Predictable Change

Flexing the Five Core Levers of Software Development

Dr. Andy Berner

QSM® The Intelligence behind Successful Software Projects

Contents

QSM® The Intelligence behind Successful Software Projects

Embrace Change

For as long as there has been a software development industry, management and development have fought about change. Managers want to add functionality as the project is in flight while insisting on the original estimates of schedule and effort. Developers hate scope creep when it pulls them away from their elegant analysis and design while insisting they need for more time and resources to get the work done.

Agile philosophy is to "embrace change." Both project management and engineering techniques built into Agile methodologies *facilitate* change, rather than merely tolerating it.

But Agile techniques don't eliminate the need for effective project estimation. No project charter says, "Get done whatever you can, take as much time you want, and use whatever resources you need - we'll be happy with the results, whatever the quality." Organizations still need a business case that shows the project is worth doing. The challenge is to predict the cost and value in advance when we know and accept there will be changes.

The Same, Only Different

Software developers resist uniformity. *"My project is different,"* they say. And they are right. As Ken Schwaber and Mike Beedle noted, software development is not very amenable to being managed like a repeatable manufacturing process. [i] If every project is unique, how can we estimate unique projects in a consistent, repeatable way?

As different as they are, all software projects share certain characteristics. Larry Putnam, Sr. and Ware Myers describe "Five Core Metrics" that characterize the performance of software development projects: size, productivity, time (duration), effort, and reliability[ii]. These metrics represent five "levers" we can use to embrace and manage project change.

For over 30 years, QSM has studied the relationships among these five levers and has built a large and growing database of information from projects of differing sizes, application types, and development methodologies. Most recently we have been collecting data on Agile projects. We are learning how Agile methods affect the relationships among the five levers.

Future articles on the QSM web site (www.qsm.com) will address each lever as it relates to Agile project management and engineering methods. Here we begin by examining the overall meaning of each metric and the relationships among them.

The Meaning of the Five Core Levers

Time (duration) and Effort—The "Easy" Ones

Time refers to the calendar duration (in months, weeks, etc.) for the entire project. ***Effort*** is the number of person months (work hours, FTE years, etc.) of all the team members on the project. There is nothing "Agile specific" about time, effort, or the units in which they are measured. In practice there can be subtleties such as, "When does the project start and finish?" The notion of "potentially releasable software" and the changing decision of when to release introduces some ambiguity, but these issues are not new; continuous delivery occurred in mainframe development decades ago. In many Agile projects the team is kept constant, and since the major cost in software development is the cost of the people, we may switch between looking at cost, effort, and team size. For both time and effort, we find the basic meaning and measurements are very natural to project managers.

Size—The Measurement of Scope

While it is clear that some projects are "bigger" than others, specific measures of system ***size*** are not often used by project managers or development teams. Scope—the requirements, functions, and stories to be developed—is the qualitative version of size and is more natural. Of our five key levers, this is the one that most shows "All projects are different, yet alike." No two software projects will develop exactly the same set of stories but frequently, different sets of stories may be about the same size.

For size-based estimation the details of the project scope are not important. What matters is the size of the proposed project, relative to other projects completed by your organization or, if your own historic data is not available, to relevant industry data.

Over the years, sizing techniques such as Function Points (see http://www.ifpug.org) have been developed, but until recently explicit size metrics were rarely used after the initial estimate. Agile methodologies are changing this. Agile teams routinely measure the size of their backlog in story points or counts of like-sized stories. These units are relative size measures, yet they are still very useful for planning iterations. But if we want to use past projects to predict future performance of entire projects, we must relate *"relative size"* in the context of one project to *"normalized size"* that can be used to compare projects.

Productivity

As much a relative measure as size might seem, ***productivity*** is even worse. We know some teams are more productive than others, but it's difficult to quantify how much more. We are challenged even to define the term and certainly hard pressed to give productivity a number.

Agile is changing this too. A key Agile metric is "velocity"—the number of story points completed per iteration. To be most useful, velocity depends on two constants that are common to Agile projects: the size of the team remains steady and all iterations in the project are the same length. When both these conditions are true, we can assert that the higher the velocity, the more productive the team.

However, this is true only within a team. If one team has 10 members and another has five, we would expect the larger team to accomplish more in the same time. How much more is an very interesting issue we'll explore later, but the primary use of "team velocity" is within a given project with a fixed team, to predict how much will be accomplished in future (same length) iterations of that project.

QSM uses a more sophisticated metric called "productivity index" (PI) which empirically measures how three of the core metrics (size, effort, schedule) interrelate on different projects. We have found it is essential to factor in this explicit and quantitative measure of productivity to provide useful estimation.

Reliability/Quality

Quality can also appear to be a vague and somewhat subjective metric. There are aspects of quality that can be measured in different ways for different purposes. At QSM we have found that the defect arrival rate (or its reciprocal, Mean Time to Defect during development) is most useful for measuring the effect quality has on the duration and effort needed to complete a project. When quality drops significantly, correcting it can wreak havoc on the best planned and resourced schedule.

Agile methods address the quality lever with some key techniques rather than direct measurements. A goal of Test Driven Development (TDD) is to catch defects on the developer's desk before they get into a build, thus increasing the mean time to defect in builds and reducing the overall project time.

Embrace Change: Flexing the Five Core Levers

No project has complete flexibility, but likewise no organization can truthfully assert, "We will deliver exactly this scope, in this amount of time, for this cost, with zero defects." All projects are constrained in some dimensions, but can be flexible in others. Often the delivery date is the primary constraint, and Agile organizations often plan with the duration lever fixed and meet the deadline by flexibly adjusting how much of the backlog is delivered in the release.

For some projects, the lever of size and scope is less flexible than the others. It wouldn't do much good to deliver an income tax program on time if it handled only deductions but not income. The notion of "Minimum Releasable Scope" is gaining favor among Agile teams who recognize that even if we have working software of "potentially releasable quality," the customer may require more features before we have a viable release.

For other projects, the quality lever is the most inflexible. On a commercial aircraft, failure of the on-board entertainment system is annoying; failure of the navigation system is something else.

Sometimes the lever that is most rigid is effort or team size. Since effort is the major cost driver, this lever may be totally inflexible due to budget constraints. In another situation we may only have a fixed number of developers available, limiting the flexibility of the staffing lever.

Since we embrace change, we don't expect all five levers to stay in a fixed place throughout a project. Some levers are more rigid than others. Which levers are more flexible varies from project to project, so we need tools and methods that let us take varying constraints into account and "solve" for optimal choices where we have the most flexibility.

Using History to Predict Future Performance—Understanding the Relationships among the Levers

The Agile community is learning to predict what a team can do in a single iteration or sprint based on metrics from previous iterations. Constants that exist within a single project (team size, iteration length) help us do this. But how can we use the information from previous projects to decide how to position the five levers at the start of a new project? Can we predict what will be feasible if we are using a new team of a different size building a system of different scope over a different duration?

To move from using the five core levers on a single project to predict the feasibility and performance of a new project, we need two things:

a) Historical data from previous projects for the five core levers (size, effort, duration, defects, productivity) from our organization or from industry data which we can compare to our own performance.
b) The relationships among the levers – how will adjusting one lever affect the others?

The good news is that most Agile projects already capture raw data around these metrics. However, leveraging the relationships among software metrics to help predict future behavior is more challenging.

Let's look at why this isn't easy. We are starting a new project, one that is key to our competitive position. We need to deliver in six months. Our competition is already in the market, so we must at least match them on features and quality. The Minimum Releasable Scope is twice as large as other projects we've done recently, but the project is important enough to put our best people on it. We've collected metrics from previous projects we can use to estimate this one. Two teams stand out—their velocity is consistently high on the projects they've completed. If we put those teams together, their combined velocity should do the trick!

Here's why it's tempting to think this works:

- We compared the new project to previous projects and we flexed the size lever to twice the previous size, so we should expect to lengthen the project.
- If we assume the same velocity from a team, it would double the number of iterations needed for the previous project.
- Unfortunately, we can't afford to double the duration. Instead, we combine two teams that both achieved the same high velocity. Shouldn't this double the expected velocity and bring our schedule back to the six month window?

Unfortunately it is not that easy: the relationships among the levers are not that simple. Adjusting one lever affects the others: usually in the directions we expect, *but not in the amount we expect*. The relationship between schedule and effort is nonlinear—doubling the team size on a project does not halve the schedule.

The nonlinear tradeoffs that exist between effort and schedule apply to effort and defect creation. In general, increasing team size for a project of a given duration and size lowers the quality. Adding people increases the number of communication paths, which leads to more defects and thus more time spent correcting them and re-testing the product. The result is we dramatically raise the total effort and cost. We see the same nonlinear tradeoffs when we adjust project size— the larger scope increases project duration, but this time the tradeoffs work in our favor: doubling the project size (keeping team size, productivity, and quality fixed), lengthens the schedule, but doesn't double it.

The simple reasoning that led us to believe that combining these teams will let us meet the schedule doesn't work. We need tools that model the complex relationships among the five core levers, let us specify which constraints

will be most stiff on a particular project, and bend the more flexible levers to get feasible plans that fit our historical capabilities.

Levels of Confidence and Risk

The question we should ask during initial planning is not just *"How long will it take?"* Better questions would be: *"How likely is it that we can make our schedule? How likely is it we will meet this cost constraint? How likely is it that our team size is sufficient?"* The questions should include this qualifier: *"How likely?"*

Likewise, the answer should not be "12 months". It should include the risk. *"It's likely we can do this in 12 months. Planning for 14 months would be very conservative. Ten months Is plausible, but quite risky. But there is no way it will get done in 6 months. Not only has our team never done that much that fast, but nobody in the industry has!"* Quantifying the risk and expressing the level of confidence allows us to keep all levers as flexible as possible. We can plan for contingencies, allowing the most flexible levers to adjust as the project is carried out so we meet our constraints.

How can we predict what it takes to deliver a new project? We can use our historical data from the five core levers as a starting point. We can account for unique inputs and constraints from the project we are estimating. We can use tools that account for the nonlinear relationships among our core levers and adjust the more flexible ones in the right proportions. We make uncertainty an explicit part of the estimate. Our estimates will reflect the level of risk we can accept and allow us to plan for the changes we know are coming.

---

[i] "Agile Software Development with Scrum," by Ken Schwaber and Mike Beedle, Prentice-Hall, Inc., 2002

[ii] "Five Core Metrics—The Intelligence Behind Successful Software Management" by Lawrence H. Putnam and Ware Myers, Dorset House Publishing Co. Inc., 2002. Lawrence H. Putnam, Sr. is the founder of QSM, the developers of the SLIM tool suite.