Does Agile Scale?

BY Larry Putnam, Jr.

Like Romeo and Juliet, government's flirtations with Agile software development practices have been the talk of the town. But there's one aspect of the story we tend to forget: government is big—really big. So what happens to Agile projects when they're forced to scale to the size of major government enterprise initiatives? We could speculate, of course. But instead, let's take a look at the data.

For this exercise, we analyzed 93 Agile projects, occurring between 2002 and 2012, mainly in the Government, Financial, and Health sectors. We divided the data into two datasets called Early Adopters (2002-2008) and Later Adopters (2009 – 2012), which gave us similar sample sizes in both groups. At the same time, we examined a sample of 93 non-Agile projects with the same sample demographics. Then, we charted how the projects fared as their sizes (measured in lines of source code) increased.

Here's what we found:

Early Adopters vs. Later Adopters

- Early adopters had a significant proportion of smaller boutique software firms
- Later adopters were mostly large enterprises, with many from the financial services and banking sectors
- Early adopters exhibited shorter project durations (faster to market) than later adopters
- Later adopters tended to use more staff than early adopters and tended to look more like the non-agile group

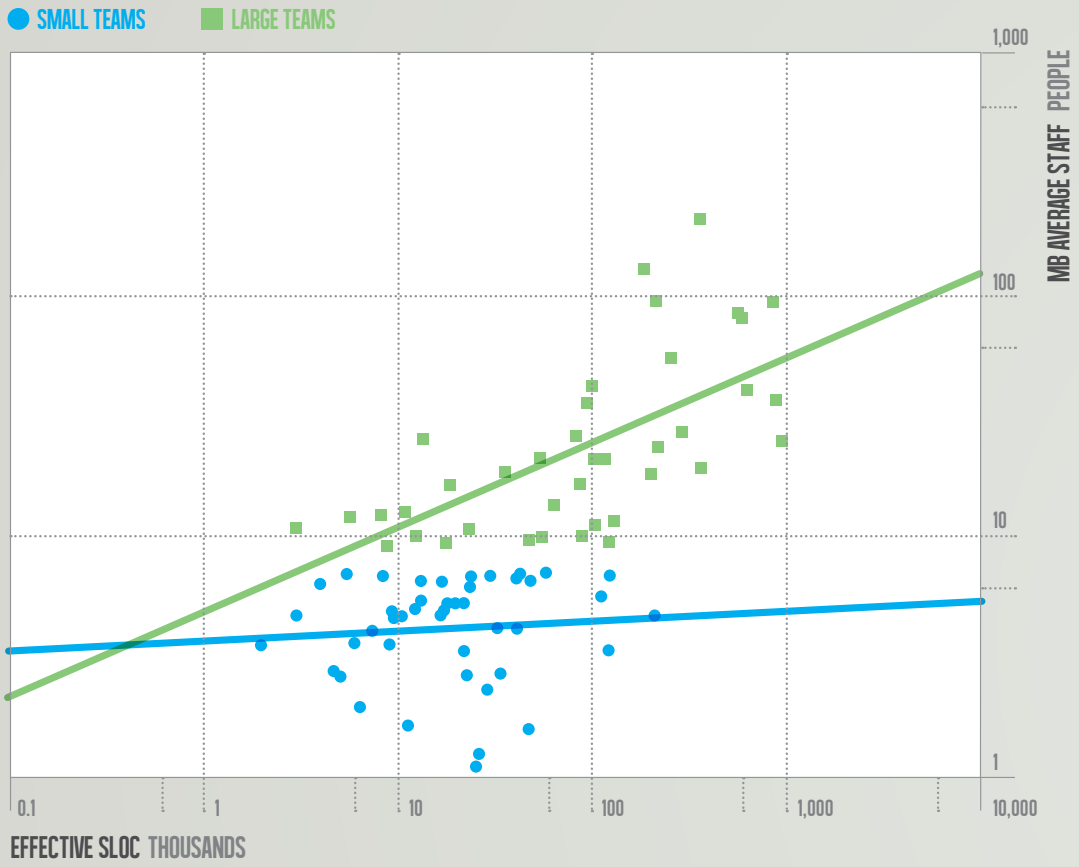
Agile vs. Non-Agile

- Duration of Agile projects was mostly lower on projects larger than 14,000 lines of code
- Number of defects created were lower on Agile projects than non-Agile projects

> "Large staff" projects cost significantly more, achieved negligible time savings, and produced many more defects than "small staff" projects.

MB Average Staff VS Effective SLOC

Figure 1: Plotting the number of lines of source code (SLOC) against the average staff sizes of "small staff" and "large staff" Agile projects.



So, if our question is whether Agile projects can scale, the answer would seem to be yes. As the projects we analyzed grew in size, Agile methods produced: shorter project durations, fewer errors, and higher productivity.

However, we also learned that as Agile projects grow, they tend to expend more effort and use more staff. And that's a problem, because we know from separate analyses (e.g., the QSM Almanac study in 2005) that high staffing levels correlate strongly with waste, regardless of development methodology.

> "Large staff" projects keep adding more people as they add functionality, but the "small staff" projects do not.

We also know that government IT departments aren't exactly flush with cash these days, and that hiring an army of Agile developers is probably not in the budget.

So just for fun (yes, this is what we do for fun), we decided to take a closer look at Agile projects in relation to staff size. Projects with 7 or fewer staff members we classified as "small staff," and projects with more than 9 staff members we classified as "large staff." When we compared the two groups, here's what we found:

"Large staff" projects cost significantly more, achieved negligible time savings, and produced many more defects than "small staff" projects.

That's fairly definitive. But does it hold true even at maximum scale? 80,000 lines of

code? 200,000? 5 million? The answer seems to be yes. And here's why:

As Figure 1 shows, the "large staff" projects keep adding more people as they add functionality, but the "small staff" projects do not. (As functionality increases, "small staff" projects add fewer than one full-time employee, whereas "large staff" projects grow fifty times in staff size.)

What this suggests is that large-scale Agile projects may not require big staffs after all. (Staffing up is just how most IT managers reflexively respond.) If, instead, we kept our Agile staff small, the data indicates we could complete large-scale enterprise initiatives much more efficiently, in nearly the same time frame.

This is phenomenal news for government, particularly during the dog days of sequestration—when keeping IT teams small and lean is worth its weight in U.S. bonds. Using this method, government organizations can keep costs down by influencing the way their contractors staff development projects, while at the same time producing more reliable products.

We must be careful, however, not to view Agile as government's panacea. If our data shows anything, it's that Agile projects are highly variable. We've seen how adopting Agile methods can potentially lead to modest schedule compression and fewer errors, but also a tendency toward bigger staff size and higher costs at scale.

So the question becomes: With all of these variables in play, how will government IT managers know when it's the right time—or the wrong time—to implement Agile methodologies?

The answer, we believe, is to do exactly what we've just done—run the numbers.

Agile practitioners might try to minimize upfront planning, but in large-scale government initiatives, there will always be a need for some semblance of pre-project assessment (even if it's done in an iterative, agile fashion). In fact, one reason government IT managers have been reluctant to adopt Agile methods is for fear of losing that planning and predictability.

However, this is where software estimation and forecasting tools can bridge the gap. In a recent *Harvard Business Review* article ("Why Your IT Project May Be Riskier Than You Think"), Bent Flyvbjerg and Alexander Budzier write:

"[Smart managers] break big projects down into ones of limited size, complexity, and duration; recognize and make contingency plans to deal with unavoidable risks; and avail themselves of the best possible forecasting techniques..."

But to forecast effectively, you need data—and lots of it—on past and present Agile development projects.

Because the truth is, Agile development in government is here for the long haul. [MG]

"We must be careful, however, not to view Agile as government's panacea. If our data shows anything, it's that Agile projects are highly variable.