McKinsey&Company

JULY 2014

# Achieving success in large, complex software projects

**Using cross-functional teams to break down silos improves the chances of success when building highly complicated systems.**

**Sriram Chandrasekaran, Sauri Gudlavalleti, and Sanjay Kaniyar**

Large technology-led transformation programs are important for creating business value and building strategic capabilities across industries. With many organizations spending around 50 percent of their IT budget on application development, the ability to execute software programs faster and at lower cost is essential to success for many transformation projects. However, the quality of execution leaves much to be desired. A joint study by McKinsey and Oxford University found that large software projects on average run 66 percent over budget and 33 percent over schedule; as many as 17 percent of projects go so badly that they can threaten the very existence of the company.[1]

Some large-scale application-development projects are particularly challenging because of their complexity and high degree of interdependency among work streams. This category includes development of systems for telecommunications billing, insurance claims, tax payments, and core retail-banking platforms. These projects demand close coordination due to frequent refinements to the original user requirements.

Such coordination can only happen by breaking down the traditional silos in application development—an achievement often associated with the agile software-development approach. But agile is mainly applicable to smaller projects with minimal up-front definition of user requirements that can be cleanly divided into a number of parallel subprojects.[2]

[1] Michael Bloch, Sven Blumberg, and Jürgen Laartz, "Delivering large-scale IT projects on time, on budget, and on value," *McKinsey on Business Technology*, October 2012, mckinsey.com.

[2] In agile application development, each subproject can be handled by a team of six to ten people. The teams work in bursts of two to three weeks to define requirements on the go and deliver updated code in each burst.

**Takeaways**

Coordination is a common challenge in application development, particularly for large, complex projects.

Moving away from traditional silos and toward work cells can help.

These cross-functional units have many benefits, including increased accountability, better communication, and shorter iterations.

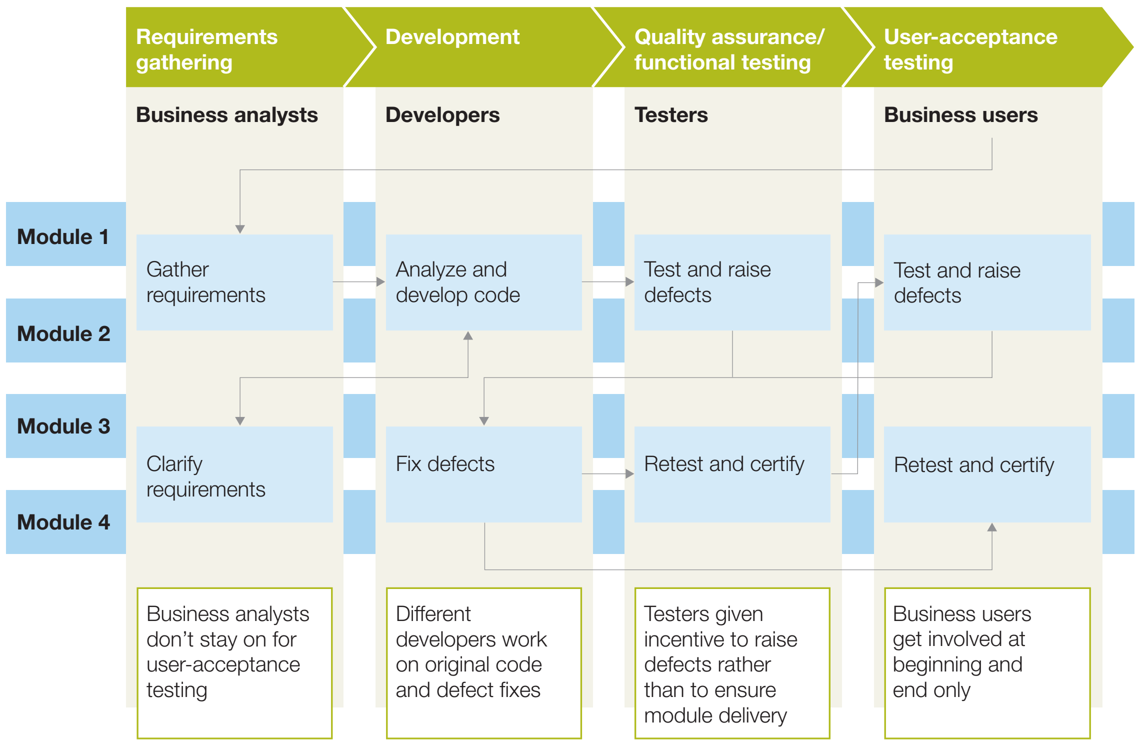
Elements of iterative application-development practices inspired by agile, lean,[3] and test-driven development[4] will certainly play roles in complex projects such as the ones mentioned above. However, in our experience, these approaches need to be combined with a new organizing construct featuring cross-functional teams. We call these teams "work cells."

## The coordination challenge in application development

The three disciplines involved in application-development projects—business analysis, development, and testing—often work in silos, with inefficient information flow between them (Exhibit 1). This is a minor issue in small application-development projects, but the communication problems grow larger in big, complex programs. The risk increases further when, as is often the case, project managers and business analysts, who gather user requirements for the applications, are located onshore while developers and testers are offshore. This slows communication because there's limited overlap of working hours between time zones. What's more, information is exchanged among disciplines in a hub-and-spoke manner. For example, the code defects identified by testers are assigned to a senior application developer, who then

Exhibit 1

**Traditional application-development teams are organized by function, with multiple handoffs.**



[3] Lean is an integrated system of principles, operating practices, and methods focused on getting the right things to the right place at the right time and in the right quantity while minimizing waste and being flexible and open to change.

[4] Test-driven development is an application-development practice where a developer writes unit tests for a piece of functionality before writing the code for the functionality.

In cross-functional teams, the role of the project manager becomes ensuring that cells deliver their application modules, rather than managing communications and handoffs.

assigns the coding rework to the rest of the team. These multiple handoffs can result in miscommunication and bottlenecks.

Lack of effective methodologies to measure productivity and quality adds to the challenge, resulting in expensive mismatches between demand and capacity, and in finger pointing among the disciplines. Development teams expect user requirements to be agreed upon and finalized when they receive them, which is not always the case. Rework and frustration within teams may result, as not all parties involved will be aligned on the latest requirements or clarification of requirements. As a result of operating in silos, work moves in lumps through the software-development life cycle. For example, all use cases are examined together in the user-acceptance testing phase, rather than in batches as they are completed.[5] This results in missed opportunities to perform processes in parallel and shorten the time to market.

We have found that for many large, complex application-development projects, functionally organized team structures are counterproductive. Each function takes ownership only of its part of the software-development life cycle instead of delivering working functionality to the end user. Given the communication challenge, the small team of project managers with end-to-end responsibility is often too stretched to coordinate across disciplines.

## The cross-functional approach

In our experience, large, complex software projects are better served by work cells—cross-functional teams with end-to-end ownership of application modules. The role of the project manager becomes ensuring that cells deliver their modules, rather than managing communications and handoffs between functional teams (Exhibit 2).

When applied well, cross-functional units can have multiple benefits, including increased individual and collective accountability, better communication and coordination, and shorter iterations.

**More accountability.** In a work cell, business analysts, developers, and testers work together as a tightly knit group and take responsibility for the whole process—definition of user requirements, development of code, functional testing, rework, user-acceptance testing, and the ultimate delivery of functionality to the customer. Such a team structure encourages a first-time-right ethic by increasing both individual and collective accountability.

**Better communication.** Cross-functional units reduce rework and delays that arise because of lack of coordination among disciplines. The complexity of a mix of onshore and offshore locations becomes easier to manage when requirement changes, updates,

[5] Use cases are a method for gathering the functional requirements of applications. For more information, see Michael Huskins, James Kaplan, and Krish Krishnakanthan, “Enhancing the efficiency and effectiveness of application development,” *McKinsey on Business Technology*, August 2013, mckinsey.com.

and clarifications happen within the unit rather than between functions. Finding and fixing defects will also be more efficient: members of the cross-functional unit will know which business analyst, developer, or tester to talk with and will be able to communicate directly. Team members may feel more empowered to give one another direct feedback, reducing the risk of error and the cost of rework. Schedule changes are communicated in a timely manner to ensure capacity is available for testing or rework. Sharing prerelease notes ahead of time gives enough information on what the testers are expected to test. A 15-minute daily huddle can help the unit discuss current work and align on priorities. In addition, each cross-functional unit may have daily or alternate-day planning meetings.

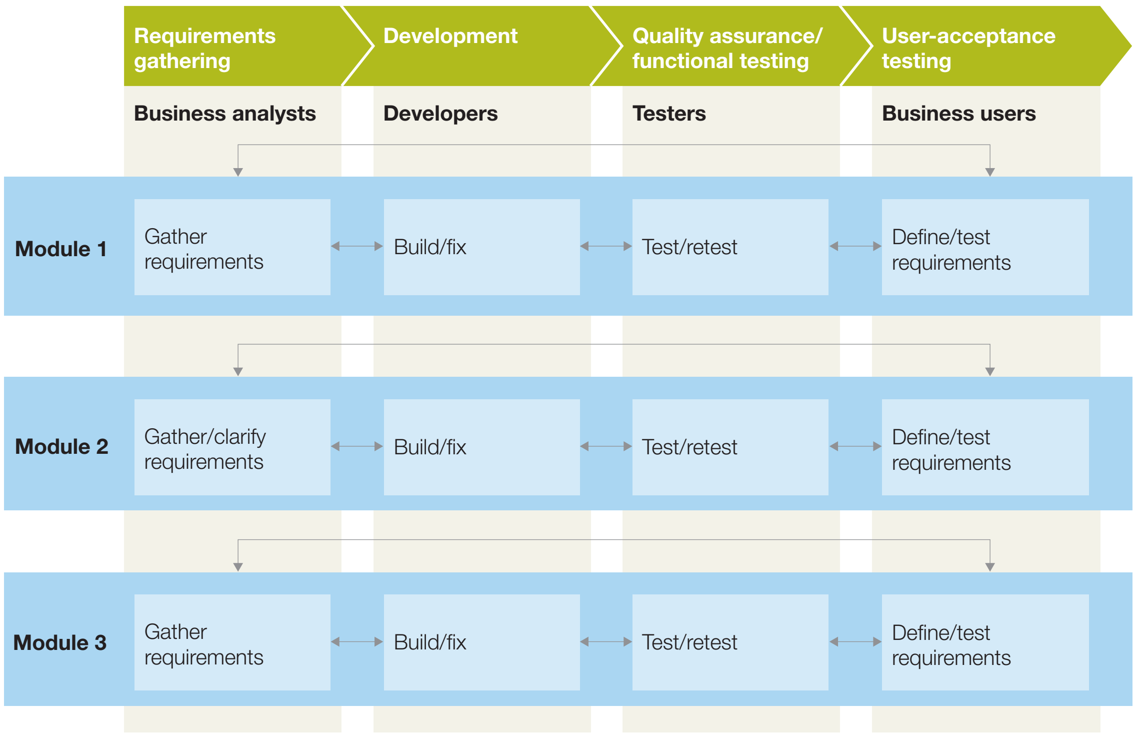
Shorter iterations. Cross-functional units enable shorter cycles for testing handoffs because coordination is simpler when each iteration remains within a small group. As a result, waiting time is greatly reduced when testers need developers to provide clarifications or fix defects.

## How an insurer benefited

A large insurer sought to develop and roll out a global claims platform. Employees

Exhibit 2

**Work cells are organized by modules, with end-to-end ownership across functions.**



assigned to the project were located in four cities across three time zones. The application-development work was organized by functional discipline (business analysis, development, and testing). While there was a common project plan, it was effectively a stringing together of three separate project plans, one for each functional discipline. As a result, teams communicated inefficiently, which led to many code defects and much rework, poor sequencing, and missed milestones because no one had responsibility for the whole project.

Midway through the project, the insurer switched to cross-functional teams, giving each one responsibility for a set of logically related use cases. As a result, team members began to focus on delivering end-to-end functionality rather than just thinking about their own roles. This approach enabled more rapid exchange of information, faster requirements clarifications, and speedier problem solving. Code defects fell by 45 percent in just one month, which reduced the need for time-consuming rework. The new way of working resulted in 20 percent quicker time to market and thus improved frontline productivity. In addition, business customers could see the end product ahead of schedule and suggest necessary changes that enhanced the customer experience.

• • •

Some large application-development projects are challenging because of their complexity and interdependency among work streams. Cross-functional teams with end-to-end ownership of application modules can improve the cost, quality, and speed of these projects by providing more accountability, better coordination, and shorter iterations.

The authors wish to thank Guy Assad, Krish Krishnakanthan, Ming Ruan, and Nihal Sarawgi for their contributions to this article.

**Sriram Chandrasekaran** is a consultant in McKinsey's New York office, **Sauri Gudlavalleti** is a consultant in the Delhi office, and **Sanjay Kaniyar** is an associate principal in the Boston office.