Rethinking Software Development Estimates:

FOCUS ON TEAM MEMBERS

By: Joe Schofield

Abstract

What this article is not. It is not a survey, not an evaluation, not a recommendation, not a comparison of estimation and forecasting tools, platforms, methods. Rather, this article focuses on the need for deep understanding of contributing team members, their individual competencies as they relate to the work they perform, their physical proximity and access to fellow team members, their willingness to collaborate amongst themselves, and their limited distraction to work commitments outside the project being estimated. And finally, the still unrelenting conflict between traditionally-based culture and project management with Agile projects that are discovery-oriented and truly practicing "changing requirements even late in development".

BACKGROUND

"Estimates are not intended to be accurate" is a phrase oft used to explain variances for cost and schedule, for work in general and software specifically. How we establish initial estimates, who develops the estimates, the purpose of the estimate (to make a proposal seem attractive to a potential client, or to predict a most realistic timeline for fulfilling that proposal), the potential risks associated with the work itself, and the use of internal historic data or benchmark data, all influence the integrity of the estimate. Sadly, but not atypically, once rendered, the estimate itself is frequently interpreted as a delivery promise. Actual performance data for project management constraints like cost and schedule—while often reported as dismal[1]—is often accepted as reality. But is our desired outcome to excel at

delivering value, or to create a reasonable bid for a fixed price contract, or to appear to have mastered inflating time and cost estimates?

Software estimation is not for novices. Software development companies can swell profit margins or damage their brands and existence as a result of their estimation processes. For internally developed software, reputations, careers, bonuses are more likely at stake. Models [COCOMO, COCOMO II], benchmarking [ISBSG], internal historical data, and tools are intended to help improve our accuracy. Organizations [ICEAA, IFPUG, BFPUG, GUFPI], functional measurement standards [ISO], and conferences [ISMA, PMI Global Summit, et al], further hone estimation processes, practices, and knowledge sharing respectively. Books, articles (still another), training, podcasts[2], and certifications attempt to plan, monitor, and control the management of such undertakings. Platforms, development languages, risks, tools, environments, margins of errors, past performance, and requirements are often included in those estimates. Team capability too is often incorporated; sometimes portrayed as an anticipated team productivity. A reflection on five decades of software development practices provokes the question "how well do we understand components of team productivity variation"?

When we don't know what we should know. Before delving into opportunities for enhancing the integrity of estimates, it is reasonable to dispel some of the beliefs that may prevail. Here are two examples of *when bad estimates appear to be good,* and their uniqueness remains doubtful.

1. Inflating estimates to provide software development teams with a larger margin of error is a decades-old ploy. However, even nascent Agile teams are already applying "estimation inflation" masterfully, and it's unrelated to economics. Refactoring, unnecessary tasks, inflated task hours, and deflated productivity rates, provide the team with an excessive of hours to complete their work. Less obvious however, the business is deprived of potential value for which they were paying. When applying these same techniques to release and project planning, such antics only increase the potential loss of value delivery.[3]

   A somewhat different spin on *estimation inflation* ensures those estimates are accurate by slowing progress on tasks to closely align with actuals. This is a "win, win, win." Myopically, the business wins because they are getting work completed "on time." The team wins because it appears that they are performing according to the plan. The project manager wins because he/she is recognized as a competent estimator. In the long term, the organization loses: value is under-delivered, actual data is specious, future estimates are based on inaccurate data, and contributors gain a false sense of self-worth.

2. How often do we hear of projects being completed on time and within cost? Extreme instances of variances between estimated and actual costs and schedules make for eye-catching headlines. They also propagate the need for project management tools and governance. Granted, some percentage of work is actually delivered on time and on budget, but this too can be deceptive. As an example, fixed-price contracts may overrun their budgets--in some cases, by substantial amounts. However, to meet the expectations of their funding agencies as well as their leadership, they discontinue charging for the actual time required to deliver a release. Often times, overtime is accrued but not reported. The project lead or the team as a whole receives accolades for a job well done despite their failing to produce in conformance with their own estimates and labor plans. Once again, the impact of intentionally hidden costs of overruns is the unwarranted optimistic cost projections for future deliveries. The unreported cost overruns of the past become "tribal knowledge" passed along to selected future team members or is lost.

This third example is also unique. In this instance we can't be sure that the estimate is good or bad because team performance is clearly lagging. At least in this example, estimation is not to blame. Early in my career I recall attending project-status meetings with my peers and our second-level management. One of my more senior peers reported that their schedule slippage would not reoccur because they were "smarter now." After a couple of meetings with the same "smarter now" antidote, I chimed in that if they got any smarter, they would never deliver. Sneers and frowns were often my reward for stating the apparent. Defects and delays were also likely the reward for those who ignored the signals emitted by their slumping performance. Poor estimating wasn't their Achilles heel; incompetence and lacking accountability were more likely suspects.

The three examples above, may not seem to evidence a lack of understanding the team's role as a source of variation; that is, not without further explanation.

EXAMPLE ABOVE	BUT A CLOSER VIEW OF THE TEAM REVEALS . . .
A – the team was employing estimation inflation	Several new team members Naiveté of all team members on a scrum project Supportive but complacent novice Agile management
B – the team delivered on time and within budget	Large program with multiple teams, all with some new membership Burnout from uncompensated overtime triggered massive team turnover after every release Executive leadership either was misled about or ignored actual progress
C – the "we're smarter now" team	Some new team members Accommodating team leadership tolerated minimal expectations Key skills absent from the team

Granted, some team capability is included in estimation models [COCOMO II, SEER, SLIM as examples] and tools offering a range of values that attempt to reflect the ability of the team to perform the work. These are *team- and organization-based indicators.* They reflect a high-level of confluence of past performances by projecting similar performance onto future work based on degrees of similarity. This is precisely why understanding *individual team member capabilities* is necessary. Summary research evidence depicts why.

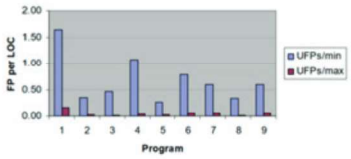
Data from software developers writing the same code, using their preferred language, each with relevant advanced degrees, in a classroom setting, subjected to the same validation tests, revealed that more than ½ of the participants wrote a program that had the fewest lines of code, and subsequently for a different program, also the most lines of code.[4] With the same control factors for each participant, a logical, perhaps the most logical, conclusion is that the specific problem was subject to the specific domain knowledge of the solution, on a developer-by- developer basis. Further, the research suggests that no "weak link" (underperformer) was evidenced in the study, and surprisingly, no "strong link" either. Outcomes were driven on a case-by-case basis specific to each participant. This "experiment" was repeated under the same control factors on three occasions. In one instance of 49 participants using "C" code, variation soared to as high as 2200%. The smallest variation was 960% and the average was 1500%. The nature of this research is still cited by the National Institute of Standards and Technology.[5] A key slide from that presentation is included below. Unlike many statistical samples, as the population grew, so did the range of variance.

Min and max values for "C" code compared to Function Point size over 9 programs (n = 49)

	*P1	P2	P3	P4	P5	P6	P7	P8	P9
Min	22	20	15	13	27	23	25	21	25
Max	221	311	336	289	270	306	242	383	284
UFPs	36	7	7	14	7	10	15	7	15
UFPs/min	1.64	0.35	0.47	1.08	0.26	0.70	0.60	0.33	0.60
UFPs/max	0.16	0.02	0.02	0.05	0.03	0.06	0.06	0.02	0.05
Variance Range	10.05	15.55	22.40	22.23	10.00	13.30	9.68	18.24	11.36	14.76

"C" Min/Max Function Point Range

Largest min to max variance is > 22, smallest is almost 10, average is almost 15.



Note that in these three examples, variance and averages increased as the population increased.

17

Individual team member competence associated with each program undertaken is an example of a *"micro" indicator.* I am unaware of any method, tool, standard, body of knowledge, or certification, that takes this possibility—individual team member expertise with the yet-unknown and unassigned task—into account. It may be impossible to know such, and thus is a cause of highly probable variation cloaked in other variables, masked by other more discernible "productivity" factors.

The Tuckman model,[6] best known for its forming, storming, norming, and performing stages of group development seems to reflect the impact of micro indicators as they pertain to team composition. Project Managers may relate to this as the Tuckman Ladder. The Examples A, B, and C above all "suffered" from the impact of frequent team turnover and therefore the consequences of groups reverting to the *forming* stage. Team advancement to the storming, norming, and performing stages is seldom linear. It is often interrupted with significant change to vision or approach, or more often, the loss or addition of team members. Since productivity is limited until the *norming* stage, and due to change in desired outcomes and the team itself, anticipating reversions to the *forming* stage is unreliable, typically unpreventable, and definitely unpredictable. Yet this highly impactful variable is seldom if ever taken into account when project plans are devised. Risk management and mitigations don't eliminate this major source of variation.

> "Your organization, your teams' recent, relevant, and valid actual data are the best predictors of future performance.

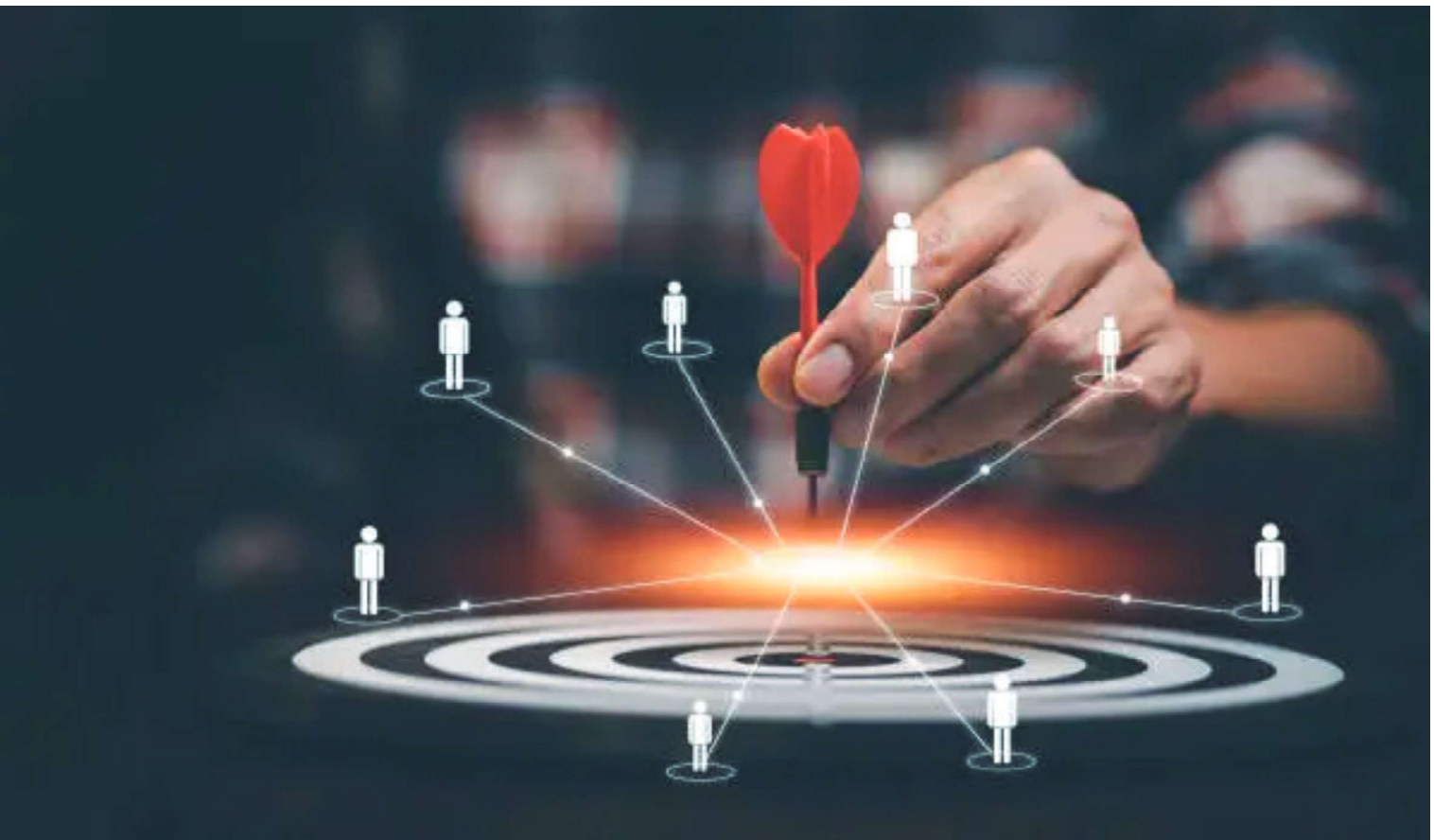
Behaviors in the *forming* stage-like variability itself doesn't cease with a frozen team. The strain on group maturity is exacerbated by the ability for individual team members to collaborate, grow cross-functionally, address their competency and skillset gaps with evolving project needs, and of course, collocate (enabling osmotic communication,[7] relationships, and mentoring) in today's upheaved work environment. More often an even larger source of variation is introduced when team members are shared across multiple projects. The phrase "I'm always in meetings" signals that team members are shared beyond their expected time to make meaningful and timely contributions. Scrum team members, for instance, lose about five hours a week of development time just to attend the minimal array of team ceremonies—per team on which they participate. For non-Agile teams, their time in meetings often grows even more quickly.

A final point on team dynamics. Large teams trigger more interactions and increased complexity. More interactions consume more time, which also negatively impacts the ability to deliver. One study concluded that 4.6 persons was the perfect team size.[8] And then there's the Ringelmann effect which demonstrated that the more folks were added to a team, the less effort each member exerted.[9] Quantifiable or not, there exists a substantial list of variables that drive productivity sky high, or into the ground, including:

- the size of teams,
- the velocity of team member turnover,
- the fractionalizing of team members across multiple teams,
- work from home vs. collocation (like osmotic communication [Cockburn]) and avoiding of isolation (it's hard to feel necessary when you're physically disconnected),[10]
- the speed at which teams can progress through Tuckman stages,
- the number of times a team reverts to Tuckman's forming stage,
- the familiarity of the team with the business, their own methods, and the target technical solution.

Recommendations towards improving current estimation practices

- Use recent data; that is, data that is three years or less to help account for constantly evolving platforms, technologies, and security needs.
- Use relevant comparisons. Traditional "predictive" examples may add data points for statistical confidence, and may actually have some relevant project management data, but
    - o if the project is developed with Agile frameworks, use Agile data,
    - o if the project is "firm-fixed-price" do not compare it to Agile projects that honor the 2nd Agile principle regarding the acceptance of "welcoming changing requirements even late in development"; this may also put the kibosh on estimation practices that are driven by a functional size value, because it only exists after each sprint planning session for discovery-driven and evolving Agile projects
    - o if the project plan calls for a single release, don't compare it to projects that "deliver working software frequently", all of these are very different in their development approaches.
- Sad to say, some projects don't record their actual data. That's not to say they don't record data. Team members may provide updates that favor what they are being evaluated on, or yesterday's marching orders. See Examples A and B above.
- Develop deep understanding of your organization's personal and team development capabilities. Build cross-functional skillsets to reduce the pain of team members that leave suddenly, are otherwise indisposed, or spikes in certain skillsets. How are hybrid and work-from-home options impacting product delivery? (Hint: the top technology companies are requiring their staff to return to the office.[11,12]) Is the spreading of team members over multiple projects hindering delivery commitment, integrity. What size teams work best for what types of work? How collaborative are teams? How quickly are new team members assimilated? What practices and activities speed cohesion?[13,14]
- Use your data. Your organization, your teams' recent, relevant, and valid actual data are the best predictors of future performance. Your data best reflects your cultural accelerators and nuances, the development environment, the capabilities and skillsets of your team. Unfortunately, few organizations do this. Those that do may have data



corrupted by inaccurate data, don't want to be compared to other teams, don't know how to use the data once its available. Most estimation-related consulting groups would argue that this lack of measurement focus isn't unfortunate at all; it has become their livelihood to the benefit of the software industry.

In closing

You can't estimate until you identify all work variables.

You can't estimate with integrity until you identify all of the sources of variation in those variables.

You can't estimate well until you associate performer capabilities with the task, assigned or selected.

Cost is a variable. Labor rates are a component within cost. Team member quality (competence) is a variable within labor rates that influence those rates. This variance is evident but not often explained when comparing and contrasting planned vs. actual values. Experience is also a contributor to competence, as is education. Focus, or the ability to focus, is a contributor to competence. Splitting people across multiple projects limits focus. Collocation, collaboration, cross-functionality are all also contributors to competence. Until all of the contributors of competence are understood and quantified in a meaningful and relatively accurate model or algorithm, labor rates will vary by task and by those completing the task. Competence will also impact quality that will impact defects, which will impact rework, cost and schedule as high-level variables. While competence is a variable that is frequently identified in labor-related estimates, others identified in this article may help to improve our understanding of the relationship between team member performance and successful delivery.

> The best estimates result from alignment of team member skillsets with each task undertaken by that team member.

The literature often highlights poorly performing projects that are staffed with teams of 10, 100, or 1000 developers, span multi-years, experience high turnover, are well over budget, and are led by a revolving door of management consultants and inexperienced developers.[15,16] Rendered solutions often point to better project management rather than more stable and capable teams. This is especially true and dangerous with Agile development teams when traditional project management metrics are forced upon non-traditional Agile teams. Senior leadership and culture are the most significant contributors to impeding Agile adoption which, includes Agile thinking. I have found that the best data for measuring teams' performances is their own data. Regrettably, few teams and organizations keep meaningful data. The best estimates result from alignment of team member skillsets with each task undertaken by that team member. The best execution of development practices come from collaboration among that team and the client, not historic benchmarks that rely on dated past performance and varying solution development approaches.

Yogi Berra was right, "prediction is hard, especially about the future."[17]

Further reading: https://en.wikipedia.org/wiki/Software_development_effort_estimation (great resource for history and survey of tools)

Special thanks to Karen McRitchie, Colin Hammond, and Larry Putnam who accepted my invitation to discuss this subject and whose comments enhanced the content of this article.[18]

REFERENCES

[1]https://management.simplicable.com/management/new/why-your-estimates-are-always-wrong; Why Your Estimates Are Always Wrong; Anna Mar, February 17, 2013

[2]70 – 80 percent of all SPaMCASTS touch on software estimation, project management, team development; Tom Cagley; 9/11/2023

[3]Inflate Gate: Mastering Overestimation for Agile Software Projects; Computer Aid's Accelerated IT Success; (Featured Article); IT Metrics & Productivity Institute; August, 2015

[4]The Statistical Case Against the Case for using Lines of Code in Software Estimation; 4th World Congress on Software Quality; Bethesda, MD.; September 17, 2008;

[5]https://www.nist.gov/itl/ssd/software-quality-group/metrics-and-measures; retrieved 9/11/2023

[6]https://en.wikipedia.org/wiki/Tuckman%27s_stages_of_group_development; retrieved 9/13/2023

[7]https://trustedinstitute.com/concept/agile-project-management/crystal-methods/osmotic-communication/ ; retrieved 9/26/2023

[8]https://knowledge.wharton.upenn.edu/podcast/knowledge-at-wharton-podcast/is-your-team-too-big-too-small-whats-the-right-number-2/ ; retrieved 9/26/2023

[9]Ringelmann effect — measured individuals and teams pulling a rope; as more folks were added to the "pull" less effort was exerted by each team member. In his day, this helped to explain "social loafing." https://knowledge.wharton.upenn.edu/podcast/knowledge-at-wharton-podcast/is-your-team-too-big-too-small-whats-the-right-number-2/; retrieved 9/14/2023

[10]"Malcolm Gladwell: https://www.foxbusiness.com/lifestyle/malcolm-gladwell-says-people-must-return-office-regain-sense-belonging; 8/8/2022

[11]https://www.cnn.com/2023/06/10/tech/silicon-valley-return-to-office-tensions/index.html; retrieved 9/29/2023

[12]https://www.cnn.com/2023/08/07/business/zoom-return-to-office/index.html; retrieved 9/29/2023

[13]https://www.foxbusiness.com/technology/amazon-ceo-staff-resisting-returning-office-probably-wont-work-out; retrieved 9/2/2023

[14]The move to bring workers back to the office for three in-person days came from a judgment call that Amazon executives made. Amazon said Jassy told employees at the meeting earlier this month. The company decided to go that route after looking at how teams were collaborating, the company's culture, performance of the business and other factors. https://www.foxbusiness.com/lifestyle; retrieved 9/2/2023

[15]https://blog.kintone.com/business-with-heart/blog/why-projects-fail-unreliable-estimates; Why Projects Fail: Unreliable Estimates; Euna Kim; Oct 28, 2020

[16]Why Big Software Projects Fail: The 12 Key Questions ; Watts S. Humphrey; The Software Engineering Institute; CrossTalk; March, 2005

[17]https://www.goodreads.com/quotes/261863-it-s-tough-to-make-predictions-especially-about-the-future; retrieved 9/12/2023

[18]I was Impressed with deep understanding of tools developers like SLIM and SEER, attempts at precision, ability to filter by various parameters, and ScopeMaster's emphasis on requirements as the basis for scope.

ABOUT THE AUTHOR

Joe Schofield SCT, SCAC, SSMC, SSPOC, SMC, SPOC, SDC, SAMC, CSQA, CSMS, SA

Independent Consultant – Enabling Organizational Capability

Scrum Certified Trainer | Certified Agile Coach | Certified SAFe® 5.0 Agilist

Past President, International Function Point Users Group

2023 National Champion; Powerlifting America; 74kg, Master IV

2022 National Champion; USA Powerlifting ; 75kg, Masters IV

Selected Key Roles: As a Distinguished Member of the Technical Staff at Sandia National Laboratories, Joe worked with the Departments of Defense, Energy, and Interior, and the NNSA. During his 31-year career he served for over a decade as the "Chief Process Officer" of an organization of 400 software engineers, which was awarded a SW-CMM® Level 3. He continued in that role to CMMI® Level 4 until his departure. Joe is a Past President of the International Function Point Users Group.

As an enabler and educator: . . . Joe is an Authorized Training Partner with VMEdu, a Scrum Certified Trainer and Agile Coach with SCRUMstudy with over 80 published books, papers, conference presentations and keynotes—including contributions to the books: The IFPUG Guide to IT and Software Measurement (2012), IT Measurement, Certified Function Point Specialist Exam Guide, The Economics of Software Quality, and his recently released Aligning People and Culture for Agile Transformation. He has taught over 100 college courses in software engineering and strategic decision making; 75 of those at the graduate level. Joe has facilitated ~200 teams in the areas of software specification, team building, organizational planning, and Agile transformation.

Lifelong learning: . . . Joe holds nine Agile-related certifications: SCT™, SCAC™, SSMC™, SSPOC™, SMC™, SDC™, SPOC™, SAMC™, and SAFe 5. Joe was a CMMI Institute certified Instructor for the Introduction to the CMMI®, a Certified Function Point Counting Specialist, a Certified Software Quality Analyst, and a Lockheed Martin certified Lean Six Sigma Black Belt. He completed his MS in MIS at the University of Arizona in 1980.