement was born. Today, software development is a multi-billion-dollar industry touching every aspect of human life from deep sea exploration and space travel to self-driving cars and smart highways. The software development landscape today includes both waterfall (sequential, adverse to change) and agile (evolving scope) approaches, with the majority of large scale government agency software development embracing a hybrid approach to software product development that can include Commercial Off The Shelf (COTS) software packages, custom software development, glue code (to put the pieces together), software as a service (think rented software) and configured enterprise resource planning (ERP) implementations. Regardless of the specific project, two major cost drivers often dominate software development costs: the size of

[3] **CHAOS Reports,** Standish Group, annually 1994-2020 https://www.standishgroup.com/sample_research_files/chaos_report_1994

the software and the productivity (complexity of the software, skills of the team, and tools) to complete the project. Software and its sustainment are now a major cost center for businesses worldwide.

As a case in point, during the winter of 2022, the US-based carrier Southwest Airlines suffered major business loss due to flight disruptions caused by out-of-date software (a business decision.) The software itself was not SWA's major business, however, it supported (and then did not) their business of getting people from point a to point b during a busy holiday season. Countless other businesses are disrupted or seriously challenged by software issues. This is peripheral to the point of software cost estimates; however, it serves to demonstrate how integral software is to our daily life.

The status quo of software development

The biannual Standish Group CHAOS reports tracked IT project success, challenges, and failures by defining SUCCESS as on-time and on-budget deliveries that met customer requirements. Since the first report in the early 1990's, the success rate has fluctuated between 30 and 42%, the latter in the last year of the formal report in 2020 for agile projects. Challenged projects were those that were either over-budget or behind schedule, while outright failures included canceled projects or projects completed but over-budget, late and/or not meeting customer requirements.

Additionally, the average cost growth of software development projects hovers around 40-50% and occurs on 80% of development projects, while the average schedule delay for the same projects is between 60-80% and occurs 90% of the time, according to Dr Christian Smart in his 2021 book ***Solving for Risk Management: Understanding the Critical Role of Uncertainty in Project Management.*** It is plain to see how overly optimistic and unrealistic (wishful thinking) software cost and schedule estimates a major root cause of software development "failure" can be regardless of the development approach.

Why do projects never seem to finish on time or on budget (and why it is not the development team's fault)

Even though project success or failure is based on on-time and on-budget project performance, most of the attention for process improvement of challenged or failed projects was focused on process and management causes. Software process maturity, development tools and methods, and management practices are all amongst the improvement areas, however, the research assumed that the estimates were realistic and accurate in the first place. In actuality, late projects and cost overruns may be attributable to poor estimates produced by immature estimating practices. over-optimism, and a lack of or ignorance of good historical data. Added to the known "Cone of Uncertainty" where preliminary software development estimates can vary by as much as +/- 400% with actuals, it is surprising that software projects hit their targets even 30% of the time.

According to the Standish Group, unrealistic estimates resulted in $81B USD in cancelled software projects, and $59B USD in budget overruns (2015 Standish Group CHAOS report, one of the last publicly available reports.) There are many reasons that software projects overrun their budgets and schedules and are worthy of discussion, however, this paper focuses on issues related to improving the underlying estimates on project success or failure are based. .

The good and bad news about status quo software cost estimating

The good news if you are a member of an agile team on a project that is over-budget and/or behind schedule, if the original project estimates were unrealistic from the beginning, the fact that the project is "challenged" is not your fault.

The bad news is that until software cost estimating is embraced as a bona-fide and formal professional endeavor, and the processes to collect historical data (actuals) are improved, software development is trapped in a tornado type cycle similar to a dog chasing its tail:

1. **Few strong inputs & low estimating maturity.** Poor requirements documents and apathy (or downright disdain for estimating) on the part of the software development team are poor inputs to the estimating process. This leads to
2. **Weak, and unreliable estimates.** Immature software cost estimating practices (ad hoc, non-data-founded, non-standard processes) together with lack of good data lead to overly optimistic and unrealistic software size and effort estimates. This leads to
3. **Unrealistic plans (cost, effort, duration).** Lack of standardized estimating process (ground rules & assumptions, cross-checking, verification by the development team) leads to

4. **Impossible contracts / internal projects.** Development begins based on flawed contract values (and undocumented) assumptions and both the customer and supplier seek to make changes/clarifications to lower their risk, which leads to
5. **Out of control projects.** Both sides end up at odds (there is no win-lose in software development – only win-win or lose-lose) and
6. **Adverse project outcomes:** uncontrolled project "growth," failed & cancelled projects, litigation, Wasted investment. Project is cancelled, restructured, descoped, or failed. Then the actuals data are not collected consistently. Few lessons learned, until…
7. **A new project emerges /or cancelled project restarts at step 1**, with minor change except for the promises/optimism to do better. The cycle continues with the same well-intentioned, but unrealistic estimates based on flawed assumptions and poorly documented software requirements.

Certainly, just improving the software cost estimating process will not guarantee project success – estimates are simply best-guesses of how the project could go if all inputs and assumptions come true. Good estimates at least position a software development effort in the running for success.

Software cost estimating is the subject of at least a dozen major US Federal Government guidebooks and courseware including works by the Government Accountability Office, GAO (2021) who produced two guides, one on cost estimating (with a section devoted to software costing) and one on agile software development (featuring sections on cost estimating), the Department of the Navy, Homeland Security, Defense Acquisition University (DAU) and others, and of commercial consulting organizations. In addition, there are several parametric cost estimating tools used successfully worldwide to estimate cost and schedules for software intensive systems development. GAO's 2021 manual outlined a series of challenges to cost estimating and recommended some mitigating factors, including a lack of high-quality software development data.

At this point with the lack of requirements specificity (software size) and the number of uncertainties involved, especially in agile development, one might wonder why bother spending time creating an estimate (that will be wrong) in the first place? The answer is that there is often no choice about whether to estimate or not to estimate. For US Government software acquisition at least, in order to advance any sort of software development program, some form of cost and schedule estimates are a requirement. While the type of estimate and the specificity of content varies based on the type, timing and size of the project, there are few programs that do not require a software cost estimate. The same is true for commercial enterprises – software cost estimates are often needed two to three years ahead of the budget cycle in order to be included in potential funding. As such, the question is not so much about Why both estimating, but rather, how can we create credible and realistic software cost estimates that will be beneficial to the software development projects?

3. What I learned about Software Cost Estimating (that I didn't know I didn't know)

The following subsections comprise the author's top ten lessons learned about software cost estimating: (Note, these points are all outlined in the PowerPoint® presentation version of this paper.)

1. **Data-founded estimates** are more defensible than theory (or expert opinion.) If you don't have historical data, seek out publicly available data (such as those found in commercial estimating tools or in such databases as the International Software Benchmarking Standards Group's (ISBSG) development and enhancement repository), use a Wide Band Delphi expert opinion approach (whereby subject matter experience/expertise are relied on in lieu of data), remember the cone of uncertainty and check for published industry rules of thumb. A defensible estimate is like a "line in the sand" and provides a backdrop on which the estimate has merit. In Agile development, opt for T&M (time and materials) contracts with flexibility for change. (Refer to the *Acquisition & Management Concerns for Agile Use in Government Series: Estimating in Agile Acquisition by the Software Engineering Institute.)*
2. There is a published **Cost Estimating Maturity Model** illustrating that Software Cost Estimating is a professional endeavor (and is non-trivial.) Most organizations (worldwide) start out at Level 1 and do not have a formal approach to software cost estimating.
3. **Estimation scope** is critical to the context of the estimate
4. There is **no "One Size Fits All"** software cost estimating approach. The estimating approach depends on the purpose, timing, and type of estimate as well as available data. Estimating involves more than estimating a size and then multiplying by a magic or historical "productivity" rate.

5. **Quantifying Software Size** is fundamental to a good parametric Cost Estimating Relationship (CER). This may be argued by supporters of the bottom-up-WBS task oriented estimating approaches. Not to disregard the value of task or activity-based estimating, but size is a known cost driver. Consider the construction of a building – the cost per square foot should be a consideration at least as much as a general contractor's estimate to frame a building based on how quickly crews can work. Cross-checks with multiple estimating methods improves the estimate.
6. **Software size can be estimated using standard units of measure (UOM)** and based on ConOps/EPICS/prelim backlog or other early requirement documents
7. **Software development cost is non-linear (S-curve) and subject to Diseconomies of Scale (EXP > 1)** Note: This is different from hardware estimating and other product development where volume of widgets comes into play. Analogous (linear) estimates are possible only within a given range of application size.
8. The software development approach (waterfall or agile) brings in **different cost considerations,** as well as whether the development is done in-house or by third-party vendor(s).
9. Most software development is a mix of **hybrid solutions, necessitating multiple estimates.** The type of development, availability of technical baseline data, historical data, level of quality required in delivered software, etc. should all be considered when preparing and presenting the estimate. Beyond software development, consider the sustainment, integration, deployment, and other factors to create and provide the context for a realistic estimate. Hybrid software solutions often necessitate multiple software cost estimates.
10. The **software development versus procurement continuum is often overlooked** and can impact the development of a realistic estimate. Additionally, the level of leadership required can impact costs (bigger programs need more leadership - e.g., running a large software development program that involves systems of systems is impossible without a strong leadership team to guarantee the delivery of quality software.)

4. Ideas and next steps (based on today's environment)

As an agile tester or agile developer, collaboration with the software cost estimating team is needed to ensure that the software estimates (4 parts) are reasonable, realistic, inclusive, and presented in context. This must be a joint effort that starts with the project initiation documents (such as a Concept of Operations: ConOps) through to estimating through to development release strategies, contracting, development and change management, and culmination in the software release(s.) Data collection and maintenance of a repository are also collaborative efforts where the quality team and project management organization (PMO) should play a lead role.

4.1 Consider the benefits of better software cost estimates on your project success and how to build the competency in your company

There is a certification: SCEC, Software Cost Estimating Certification, and a complete set of formal documentation for the ICEAA Cost Estimating Body of Knowledge for Software (CEBoK-S.) The documentation covers the aforementioned lessons learned and also the following major concepts:

- Types of software cost estimates and considerations (Lifecycle cost estimate, ROM, software development estimate, software sustainment, risk factors, etc.)
- How to select the best approach for doing a software cost estimate
- Important concepts: estimating maturity model, size, productivity, ground rules & assumptions. Data normalization and analysis
- Steps to preparing a good estimate
- Cross-checking, context, presentation to management
- Ten modules of materials plus backup slides (over eight hundred slides)

5. How can Cost Estimators and Agile Software Teams work together to achieve success

Given that software cost estimates remain a requirement for acquisition and software development funding, it is critical that the best possible, data-founded, and realistic software estimates are developed.

While it might seem that a software cost estimate would be a range of numerical values representing the estimated software development size, effort, cost, and schedule (duration), a good software cost estimate should consist of much more.

As a guideline, a software cost estimate should contain the following information:

1. **Contextual information** about the software project/release/phases included and:
    - Program/project/release identification (name);
    - Description of same
    - Type(s) of cost estimate(s) and scope of each;
    - Source documents and subject matter experts included in the formulation of the estimate;
    - Basis of estimate;
    - Ground rules & assumptions;
    - Technical baseline;
    - Scope of the estimate (e.g., software development, procurement, sustainment, maintenance, and any other inclusions. Explicitly state exclusions from the estimate.
2. **Software Size estimate:**
    - Size(s), unit(s) of measure and sizing method used (e.g., International Function Point Users Group Standard or Simple Function Points or other);
    - Growth factor(s) and/or adjustments applied;
    - Source document(s);
    - List of included software requirements;
    - Templates or shortcuts used.
3. **Software Effort, Cost and Schedule estimates**:
    - Estimating approach(es) (e.g., analogy, published parametric equation, derived CER);
    - Assumptions not covered in #1 above
    - Effort estimate and unit(s) of measure (person hours);
    - Cost Estimating Relationships (CERs) /Schedule Estimating Relationship s (SERs) used and range of applicability
    - Productivity assumptions
    - Historical data used (as applicable)
    - Cross-checks and assumptions
    - Growth factor(s), risk, and uncertainty considerations (and confidence levels)

6. Conclusions

As outlined in this article, software cost estimates prepared without care or knowledge can contribute to underbudgeting of software development projects, thereby reducing a project's chances of success. While estimating software costs will always involve subjectivity and assumptions about the development which may or may not occur, an estimate based on formal estimating processes, normalized historical data, and participation by subject matter experts is a first step to creating a project environment where success is possible. While better software cost and schedule estimates will not guarantee success, better estimating processes coupled with better data and collaboration between developer and estimators will help. In addition, a recognition that software cost estimation is a professional endeavor to be practiced, formalized, and standardized can provide projects with contracting environments and funding to increase successful on-time and on-budget software deliveries. While estimates are the best guess of what the project should cost and how much effort it should take given ideal circumstances, they are still estimates and should be taken as such. There are no guarantees of actual project performance, especially when historical data prove otherwise. Estimates based on realistic historical data and data-based cost estimating relationships (CERs) give projects a baseline on which to base changes, and a fighting chance for success.

References

CHAOS Reports, Standish Group, various years biannually 1994-2020
https://www.standishgroup.com/sample_research_files/chaos_report_1994

Cost Estimating Body of Knowledge for Software (CEBoK-S), International Cost Estimating and Analysis Association: ICEAA, Dekkers, Carol- Leader Author, 2023

Solving for Risk Management: Understanding the Critical Role of Uncertainty in Project Management, Smart, Dr. Christian, 2021

Software Measurement Compendium: Estimating and Benchmarking Success with Functional Size Measurement, Bundeschuh, Manfred and Carol Dekkers, Springer Publishing, Germany, 2008

The Agile Manifesto, aka **The Manifesto for Agile Software Development**, Beck, Kent, Mike Beedle, Aire van Bennekum, Alistair Cockburn, et al, 2001, https://agilemanifesto.org/

Websites:

- International Cost Estimating and Analysis Association (ICEAA) www.ICEAA-online.com
- International Function Point Users Group (IFPUG) www.ifpug.org
- Quality Plus Technologies, Inc. www.qualityplustech.com