Traits of Successful Software Development Projects

Enough already with Healthcare.gov and its embattled IT cousins; let's talk about government software projects that actually worked. Specifically, what do successful projects have in common, and how might forward-looking agencies replicate those conditions for success?

It's a difficult question to answer, but if we use the Quantitative Software Management Inc. (QSM) database, which holds carefully vetted information on over 10,000 completed software projects, including thousands of government projects, we can uncover common traits that can predict success for government IT projects.

Defining Success

Because "successful" and "embattled" are relative terms, we'll use the designation "best-in-class" – for projects that performed at least one standard deviation better than average in time-to-market, effort (or cost) expended and quality – and "worst-in-class" for the exact opposite.

To put this in perspective, using a sample of over 500 business IT systems, best-in-class projects were (on average) 3.5 times faster to market than worst-in-class projects, and required 8.1 times less effort. When narrowing the sample to government-only business systems, the best-in-class projects were, on average, 3 times faster to market than what was typical for the IT industry, and required 6 times less effort. Meanwhile, the worst-in-class projects, on average, took twice as long to complete and required 5 times more effort than the typical industry standard.

In software development, effort and time-to-market often work against each other. (When schedules are compressed, we typically pay a premium in effort.) Hence, projects with outstanding marks in both effort and time-to-market are quite rare and indicative of uncommon success. So given that background, what are the most common success factors for a best in class project?

Observation #1: Smaller Is Faster

The first and most prominent commonality among our best-in-class projects was a relatively modest team size. Contrary to the widely held belief that one can expedite a project by adding people to it, precisely the opposite is true. Larger teams often take longer to develop the same functionality as smaller teams.

Statistically speaking, the most efficient project teams rarely exceed 7-10 people (except on some of the largest, most complex systems greater than 200,000 lines of code). This represents an 18-39% reduction from industry averages. Furthermore, teams falling within this "optimal" size range completed projects with 28-69% greater productivity than larger teams with comparable tasks.

Based on this data, it stands to reason that government agencies should keep their IT project teams as small as possible – both for reasons of efficiency and for the added flexibility to take on new projects and reduce backlog.

Observation #2: Requirements Required

The second commonality pertains to the allocation of time and effort. Statistically, we found that best-in-class projects invested a much greater proportion of total effort in pre-coding activities – requirements analysis, architecture, and high-level design – than worst-in-class projects (28.0% vs. 7.6% of the total effort).

These findings were even more pronounced for government projects (36.3% vs. 4.4% of the total).

This data appears to support a "pay up-front or pay later with interest" concept, whereby best-in-class projects invest time and effort up-front through analysis in order to bypass greater time and effort expenditures later through maintenance and repair.

It's important to note that QSM's data set included both agile and non-agile projects. Regardless of whether or not the project is following an agile approach, sufficient requirements analysis, architecture and high-level design work needs to be done to ensure the backlog of required functionality has been clearly identified, sized and prioritized prior to construction. Otherwise, there will be extensive rework that can quickly turn an otherwise "best-in-class" project into a "worst-in-class" project.

For government agencies, the lesson would seem to be one of patience and wise investment. More often than not, the QSM data shows that management decisions to overstaff or to begin coding prematurely in an attempt to achieve an aggressive schedule backfire and ultimately produce inferior results at greater cost in the same or longer time frames.

The Case for Quantitative Data

Of course, individual projects will vary, and factors like team size and percentage of effort spent in analysis cannot guarantee the success of any single initiative. But the principle of using past software project data to predict future project outcomes is crucial to modern government.

And considering government's propensity for outsourcing, the ability to perform feasibility estimates (based on past project data) holds greater value for contracting officers who use those estimates to spot unrealistic bids up-front and, on occasion, to defend the procurement process from unhappy vendor protests.

These analyses are typically called Independent Government Cost Estimates, and they're critical weapons in the war against project inefficiency.

But if the data tells us anything, it's that estimation shouldn't stop when development starts. Project managers should be continuously tracking, reforecasting and reassessing their project decisions based on shifting requirements and budgets.

Perhaps, in addition to small team size and robust analysis, we should add a third common trait among best-in-class projects: active, dedicated project managers.

After all, what use is data without the right people to interpret it?