McKinsey&Company
Operations Extranet

Teaching elephants to dance (part 2)

Empowering large organizations with agile software development

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

February 2015

In big companies, lightweight development methodologies require heavyweight support behind the scenes for maximum benefits and minimal cost

In **part one** of this series, we showed how the agile development methodology produces better, more reliable software, faster and more cheaply than traditional approaches. Capturing such benefits in large organizations, however, has a price: it requires companies to establish additional governance structures and accept a short-term loss of developer productivity. In this article we look at the key practices that large companies must master to lessen the challenges of managing agile development and deliver the most value the approach can provide.

All product development processes have the same broad objectives: to deliver the right features to the customer at the right time and at the right quality, all at the lowest possible cost. For companies implementing agile at scale, doing that requires the right design choices, practices and processes in four broad categories:

1. **What they develop**: deciding on the right portfolio strategy, the right architecture and the right scope for each product.
2. **How they develop**: picking the right methods and approaches during the development process.
3. **Where they develop**: choosing the right organizational design and footprint.
4. **Enabling the development**: implementing the capabilities, tools and performance management systems to enable teams to perform at their best.

Let's look at some example best practices for each of these areas.

What: Developing the right software

Architecture

A robust modular architecture is essential to the success of large-scale agile projects. Modularity aids the division of work between teams, minimizes conflicts and facilitates the continuous integration of new code into the product for testing and evaluation. Under the agile methodology, the rapid evolution of the product during development sprints makes it difficult to fully define such architecture upfront. Instead, leading companies reserve development capacity specifically for modularization work. This means developers can concentrate on the delivery of features and refine the surrounding architecture as they go.

In addition to agile teams' work on the architecture of their own modules, a dedicated cross-project architecture team can help to manage the major interfaces and dependencies between modules.

To minimize dependencies and the resulting waiting times, agile teams are usually formed around the delivery of features (rather than around modules or components). It is likely, then, that individual teams will end up working on multiple modules, with more than one team potentially contributing the development of individual modules. While this approach creates the possibility of code conflicts, which must be managed, it also promotes close coordination between teams and encourages the development of simpler interfaces between modules. Leading companies consider the trade-offs between simplicity and the potential for conflicts when allocating development resources, so some particularly complex or critical modules may get their own dedicated teams.

Requirements management

Adherence to the agile principle of simplicity helps to ensure that individual teams remain focused on goals. Leading companies use the idea that each agile team has only one backlog and one product owner to avoid duplication, conflicts or "scope creep." By doing this, they ensure that product requirements are clear and responsibility for the delivery of each requirement is ultimately allocated to a single team.

The product owner (PO) plays a critical role in this process. During early stage testing, for example, the PO will collect and filter feedback from end users, decide which requests should be added to the backlog of feature requests and allocate those requests to the appropriate teams. Typically, the backlog is structured along several different levels of granularity, starting from the original requirement as formulated by the end users and subsequently broken down into smaller and more detailed requirements, often called user stories. To enable coordination across multiple teams, PO organizations often follow a similar structure with a hierarchy of POs, senior POs and chief POs owning backlogs at the team level, program release level and portfolio or suite level.

How: Using the right methods

Test-driven development

The rapid, iterative nature of agile development makes maintaining quality a challenge. Companies who do this well are adopting test-driven development methods that help them accelerate the development process by increasing the chance that the software is right the first time.

Under the test-driven development approach, agile teams begin by writing test cases for the specific features they are implementing. Then they write product code to pass those tests. Through the development process, the tests are updated alongside the code, and every iteration must pass the test prior to release.

Beyond accelerating the test process, test-driven development has a number of other advantages. It helps make requirements clearer and more specific, since they must be built into the test protocols. It enables real-time reporting of the progress of the whole project, since managers can check the number of tests passed at any one time. It encourages teams to write simpler and more rigorous code that focuses on meeting user requirements. Finally, the availability of the test protocols simplifies future updating and code maintenance activities.

Continuous integration

Early and regular testing requires access to the latest version of the product under development. To avoid labor-intensive and potentially error-prone manual recompiling and rebuilding, best-practice companies support their modular architecture with a continuous integration infrastructure that makes regular (daily or every few hours) builds of the product for testing, and use the latest version of code released by the development teams.

Some very mature agile development organizations will make these daily builds of their product available directly to customers with the confidence that their test-driven development and continuous integration processes will ensure sufficient quality and reliability. Such a rapid release cycle is not always desirable, however. It is more common for the PO to make a release only when there is sufficient new functionality available in the product. In addition, some organizations have one or two “hardening” sprints before each scheduled release, in which teams focus on improving product quality and performance, rather than adding new features.

The systematic use of agile practices like continuous integration and test-driven development leads to quantifiable benefits in the quality of the software developed. Data from the McKinsey Numetrics[1] benchmark of more than 1,300 software projects shows that software produced in agile projects has on average a residual defect density which is three times lower than software produced using other methodologies.

Protect the teams

In large and complex projects, development teams can easily be distracted by requests from users, managers and other teams. Taking steps to minimize such distractions during development sprints allows teams to focus on achieving the objectives of the sprint. At the top of this shielding infrastructure is usually the PO, who prioritizes requests for new features or product changes. The PO will be assisted by a dispatch team, which is responsible for the incoming stream of bug reports and minor change requests arriving from users, field test teams and other stakeholders. The dispatch team will eliminate duplicate requests and validate, categorize and prioritize issues before adding them to the backlog and allocating them to the appropriate team. In

[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

many project organizations, the project manager will collaborate with the PO and support shielding the teams.

Manage interfaces with the wider organization

In order to effectively protect their development teams, the best companies manage the various essential interactions between the teams and the rest of the organization. This includes staffing teams appropriately from the beginning, so they have all the capabilities they need to complete their current sprints without additional resources. It also involves procedures to ensure teams complete all the testing and documentation required to comply with corporate standards and security requirements.

Finally, companies establish clear interfaces with relevant parts of the wider organization, so development team members know where to go for advice on the company's graphic design standards, for example, or to check that products and features will meet legal requirements in all relevant markets. A useful construct is to appoint a Single Point Of Contact (SPOC) from each relevant organization. The SPOC is required to attend the release and sprint planning meetings and reviews to ensure appropriate coordination and engagement while limiting the additional load on central functions.

Where: Building the right organization

Team coordination

The need to keep multiple teams coordinated is the most significant difference between agile in large and small projects. Getting this right requires mechanisms for strong coordination and monitoring of code conflicts. Strong coordination starts by holding regular meetings within each agile team (typically every day) and among the various teams in a project (usually two to three times per week). To monitor and minimize code conflicts, two important steps can be taken. First, the product architects can monitor system interfaces and the impact of changes. Second, dedicated owners can be assigned to each product module to continuously monitor and assess the quality of the code.

Managing distributed teams

As we noted in the first part of this series, agile development works best when all developers sit at a single location. Close physical proximity facilitates communication and collaboration by allowing regular formal and informal meetings. In many organizations, however, such co-location is not possible. Their development teams may be sited in different countries, for example, or some parts of the development may be outsourced to external organizations.

To make agile work well in distributed environments, companies must make further modifications to core agile practices. Keeping multiple teams coordinated, for example, may require additional upfront planning prior to the start of development sprints. The sequence of development activities requires extra care too. Focusing early on aspects that will have significant implications for many teams, like the architecture of the product or its user interface, helps ensure consistency later on.

The best companies also work hard to facilitate communication between distributed teams. They do this using virtual communication tools like video-conferencing, and web-based document management and sharing tools. They also facilitate visits, exchanges and face-to-face meetings between teams where possible. Requiring more detailed documentation as part of each agile sprint also helps subsequent teams to understand and build on work done by distant colleagues. The dedicated effort required to make this documentation can be minimized with the use of automated documentation generators.

Partnering with other vendors

Even companies that run successful internal agile processes frequently fail to apply the same principles in their interactions with other vendors. Rather than investing time and effort negotiating traditional—and inflexible—contracts that aim to capture all the requirements of the project up front, some leading players are now working with external suppliers in the same way they do with their internal agile teams. By encouraging collaboration and a focus on output, this approach aligns internal and external development efforts, and promotes greater efficiency for all parties involved.

Enabling the development: Capabilities, tools and performance management

Capabilities

As they scale up their agile transformation, companies need to dedicate special attention to developing the right capabilities across their organization. Agile places new demands on software developers, who may have to learn to operate in a less specialized, more flexible, more self-reliant and collaborative environment.

Leading companies promote multi-skilling in their development teams. Typically, individual developers will have one or two core areas of expertise, but will also acquire skills in related areas. Multi-skilling helps the development teams adjust to inevitable workload changes and other skills required during the project. Combined with collective code ownership, it also helps different agile teams to work independently. Multi-skilling also works well for developers, giving them plenty of opportunities to upgrade and extend their skills, and it facilitates communication and collaboration with colleagues working in different areas. Last but not the least, having top-notch developers usually makes a big difference in the productivity of the team.

Agile places new demands on managers too, particularly product managers and R&D line managers, whose role under agile may change radically. Product managers need to operate much closer to the development engineers, prioritizing and explaining the work that needs to be done. Similarly, the most effective line managers in agile software development environments will focus on enabling their engineers to do what they are best at, which is to develop new products. The role of the line manager is to ensure that the development team holds the required capabilities, a high motivation level, and a strong “can-do” mind-set. Importantly, line managers also ensure there are no impediments slowing down or blocking the development progress. It is vitally important to support managers through this transition, but it is frequently ignored.

Good capability-building efforts make use of a range of methods, with classroom learning supported by extensive on-the job coaching, mentoring and support to reinforce the use of new practices.

Tools

A common development tool chain across all agile teams is an important element of effective project execution and control. This needs to be in place from the beginning of the agile rollout. Examples include technical tools for automated testing, quality analysis, configuration management, continuous integration, fault reporting and product backlog management systems.

With these tools, companies can mandate the adoption of certain agile methods right at the start of the transformation process. For example, they can ensure that testing takes place from the beginning of each development sprint, to catch as many issues as possible before code is released into production. They also are able to ensure code actually is released into production at the end of each sprint, to continue the rapid identification of issues.

Performance management

Good management isn’t all about IT systems and tools, however. Leading companies also make extensive use of visual management techniques—another core agile practice—with teams using whiteboards that show the status and pending actions associated with their current sprint. Not only can teams use these boards as the basis for their daily meetings, they also allow product owners and members of other teams to see what is going on at a glance.

Finally, companies need to balance the independence and flexibility of agile teams with the need to ensure the wider organization has a clear understanding of their progress, and can intervene to provide extra support when problems occur. A best practice is to do this by adopting standard systems and processes for performance management, as well as by using clear and closely tracked metrics, including measures of engineering productivity.

* * *

The agile approach has proved greatly effective in improving the speed, productivity and responsiveness of software development. Applying a methodology that was developed for small teams across a larger organization requires companies to make some specific changes and additions, however. Adopting tools and practices described here allows even the most complex development projects to capture identical benefits.■

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.