McKinsey&Company
Operations Extranet

Teaching elephants to dance (part 1)

Empowering large organizations with agile software development

by Peter Andén, Santiago Comella-Dorda, André Rocha and Tobias Strålin

February 2015

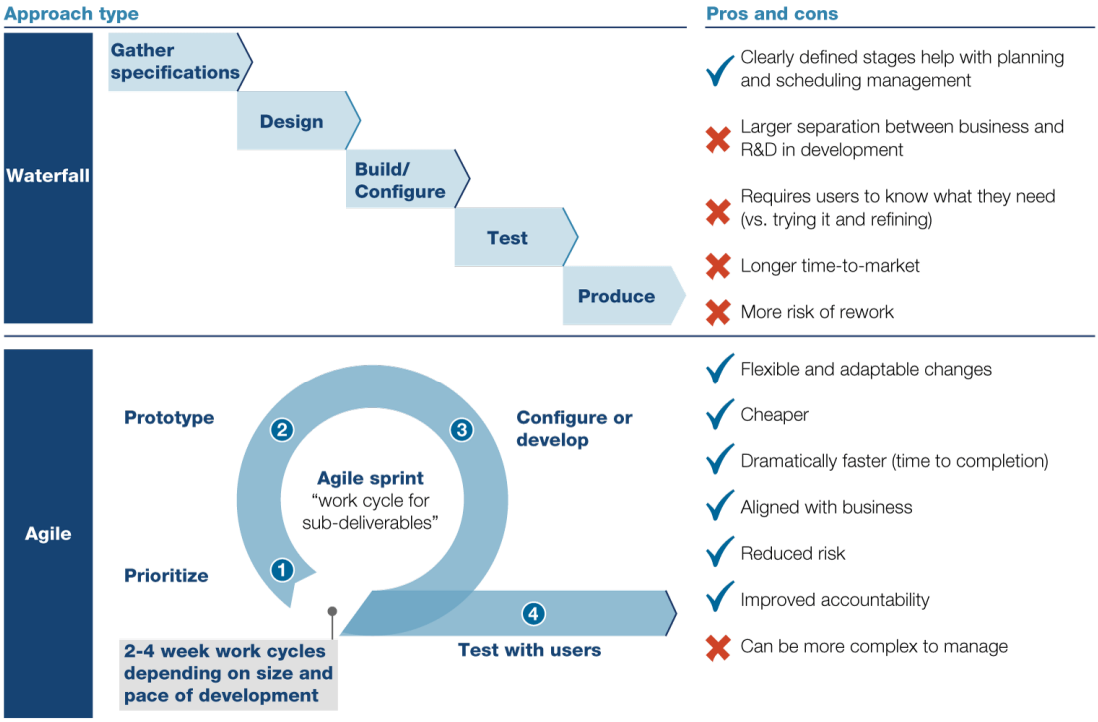
Getting agile development right and at scale requires new processes, governance models, capabilities and mindsets.

Good software is hard to build. The history of software development is full of projects that took too long, cost too much or failed to perform as expected. Agile software development methods emphasize tight collaboration between developers and their customers, rapid prototyping, and continuous testing and review (see sidebar), and have evolved as a direct response to these issues.

Fans of the agile approach say that it improves many aspects of their software development processes, such as an increased ability to handle changing customer priorities, better developer productivity, higher quality, reduced risk and improved team morale. In an effort to quantify these benefits, we made use of the McKinsey Numetrics database.[1] This proprietary benchmark contains data on the approach, costs and outcomes of more than 1,300 software projects of different sizes, from different industries and using different programming languages. When we compared the cost, schedule compliance and quality performance of the 500 or so projects that used agile methods with those that applied the “waterfall” methodology, the agile projects demonstrated 27 percent higher productivity, 30 percent less schedule slip and three times fewer residual defects at launch (Exhibit 1).

[1] McKinsey Numetrics is a proprietary benchmark containing data on the approach, costs and outcomes of more than 1,300 software projects of different sizes, from different industries and using different programming languages. See more at http://www.mckinsey.com/client_service/semiconductors/tools_and_solutions

Exhibit 1: Agile software development has a number of advantages of conventional methods



Source: McKinsey analysis

Old habits die hard

For many companies, however, the move to agile development is a significant cultural shift. Not all organizations succeed in the transition. Numetrics data indicates that top quartile agile projects are three times as productive as those in the bottom quartile, for example. In one survey of software development organizations, almost a quarter of respondents said that the majority of their agile projects had been unsuccessful[2]. Asked to pinpoint the root cause of their problems, respondents most often cited a corporate culture that was not compatible with agile methods, or a lack of experience in the use of those methods.

When we talk to senior executives about the potential benefits and risks of adopting the agile approach, three questions commonly arise:

1. How can we modify the agile approach to work in a large, complex organization like ours?

2. Can we apply agile techniques across all layers of the software stack, or just in end-user applications?

3. How do we roll out the agile approach across our organization?

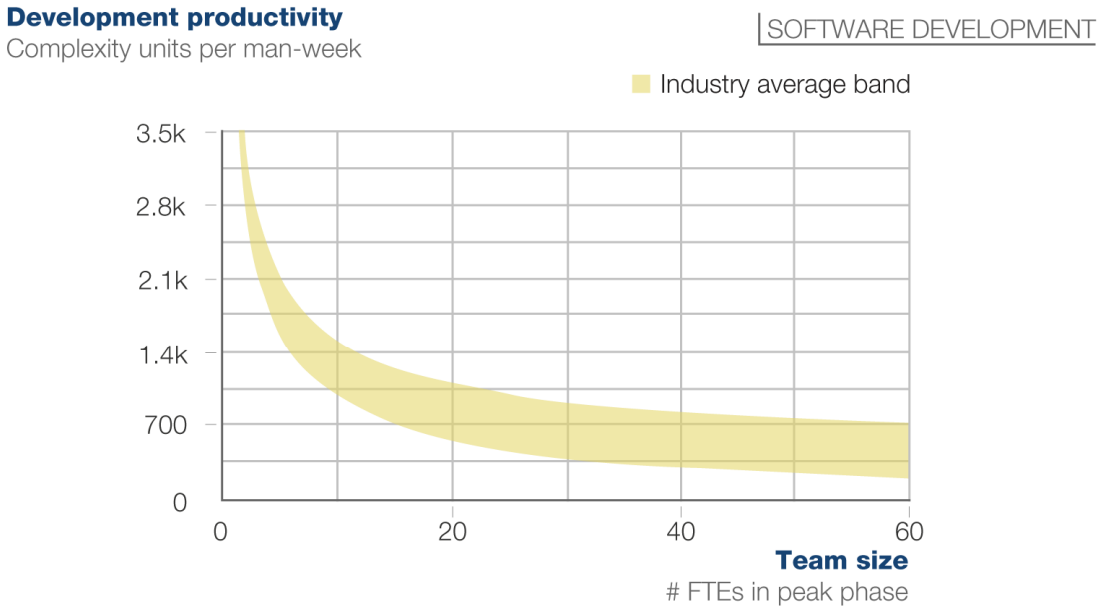
Let's look at each in turn.

[2] Source: "State of Agile development" 2010 survey by VersionOne

Making agile work in large projects and large organizations

The agile approach is compelling in its simplicity: one team, reporting to one product owner, conducts one project. To maintain the key benefits of flexibility, clear communication and close collaboration, the best agile teams are small: usually five to ten members. Translating that structure to large, enterprise-level software projects with many tens or hundreds of developers can be tricky. As development teams get bigger, they quickly become unwieldy as progress slows, communication gets difficult and the benefits of agility evaporate. Exhibit 2, again based on McKinsey Numetrics data, shows just how fast the productivity of individual developers drops away as the size of their team increases.

Exhibit 2: The impact of team size on software development productivity



Source: McKinsey analysis; July 2010 State of the IT Union Survey, www.ambysoft.com/surveys/; Client experience; Scott Ambler, “Supersize me” in Dr. Dobb’s. March 01 2006

A more effective approach is to maintain the small size and working characteristics of the core agile teams, and to adopt an architecture, organizational approach and coordination process that shield those teams from additional complexity. The first step in this approach is robust modular software architecture with clear interfaces and dependencies between modules. For this purpose, many organizations choose to have a team of architects moving ahead of, and laying the groundwork for, their development teams.

Often, however, organizations have to deal with a large legacy code base, making modularizing every technical component challenging. In these situations, leading organizations are adopting a two-speed

architecture, so that certain elements of the stack are modularized to enable agile developments, while legacy elements are encapsulated with a relatively stable API layer.

With these foundations in place, individual teams can then work on their own part of the product, with multiple teams reporting to a common product owner (PO). The PO will be responsible for managing the "backlog" of features, work packages and change requests and for coordinating the release of product versions. A separate integration team will help the PO make optimal short-term planning decisions as customer requirements and dependencies change. Development work takes place in two to four week "sprints" during which the agile teams operate with a high degree of independence. Between sprints, a joint product demo, sprint retrospective and planning stage ensure all teams maintain coordination.

Regardless of the size of the organization or the project, one tenet of agile development remains true: it is good for all developers to work at a single location. The benefits from improved communication and ease of collaboration that arise from close physical proximity are hard to overstate. While some organizations do manage to run effective distributed development teams, analysis of our software project database reveals that organizations pay an average productivity penalty of 15 percent for each additional development site they use. The change from one site to two incurs the largest productivity drop: organizations with two development sites in our database were 25 percent less productive on average than those with just one. It is no coincidence that many of the world's best software development organizations chose to concentrate more than 90 percent of their software developers in a single location.

Tailoring agile across the software stack

Many of the biggest benefits of the agile approach arise from close cooperation between developers and the end customer. As a result, the development of end user applications has been the principal focus of many agile efforts and organizations often find it easiest to implement agile methods in the development of their own application layers. Most software systems are built as a stack, however, with a hardware integration layer, and one or several middleware layers below the application layer. Companies sometimes struggle to understand how they should best apply agile techniques, if at all, to the development of these lower level layers.

In practice, the most successful organizations take a selective approach. They pick and adapt a specific subset of agile tools and techniques for each layer in their stack, and they alter their development approach to take account of the way agile is applied in the layers above and below.

Middleware development, for example, with its slower evolution of requirements and emphasis on standardization, can lend itself to an approach in which individual development cycles or sprints are longer than in application development. While doing this, companies should

strive for a pace that is synchronized with the sprints in the higher layers. Middleware teams will also take extra steps to ensure their test cases reflect the impact of rapidly changing applications driven by faster moving agile teams.

At the level of hardware adaptation, however, freezing requirements early remains a priority, to allow sufficient time for hardware development. Here an organization may still find it beneficial to pick specific agile tools, like continuous integration, test automation and regular production of prototypes, to capture the benefits in productivity, time-to-market and quality they provide.

Rolling out agile development

For many organizations, the shift to agile software development represents a significant change in approach and culture. Like all large-scale organizational change, a successful transformation requires care in planning, execution and on-going support.

Most organizations begin their change journey by assessing their current practices and developing a blueprint for improvement. This blueprint will define all the extra capabilities, new management processes and additional tools the organization will need. These may include extra training for developers, investment in test automation infrastructure and a clear approach for the management of release cycles, for example. The adoption of agile methods will have implications that go far beyond the software development function. These must be taken into account during the development of the blueprint. Companies can engage the wider organization in a variety of ways, from conversations with leaders in other functions to crowd-sourcing-style suggestion schemes.

The blueprint is validated and refined using a targeted pilot in one or a few teams. This pilot serves several functions. It helps the organization identify and adapt the tools and techniques that best suit its needs. It helps developers, product owners and managers to develop the skills they will need to apply, and teach, in the wider roll out. And it serves as a demonstration to the rest of the organization of the potential power of the approach.

As an organization moves from its initial pilot phase towards a wider roll-out, the need for a long term perspective and strong top management support is particularly critical. This is because most companies experience an initial drop in productivity as their developers, product owners and managers get used to the new way of working. Analysis of our benchmarking data suggests that this drop is usually around 14 percent at its deepest before productivity recovers and goes on to surpass pre-agile levels by 27 percent or more.

Beyond preparing themselves for the inevitable initial dip in productivity, the best organizations take steps to preempt it. One key step is ensuring that the right engineering practices, capabilities, tools and performance management mechanisms are in place as the rollout commences—the subject of the second article in this series.

* * *

Adopting the approach described in this and part two of this article has delivered remarkable software development performance improvements to some large companies. One North American development organization, for example, created a modified version of the agile "scrum" approach to improve the reliability of mission-critical software delivery. The company rolled out the approach to more than 3,000 developers using a large-scale change management program including training, coaching, communication and role modeling. The results of this effort exceed the company's original expectations, with throughput increasing by more than 20 percent, significant cycle time reductions and higher customer satisfaction.

Applying agile in larger and more complex efforts requires modifications to standard agile methodologies. Specifically, organizations need to create structures and practices to facilitate coordination across large programs, define integration approaches, ensure appropriate verification and validation, manage interfaces with other with enterprise processes (e.g., planning and budgeting) and engage with other functions (e.g., security and infrastructure)■

About the authors: Peter Andén is a principal in McKinsey's Stockholm office, Santiago Comella-Dorda is a principal in the Boston office, André Rocha is a consultant in the Munich office and Tobias Strålin is a principal in the Seattle office.

The authors wish to thank Ulrich Naeher and Florian Weig for their contributions to this article.

Sidebar: Agile development

The term **agile** describes a collection of **rapid, iterative** software development approaches. Agile involves a wide variety of tools and techniques, with certain common elements at its heart. Tightly integrated **cross-functional teams** that include end-customer representatives ensure the product reflects real user needs. Those teams do their work in **rapid iterations** to refine requirements and weed out errors, with **continuous testing and integration** of their code into one main branch. Techniques like co-location of project teams, daily meetings, pair programming and collective ownership of code promote **collaboration** within the team.