McKinsey&Company
Digital McKinsey

Article
September 2015

Beyond agile: Reorganizing IT for faster software delivery

By Oliver Bossert, Chris Ip, and Irina Starikova

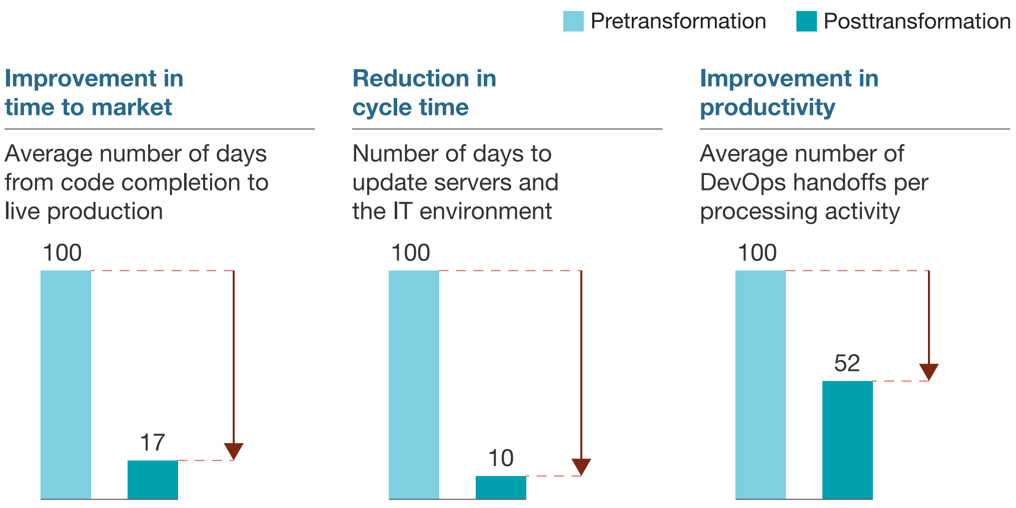
The integration of software development with IT operations can rev up companies' delivery of new applications. But this approach may not be right for every part of the IT portfolio.

A**fter more than** two decades of experimentation among Silicon Valley giants, "agile" has finally gone mainstream. Companies inside and outside the Valley are using some form of this software-development methodology, which emphasizes, among other things, rapid building and frequent delivery of software and system updates, with continual user involvement. Under this approach, companies are seeing increased productivity within their software-development teams, faster release of digital products and services, and improved customer experiences. Our experience suggests, for instance, that companies can reduce the average number of days required to complete code development and move it into live production from 89 days to 15 days, a mere 17 percent of the original time (Exhibit 1).

Exhibit 1

The value of adopting DevOps can be significant.

Indexed to 100



- Eliminate rework through integrated change management and automated deployment and testing

- Eliminate wait time and rework through standardized processes
- Eliminate non-value-added work through automation

- Eliminate wait time and rework through improved development and operations communication

McKinsey&Company

A lot of companies are now kicking the tires on DevOps, the next wave of innovation in software development and delivery and a critical enabler of agile software development. Under this product-development approach, companies seek to fully integrate their software-development functions with their IT operations so teams can jointly build, test, release, and maintain new digital applications more frequently and more efficiently.[1] Software is designed with discrete business requirements and system integration in mind, rather than in a vacuum, and developers and operations staffers are equally responsible for the delivery and stability of code.

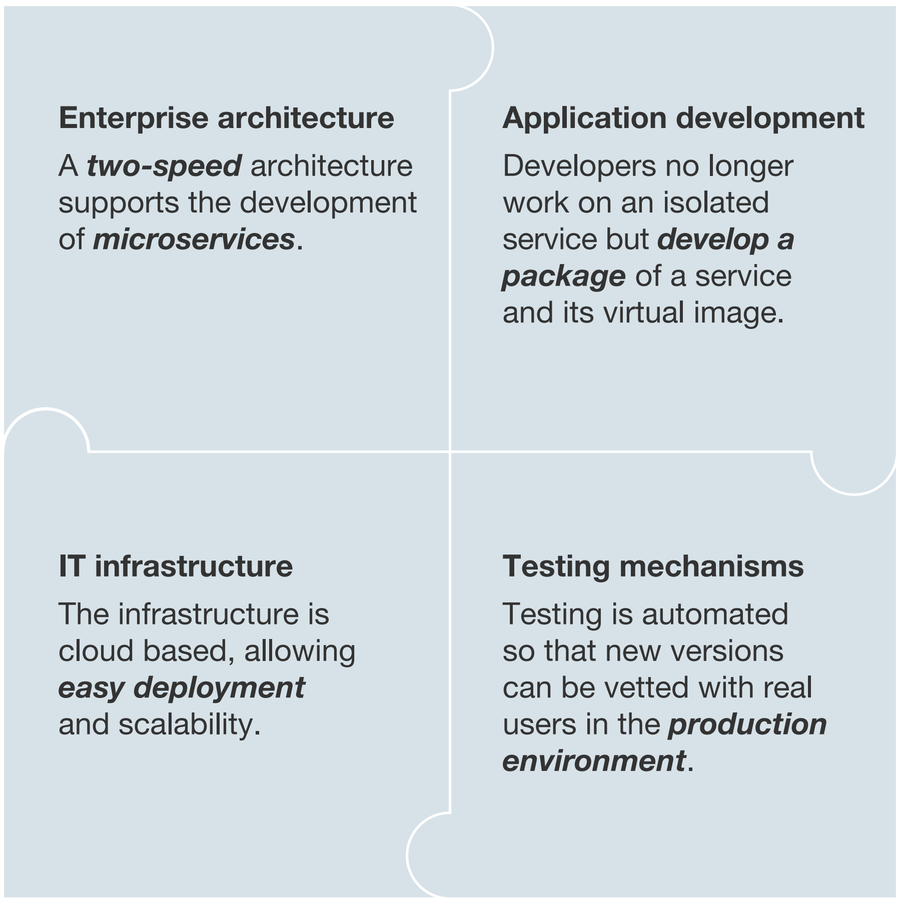
However, few companies, regardless of industry, have been able to reap the full value of DevOps. The implementation of agile has typically affected interactions only among small groups of business stakeholders and discrete application-development teams. By contrast, the move to a DevOps model requires that companies make broader, more systemic changes that could significantly alter interactions among *all* software-delivery teams, IT-operations staffers, and business stakeholders. This is a more complex undertaking.

For most established players, reorienting IT operations around a two-speed IT architecture—which features stable, transaction-oriented systems on the back end and fast-changing, customer-facing applications on the front end[2] —is a prerequisite for implementing both agile and DevOps approaches. But not every application the company develops or every update in a two-speed environment will require the joint collaboration that is central to a DevOps model. Some of the mechanisms used to support rapid development of e-commerce applications, for instance, may not be as successful in building or maintaining applications for a core transactional system developed in COBOL. In those instances, the traditional split of roles and responsibilities among IT operations, software development, and business stakeholders may actually be more acceptable.

In this article, we will discuss the considerations IT executives face when trying to adopt a DevOps model within a two-speed IT environment (Exhibit 2). They will need to determine how and where to introduce new technologies, such as automation and cloud platforms, depending on which parts of the company they think would benefit most from a DevOps approach. And they will need to explore new production processes and forms of governance so that IT operations and software-development functions across the company can work together effectively, despite the fact that they may be operating at different speeds.

Exhibit 2

To deploy DevOps in a two-speed IT environment, companies need to pay attention to the following factors.



Enterprise architecture
A ***two-speed*** architecture supports the development of ***microservices***.

Application development
Developers no longer work on an isolated service but ***develop a package*** of a service and its virtual image.

IT infrastructure
The infrastructure is cloud based, allowing ***easy deployment*** and scalability.

Testing mechanisms
Testing is automated so that new versions can be vetted with real users in the ***production environment***.

McKinsey&Company

Running at two speeds

Over the past decade or so, companies that were born online have revolutionized how technology infrastructure is built and maintained, and how software applications are developed and deployed. They have been among the first to integrate their software-development functions with their IT operations and focus on continuous delivery of small upgrades, where teams rapidly design, integrate, test, deliver, and monitor software changes.

Netflix, for instance, has created a cloud-based IT architecture that allows its developers to launch hundreds of software changes a day. Its website comprises hundreds of microservices hosted in the cloud, and each service is maintained by a dedicated DevOps team. Developers don't need to request resources from the IT operations team; instead they can automatically build pieces of code into deployable web images. As those images are updated with new features or services, they can be integrated with Netflix's existing infrastructure using a custom-built, web-based platform on which infrastructure clusters are created. Testing is carefully done in the production environment with a subset of users. Once the web images are live, a load-balancing technology routes part of the traffic to them from older versions. Automated monitoring ensures that if something goes wrong with the deployment of new images, traffic is routed back to older versions, and the new images are rolled back. Because of this level of automation, Netflix can deploy new code into its production environment within hours, where most companies would need months.

Of course, Internet companies such as Netflix have had the advantage of being able to start from scratch with their IT architectures—with no complex legacy systems to either reconfigure or maintain. And because their main products, web applications, are 100 percent customer facing, these companies have learned how to react quickly to customer feedback and release new features and improvements on the fly.

By contrast, most non-Internet companies seeking to similarly adopt a DevOps model are often saddled with older, transaction-based systems that they must somehow reconcile with agile approaches to software development. What's more, not every function within the brick-and-mortar organization will require DevOps; this would be the case, for instance, for systems of record that are not time sensitive, such as a general ledger. These companies therefore must not only contend with developing a two-speed IT architecture but also enabling a two-speed IT organization.

Managing a two-speed IT architecture

A two-speed IT architecture allows large-scale companies to accelerate the release of innovative products and applications that could make a substantial difference to customers while maintaining legacy IT systems that are less innovative but still necessary for the stability of the business. This sort of architecture emphasizes tight integration between the software applications being developed and the hardware infrastructure supporting them. Historically the IT operations teams maintaining software and hardware have been kept entirely separate. But with the increasing prevalence of vertical enterprise-resource-planning systems, the advent of network virtualization, and the emergence of software-as-a-service models, the two sides have moved closer to one another. These technology trends have had the effect of removing complexity from hardware stacks and making them more accessible to software developers.

A two-speed environment requires that companies introduce automation tools to support continuous delivery of software—especially in the testing and production phases. Automation can allow for better management of, among other things, the release of software updates, the porting of new code, and the general processing environment. Most important, automation tools and cloud-based technologies can serve as the bridge between legacy IT systems on the back end and consumer-facing applications on the front end, allowing companies to pursue seamless testing, provisioning, deployment, governance, and security of servers and new software releases (Exhibit 3).

Exhibit 3

It is possible to deploy new code on a site within an hour.

	Code	Build file	Jenkins[1] build	Develop	Test	Production
Duration	Varies	15 min	20 min	5 min	10 min	10 min
Description	Write the actual code for the service	Write the build file and define depen-dencies	Create the Jenkins[1] build and run the build job	Create new web images	Deploy the new web images in the test environment	Deploy the new web images in the production environment

[1]Jenkins is an open-source continuous-integration application that monitors execution of repeated jobs, such as building a software project.

McKinsey&Company

A two-speed IT architecture conveys a number of critical advantages, but it takes time, careful consideration, and commitment to establish. Netflix, for instance, developed most of its cloud and automation technologies in-house, but companies have any number of products and packages (some open source) to choose from that can allow them to achieve similar dual-speed performance.

The most critical factor in establishing a two-speed architecture is for IT leaders to adopt a capabilities-based view of the IT architecture, rather than a system- or process-oriented view. This means identifying and clearly defining those software applications that cut across multiple business units. From a capabilities perspective, for instance, IT leaders could see that certain applications developed for the company's customer-relationship-management (CRM) system may require a DevOps approach while others, such as core banking systems or transaction-processing applications, would not. The CRM system would not simply be considered a system of record, too slow to qualify for a DevOps program. Instead, IT leaders could allocate resources toward "fast" and "slow" applications as required—gaining the critical benefits of the DevOps approach where it is possible to do so.

Managing a two-speed IT organization

While addressing the technology architecture and infrastructure required to enable DevOps, companies should simultaneously consider making changes to various operations, processes, and governance structures in the IT organization and within the business overall.

The DevOps approach challenges the established product-development norms in most IT organizations. Historically, companies have separated their infrastructure (hardware) from their application-development (software) organizations and have kept the "build" staff away from the "run" staff. A DevOps approach requires that companies tear down these organizational silos, thereby marking a significant change in IT management strategy. Additionally, IT leaders adopting DevOps organizational models may need to reconsider how technology partners are integrated into their software-delivery processes—a trend that is forcing some system vendors to consider ways to make their platforms available as a service.

The biggest task for IT leaders is to identify those parts of the company where the use of DevOps would make most sense—likely focusing on those parts of the business where speed is at a premium, and where there is a significant opportunity for the company to differentiate its customer experience from the competition. (Think of a retailer using DevOps to improve its website checkout experiences, or a bank offering new fund-tracking capabilities at its site.) For those parts of the business where DevOps might make less sense—where reliability and resilience of software is more important than speed to market—IT leaders will need to determine how to maintain the split between software development and IT operations, and which roles and processes to adapt for a culture of continuous delivery.

Redefined roles

By its very nature, integrated product development requires strong collaboration between business and IT—and in some cases new or redefined roles. Business analysts must communicate the requirements for new software features and functionality in terms that employees in all departments can understand—and they must be flexible and willing to change the business requirements slightly when doing so could speed up implementation. Engineers and product developers must work across functions and among different product teams—under a DevOps model, informal collaboration and coordination among these business and IT coworkers actually becomes more important than formal reporting and approval processes. Software testers must collaborate with developers and business analysts—first with business analysts to clarify feature requests, and then with developers after the code has been developed, giving them immediate feedback on software performance. With DevOps, end users are no longer passive recipients of "big bang" software or service releases—companies seek their input early and often as they develop and test new software features.

Cross-functional teams of application-development, infrastructure-management, and operations professionals should be convened to streamline the ownership of stacks across the application-delivery pipeline. In the case of continuous delivery, for instance, a joint team would oversee all processes (and associated tools) relating to this development activity, such as application building, testing, and deployment; performance management and monitoring; and virtualization and configuration management. Previously, some of these components would be owned by different organizations. Also, infrastructure teams should be given a seat at the table, with decision rights equal to those of software-development teams.

Redefined culture and talent

Integrated development and continuous delivery can only happen within a corporate culture that empowers its software developers and refines its IT and R&D reporting structures. In most organizations, product development and IT operations live in separate towers, with people of different mind-sets, skills, and experiences. IT and business executives will need to break down these barriers. For instance, rather than have all developers report to the head of "build" and all operations employees report to the head of "run," some must be purposely assigned different reporting lines. Further, employees will need training opportunities, and their salary schemes may need to be reconsidered. Traditionally, product developers have focused mainly on programming frameworks; in a DevOps environment they will be held responsible for the quality of their code. They will need to know operating-system basics and must show strong collaboration skills as they work jointly with operations engineers to determine how best to solve application-development or deployment problems. As a result, many companies are already modifying their recruiting practices to hire "full stack" engineers—professionals who understand all aspects of computing, including user interfaces, databases, and networks.

Redefined processes and governance

Companies may want to look across the entire spectrum of software-delivery processes to determine which will need to be redefined or fully automated so that development teams can take advantage of infrastructure as a service, as needed, and so that code can be ported into testing and production environments in a standardized way. There are a number of lessons companies can take from Internet pioneers on the types of process and governance changes to deploy in support of DevOps. For instance, Internet companies enforce "self service" for developers; teams can test, promote, and deploy code in production environments without requiring constant hands-on involvement from infrastructure-operations teams, although both teams share responsibility for code performance. Internet firms also impose rigorous, automated testing of new code at all stages of the application-development process; in some dot-com companies, sophisticated tests are completed automatically every 10 to 15 minutes. Additionally they take advantage of advanced analytics and other tools to preemptively scan code for exceptions and send developers automated reports about the code segments that are most likely to create errors.

The value of implementing DevOps can be significant with respect to both productivity and time to market. But the implementation of DevOps is not simply about the deployment of new IT methodologies. It must be treated as a company-wide transformation—one that incorporates process and governance considerations as well as technology-related ones.

1. Satty Bhens, Ling Lau, and Shahar Markovitch, "Finding the speed to innovate," April 2015.
2. Oliver Bossert, Chris Ip, and Jürgen Laartz, "A two-speed IT architecture for the digital enterprise," December 2014.

About the author(s)

Oliver Bossert is a senior expert in McKinsey's Frankfurt office, **Chris Ip** is a director in McKinsey's Hong Kong office, and **Irina Starikova** is an associate principal in the Silicon Valley office.