Chapter 5

Combining Soft Skills and Hard Tools for Better Software Estimates

carol dekkers - October 2, 2014

Practice Areas: Scheduling

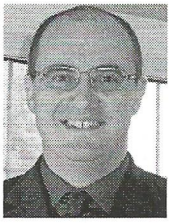
Anyone in the IT industry knows that software projects are the perfect combination (some might say the perfect storm) of art and science. There is the creative, innovative (right brain) aspect of designing a new system and the engineering (left brain) analysis and construction required to get the project off the conceptual drawing board and into production. The industry (and its stakeholders) houses a cultural melting pot of professionals, boasting artists (musicians, designers, writers) and technical gurus (computer scientists, engineers, mathematicians, accountants, etc.). IT may be a young industry, but it is a uniquely diverse one.

Trending Articles

Slashing Risks with User Experience Engineering (Part 4)

by Dmitri Khanine January 14, 2016

This is the fourth and final installment in this series on using the latest UX methods for focusing on the right problems and slashing requirements-based risks. In this installment, we will be validating designs, using our prototypes for conducting usability tests.



Learning from the New PM

by Andy Jordan January 19, 2016

We like to think we can impart some wisdom into people new to our profession, but we also have opportunities to learn from them. We have to try and look at how we execute projects in our organization through the lens of that new PM--and be prepared to question even the most fundamental part of that approach.

Agile for Hardware Product Development

by Jason Chisham, PMP January 21, 2016

It's rare to read about product development processes these days without mention of the benefits of using agile methodologies, yet one might be led to believe this only works for software development. Are hardware teams left out of the agile world? Not at all. You just need to be agile and adapt.

IT-centric jobs comprise four of the top ten featured in the 2013 edition of *U.S. News and World Report's* "The Top 25 Best Jobs"—#4 Computer Systems Analyst, #6 DBA, #7 Software Developer, #9 Web Developer—with percentage growth rates predicted to be in the double digits. Information technology remains for many an exciting career choice that promises innovative and creative opportunities for technology professionals.

It may not seem surprising that such a young and burgeoning industry is also plagued with burnout and unintentional dysfunctional behavior. After years of rework (over 40% on most software projects), 60+ hour work weeks, weekends, and midnight programming binges, beleaguered IT pros leave the industry. Those who make it to retirement can attest that there's never enough time to do it right the first time but there's always enough time to go back and do it again.

When you couple talent drain with global studies citing a mere 30% project "success" rate (with success defined as on-time and on-budget delivery), you might wonder where the industry is headed. But there is good news. If we combine lessons from psychology (specifically, human behavior and communication) with some new science-based estimation tools, we could multiply our present success and create a better future for IT professionals.

Dysfunctional Behaviors in IT

There is a joke circulating on the internet that goes like this:

The project manager walks into his boss's office and says, "Here is the bottom line budget needed for the success of the project." The boss says, "What can you do for half the money?" The project manager says, "Fail." The boss says, "When can you get started?" The project manager says, "I think I just did."

Here are a few major problems that threaten our projects before they even start:

1. Overly optimistic and unrealistic estimates (often based on concepts rather than solid product scope);
2. Management distrust (and flawed assumptions about software development);

3. Lack of respect for expertise on both sides ("father knows best" syndrome);
4. Negotiation rather than collaboration (a culture of blame and risk transfer/avoidance).

This article focuses on the first one (the latter three are subjects for future posts), and takes a look at applying aspects of psychology and human behavior for insight.

The Essence of Estimating

As professionals (defined as people with knowledge to do specialized work) throughout a corporation, we've somehow altered the definition of "estimate." Dictionary.com defines the word "estimate" as:

- *An **approximate judgment or calculation**, as of the value, amount, time, size, or weight of something.*
- *A judgment or opinion, as of the qualities of a person or thing.*
- *A statement of the **approximate** charge for work to be done, submitted by a person or business firm ready to undertake the work.*

Yet when it comes to IT, the word estimate is often used interchangeably with terms like budget, schedule, and guarantee. *An estimate is never a guarantee*, but rather a best guess of how big, how much, and how long something should take to build/develop/deliver *given the data that we know and is available at the time.*

It should be a no-brainer that preliminary estimates based on incomplete or missing data, or based purely on theory (rather than history), will end up being wrong. As such, the notion of project success defined as being on-time and on-budget when compared to preliminary estimates is sheer folly. But, since perception is reality, if we want to increase project success, it's really a matter of getting better estimates.

In other words, the industry definition of project success hinges on the ability of the estimator to predict the right schedule and budget, since projects that go over budget or fall behind are deemed failures. This is where the challenge lies. How can one estimate a project while it is still as amorphous as a cloud in the sky? It is difficult, but not impossible if you put limits on your estimate and base it on historical reality—and, I believe, we should always verify our estimates with a second opinion.

Estimating Psychology

Estimators often find themselves tasked to create estimates before the requirements are complete—sometimes budget cycles ahead of time—and must factor in priorities, constraints, and even future regulations. When this "best guess" (most often a single estimate), ends up being wrong in terms of the project being late and over-budget, estimators end up looking bad. Even with the best models (top down, bottom up, work breakdown structure (WBS) based, spiral, agile, etc.), passion for the job, and the best intentions, accounting for all this uncertainty can seem like an uphill battle.

A speaker at the University of Washington college commencement address shared ten aphorisms with the graduates. While citing from a graduation speech might seem completely unrelated, a couple of statements struck me as being relevant to this post:

> #7: Define yourself by what you love… Be demonstrative and generous in your praise of those you admire;
>
> #8: Respect those with less power than you. (I would amend this point to add "or knowledge")

I believe that the majority of software developers, estimators, and project managers love what they do, and do the best job they know how to do, and I believe that there is a lack of respect for such expertise when things (such as estimates) go wrong. The typical reaction in the IT industry when an estimate is wrong is to blame, deflect, deny, denigrate, fire, or demand change from those involved—none of which helps the project. If we followed the two points above, a better response would be to examine what changed (since the time of the estimate) and find ways to reaffirm/adapt/better the estimating assumptions.

Often, consultants are hired to fix the estimating process and bring with them new tools/models to replace current methods. When this happens, it is tantamount to disrespecting the masses, when in fact, the new tools/methods should augment, supplement, and tweak the existing methods to provide a solid second opinion.

Are bottom up estimates flawed?

Bottom up estimation occurs when a project is decomposed into a set of project tasks. Time or effort required to complete each task is estimated based on expert judgment as to how long it *should take* to do, then a project total is derived by summing the individual task estimates. There is nothing wrong with bottom up estimating per se; however, one issue that arises is the assumption of optimistic conditions to complete tasks that preclude scope or historical realities. Estimated hours seldom consider the overall project size ortake into account the realities of rework, contingency, or interdependencies. For example, when a task such as "code module x" is estimated, the hours typically represent how long it should take a dedicated programmer to sit down with well-defined requirements and code it perfectly the first time. This seldom happens—interruptions occur, rework happens (40–60% of project effort is rework), requirements are incomplete, upstream tasks delay downstream tasks, related requirements come into play, the specifications are more complex than anticipated, code is copied and pasted (without customization), and a myriad of other issues occur. When we estimate hundreds (or thousands) of these individual tasks in relative isolation, the estimate looks good at face value because it is so detailed. What is missing is a range for the estimates, incorporation of time for rework and contingency, and a way to account for the interdependencies of individual tasks. In addition, because the estimate is so detailed with precise sounding numbers (often including decimal point fragments of hours), anyone looking at the estimate assumes the estimate is almost a guarantee of what the budget and schedule will be.

Combining Soft Skills and Hard Tools to move forward

Bottom up estimating doesn't work well as the only estimating method—even if we could increase the estimate to address the issues outlined above. I believe we need to find a more robust and empirical basis for project estimates. (Note, this doesn't mean discarding the bottom up estimating, it simply means augmenting them using additional techniques.)

The following points can be used to advantage with estimating:

- Historical project totals (effort, duration, costs) are facts. It matters not what the original estimates may have been; the reality is that completed projects are reality.
- History repeats itself and is far more reliable than theoretical models.
- Human behavior is repeatable—we are overly optimistic that we can do better next time even without changing how we do projects. Even when the last 15 projects had a particular issue, we still think that whatever caused it is an anomaly. It isn't.
- Projects are seldom smaller and easier than we anticipate. Again, we pride ourselves on being good communicators and project managers who can overcome problems that arise. Even though we know that projects grow (the rule of thumb is 1.5% per month of the project), we tend to ignore this when performing estimates.
- Rework is not a legitimate task. Even though research on completed projects shows rework figures hovering between 40% and 60%, we cannot include rework as a lump sum task on projects. Because of this, our estimates end up being theoretical (based on idealized task effort or perfect work conditions), rather than practical (based on past performance).
- Project teams want and deserve respect from their peers and customers. When professionals are blamed or disparaged for work they diligently and knowledgeably perform (such as estimating or project work), they lose faith in the process and the project. Apathy, in terms of not contributing to planning or estimating, ends up creating worse outcomes. Respect for people and their knowledge (given their training) is an important part of project success.

Today, there are tools on the market that address and incorporate these factors. These tools allow estimators to use their own completed projects for trend line comparisons and calibrate the estimates to their corporate behaviors and proven ability to deliver software.

Education equals knowledge equals respect and improvement

When we boost up our estimating knowledge using proven tools to augment our existing approaches, there is no doubt that our skills, confidence, and success rates improve.

On software projects, it is about time we started improving our project estimates, which in turn leads to higher levels of project success (on time and on budget), which in turn leads to higher respect and better overall corporate morale. Using these tools allows us to scientifically do this with the confidence that past projects are great predictors of future performance.

It's time we use psychology (soft skills) and science (robust estimation software) to better our project success rates, and to increase the confidence of the business, industry, and our peers on software projects.

About the Author

Carol Dekkers, PMP, is a consultant, author, and speaker at international project management conferences, and has been a member of the U.S. delegation to ISO software engineering standards for 20 years. She is the co-author of two books: *The IT Measurement Compendium: Estimating and Benchmarking Success with Functional Size Measurement*; and *Program Management Toolkit for Software and Systems Development*; and a contributor to many more. Carol does consulting for QSM, Inc. and Quality Plus Technologies, Inc., and resides in Florida.