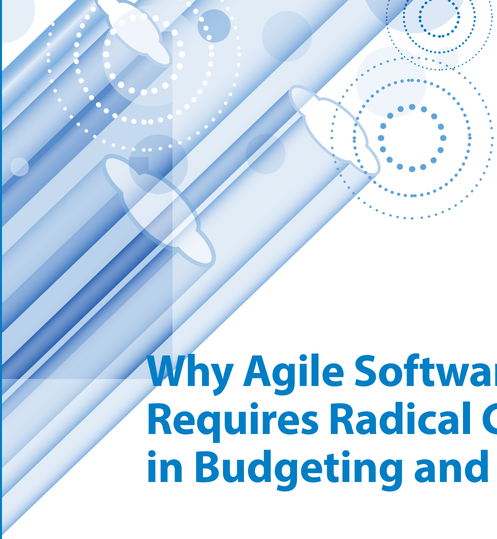


Why Agile Software Development Requires Radical Changes in Budgeting and Scoping

Authors

Nidhi Srivastava
Global Head of IT Consulting, Tata Consultancy Services

Apala Mukherjee
Practice Head, Tata Consultancy Services

Somnath Ghosh
Technical Architect, Tata Consultancy Services

With digitally savvy disruptors seemingly making inroads in every sector, companies using so-called agile approaches to building new digital products and digital processes are becoming hard to beat. In just one example, the speed with which Amazon.com unveils new capabilities to its website—dozens of changes every day—has left many longstanding retailers in the dust or racing furiously to catch up. Even large financial services companies like ING of the Netherlands, have realized that they must organize and develop online systems in agile ways to keep pace with the explosion of online banking options.

However, in the last couple of years, companies that have mastered agile development have been leaping ahead in industries in which having excellent digital connections to customers is crucial to success. (Think financial services, high tech, retailing, and media.) It's why 94% of companies are using agile approaches, according to a 2016 survey.[51] The two biggest reasons they cited were to accelerate the delivery of software and to better manage changing priorities.

Companies that introduce new online processes or online products for customers are chasing competitors that can make changes to their online offerings weekly or even daily. If they used traditional approaches to building such systems, they would come to market way too late and (most likely) off target.

Yet not all agile projects are successful, and a key reason is how they are scoped and funded. Companies that scope and budget their agile development projects the same way they do their traditional systems projects are likely to come up far short in two ways:

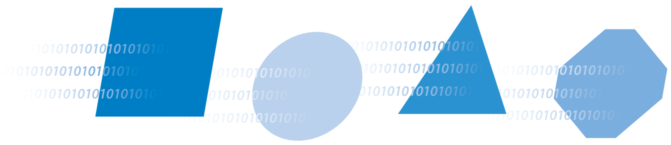
1. They lock development teams into working on features of systems that may turn out to be not worth investing in (or, at least, spending as called for in the original plan) because customers don't find them valuable. In this case, traditional budgeting doesn't take advantage of the ability of agile teams to put products online and test them with customers early in the process. Getting early feedback from customers can save a company a lot of money. A firm can redirect funding away from less valuable to more valuable features.
2. They risk losing key members of their development teams, who are often assigned to other systems while the current projects are waiting for the next round of funding.

Research has found that nearly two-thirds of the features built into systems using traditional development methods are rarely or never used, and that the 20% that are used provide 80% of the value to customers.[52]

[51] VersionOne survey conducted in the second half of 2016, Accessed July 20, 2017, https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2

[52] Standish Group report, 2002, Accessed July 31, 2017, http://www.featuredrivendevelopment.com/node/614

In this article, we discuss why scoping and budgeting of software projects delivered through agile development methods must be different than those used in traditional software development approaches. We explain why agile methods should initially provide a high-level, rather than a detailed, estimate of the time and the scope of effort required to deliver a system. Over time, as they build and test components with customers, the team will get a much more accurate and detailed estimate of time, scope, and budget. We also show why companies should fund agile programs not by their scope but rather by the value that their products bring to end customers or users (also known as "value streams").



The Rise of Agile Techniques for a Digital Online World

Agile software development methods arrived at the turn of the 21st century, a time when companies had to build e-commerce, websites and other systems to compete for customers on the Internet. With the advent of the web in the mid-1990s, large companies had to develop systems that would be outward-facing—i.e., used by end customers outside their organizations—not just by their internal users. Suddenly, the traditional systems development process (often referred to as using the "waterfall" method to depict how the phases of requirements analysis, architecture, coding, testing and launch would follow one another across a calendar), which reliably produced internal systems, had become too slow, off-target and in other ways inadequate.[53]

A variety of agile methods arose: Scrum, Lean, Extreme Programming, Kanban, and others. They were different answers to the very same need—to produce software much faster, try it out on real customers far earlier (building systems for the web made that easy, since customers could use the system from their own computers and digital devices), and (based on that early customer feedback) allow developers to make quick adjustments to the system so it wasn't obsolete the moment it went live a year or two later.

[53] For a history of agile software development, read how its founding fathers convened in 2000, Accessed August 04, 2017, http://agilemanifesto.org/history.html

Agile methods differed dramatically from traditional development in at least five ways: scoping, schedule, cost estimates, deliverables and key success factors. To scope systems developed through traditional methods, teams are asked to make detailed estimates of project length and budget after doing extensive analysis of the system's requirements. The more features and functions, and the more connections to other systems there are, the greater the system complexity and therefore, more people, time, and budget to do all the work.

In other words, for traditional systems development, the schedule depends on the scope of the effort estimated, and the cost is a function of that scope, and when the company needs the system to go live (i.e., the schedule). (Building a system faster requires assigning more team members, and thus increases the cost.) Yet developers know the least about the scope (and thus the time and cost) of a software project at the beginning. Their knowledge increases once they start delivering code, and it keeps improving the more they deliver.

The key to successful software projects based on waterfall methods is doing exceptional upfront requirements analysis so that the schedule and cost turn out to be fairly accurate.

Dimensions of Comparison	Traditional Development (Waterfall)	Agile Development
Scope	Detailed estimations based on extensive upfront requirements analysis of the whole system	High-level estimation upfront; scope is flexible, and depends on how much work can be done for a fixed time and cost
Schedule	Depends on scope of effort	Fixed
Cost	Based on detailed upfront estimate of system scope and schedule	Fixed
Deliverables	Based on project stage: requirements analysis, code, testing	Working product with value for customers/users
Key Success Factors	Doing upfront requirements analysis so that systems behave as per pre-defined business needs	Keeping teams intact so key knowledge remains in place over the product lifecycle Changes are made with an "inspect and adapt" approach
Development Team Credo	"Tell us what you need, and we'll tell you how long it will take and how much it will cost"	"Tell us when you need something and your budget, and we'll tell you what we can deliver"

Figure 6: Comparing Agile and Traditional Software Development

Agile development is built on this premise: If developers can release pieces of a system to users early in the development process, those developers will have a much better idea early in their process about the scope of the system. Those pieces are completed in short time frames—weeks, rather than months—in "plan > do > check > act" cycles.

In agile development, scoping, scheduling and cost estimation techniques are thus very different than they are in waterfall development. (See Figure 6)

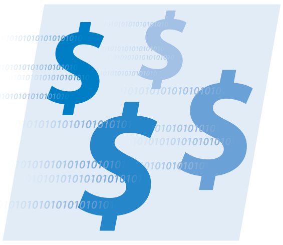
The Problems of Software Budgeting in an Agile World

So here we are, 16 years after the Agile Manifesto emerged from the primordial software development soup. Yet most organizations (while embracing agile) are still scoping and budgeting them in the same way they structured and budgeted the big, monolithic systems projects of yesteryear. In other words, while agile development techniques are all the rage in the IT organization and the business functions they automate, the budget holder for technology in most companies (the finance organization) still doles out funding the old-fashioned way: by program, projects or stage gates (such as requirements, design, coding, testing, and so on). After requirements analysis is deemed complete, funds are released for the next phase (system architecture and development). Then once that's done, funding needs to be approved for system testing, and so on.

For software development teams that build systems using agile methods, this has become a big problem for several reasons:

- It locks teams into continuing to develop pieces of software that may not be useful to customers.
- Teams get disbanded if project funding dries up. Valuable knowledge about the business processes of the product and technologies that enable those processes is lost.

The Answer: Scoping Smaller Releases and Funding Them Faster



So how should companies that need to develop systems through agile methods rethink their scoping and budgeting? We believe they should make three key changes: funding by “value stream” rather than by the entire program; launching low-cost, high-value products/features first and high-cost, low-value ones last; and continuously funding and giving the development team work that’s been prioritized thereby ensuring that the teams aren’t on hiatus.

Let’s look at each change.

Funding by value stream. The work of agile development teams should be funded with an economic view—i.e., how useful the product or feature is to customers or other end users. Let’s say a retailer’s top management gives its new e-commerce website a full year to bring it to market. Developed through traditional methods, none of the system would see the light of day (i.e., be online for customers to use) for that year. However, using agile methods, e-commerce and IT managers would prioritize the features of the system (i.e., price lookup, product descriptions, product reviews, one-click pricing, inventory checks in store locations, and so on) and begin putting them online far earlier—perhaps within weeks after they launched the project depending on the release cadence.

Funding this project the traditional way would have called for drawing on its total budget at each step of the development process: at the start, for requirements gathering and analysis; then (after requirements were inspected) before architecture and coding; then (after the architecture and coding were checked) before testing; and finally (if the testing went well) to put it into production. Or top management would allocate budget for the full scope of the project, tying up funds for a long duration regardless of whether the project meets its intended purpose or not when it is finally completed.

Launching high-value but high-cost value streams early. One of the biggest advantages that agile has over traditional systems building methods is the ability to break a big monolithic system into smaller pieces, and then go live with some of those pieces very early in the project, even in the first few months. The pieces you want to go live with first are those that everyone believes at the outset to be low-cost, yet high-return.

How budgeting of agile projects needs to change during the software development lifecycle

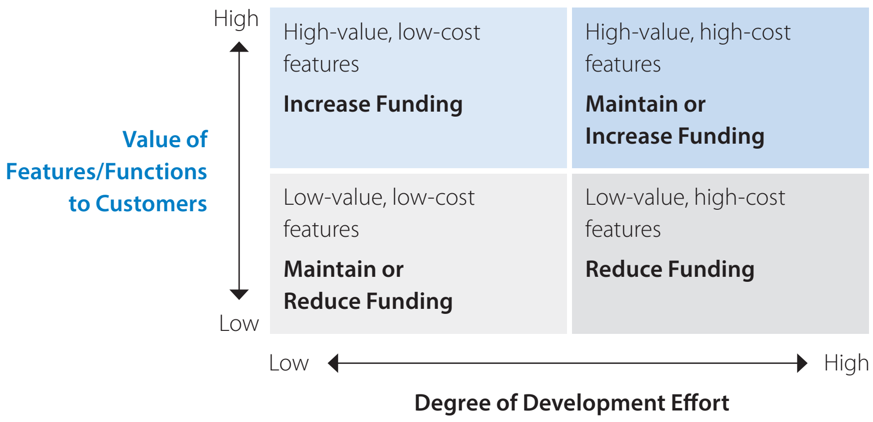

High
High-value, low-cost features
Increase Funding
High-value, high-cost features
Maintain or Increase Funding
Value of Features/Functions to Customers
Low-value, low-cost features
Maintain or Reduce Funding
Low-value, high-cost features
Reduce Funding
Low
Low
High
Degree of Development Effort

Figure 7: Adjusting the Budgets for Systems Built Through Agile Methods

The market feedback that the hypothetical retailer would get from introducing pieces of its new e-commerce website in weeks like those we've discussed would be highly valuable. It will reduce further funding of low-value but high-cost features. It will continue funding of high-value features. And it should increase funding for features proven to have high value but which don't cost very much. (See Figure 7)

Delegating scoping and budgeting to the agile development teams. The whole reason to use agile development is to come to market far faster with a new online system, and get much earlier and frequent customer feedback on which features of the system to further develop or abandon. Hence the funding decision for agile development needs to be top down, as well as dynamic.

As such, we believe that making all this work well is best done when a company delegates the decisions on what pieces of the system to work on next to the product owners in the business (who best understand their product and its performance) and to the development teams (who best understand how long it would take to build it).

Of course, vesting that decision in the development team requires great trust by those who fund the system: the business unit, finance and IT heads. Your CFO must become comfortable with the notion that agile teams (which must include product managers/owner(s) from the business units and functions) will know best what system features are delivering value or not.

It's why the team, product owners, and managers must keep reviewing what's being delivered and monitor its performance in the marketplace to utilize available funding most judiciously.

We don't advise companies that are new to agile development to change the way they budget overnight. They need to first see the advantages of agile development and get accomplished at it. But those that are proficient with agile should consider giving their teams fixed budgets and then leave the scoping decisions to the team.

Ultimately, as leadership guru Stephen M.R. Covey once wrote about trust in business: whether the agile teams gain and keep the trust of senior executives will depend on the results they generate and their integrity, intent, and capabilities.[54]

[54] The Speed of Trust, Accessed August 04, 2017, http://www.speedoftrust.com/How-The-Speed-of-Trust-works/book

How This is Working at a Major Retailer

The retailer, whose name will go unmentioned, is competing against Amazon, as are numerous other retailers these days. But it realized in the last few years that how often it could update its e-commerce website, the systems in its stores, merchandising, buying, and other system were critical to competing effectively in a bricks-and-clicks world.

At this retailer, portfolio level budgeting is done top-down with joint decision by business and IT. Highest priority value streams[55] get fixed funding for a fixed period without any elaborate bottom-up cost estimations based on detailed requirements upfront.

In fact, the company found that it had to change the organization structure of the IT department to make agile methods work well. So rather than being organized around IT domains such as business analysis, coding, systems analyst, testing, and so on, the new organization is structured by online "product" area: point of sales applications, HR applications, vendor negotiation applications, and so forth. Agile teams in each product area have the requisite business analysis, systems analysis, UI designers, developers, testers, and other skills necessary to build, test, and go live with an online product in their area.

The result of embracing agile, and new ways to scope and budget for agile-developed systems, has been sizable: faster system rollouts, fewer technology incidents, and a workplace that lures new systems engineering talent to the company.

The New World of Systems Scoping and Budgeting

Companies like the retailer mentioned above, ING, Netflix, Spotify, and Google are leading the way on how systems scoping and budgeting must change in a world of digital systems built through agile methods. Those companies that embrace and master agile methods, and then scope and budget for them in ways that provide highly valuable early feedback, are getting a big leg up in the race to be digital.

[55] SAFe® for Lean Enterprises, Accessed August 04, 2017, http://www.scaledagileframework.com/