APRIL 2015

BUSINESS TECHNOLOGY OFFICE

Finding the speed to innovate

Companies can test and launch digital products and services faster, and at lower cost, by integrating their product development and IT operations, also known as DevOps.

Satty Bhens, Ling Lau, and Shahar Markovitch

Few executives would argue that speed and innovation are critical success factors in today's digital world. Being able to innovate quickly and cheaply, test digital products and services in the market, refine them, and release them on a regular basis has become a competitive advantage. But how can companies do it consistently and succeed with their digital products and services at scale?

While many companies have been trying new methods to get there, two complementary lean capabilities are showing great promise: the integration of product development with IT operations (commonly referred to as DevOps), and a focus on continuous delivery. Both emphasize automation, continuous monitoring, and sharing of information and processes across product-development and IT operations groups to enable the frequent release of small internal software updates (see sidebar, "What are continuous delivery and DevOps?").

Both capabilities can greatly increase companies' speed to market (cutting product-testing times from weeks to minutes) while significantly reducing the cost of delivering new products and services. They can help to ensure higher-quality software, better rates of customer adoption, and fewer risks in product and service delivery. Indeed, these capabilities can become a true engine for innovation.

Leading technology companies have been early adopters of these capabilities and have reaped the benefits. Facebook, for instance, releases millions of lines of code and implements hundreds of small changes to its website daily without downtime. And Amazon can release code every ten seconds or so, update 10,000 servers at a time, and roll back website changes with a single system command.

A number of nontech companies have also successfully explored and built up capabilities in continuous

Takeaways

Innovating quickly and cheaply, testing digital products and services, refining them, and releasing them regularly has become a competitive advantage in today's digital world.

How can companies do this and succeed with their digital products and services at scale? Two promising approaches may help: integrating software-development functions with IT operations (DevOps) and focusing on continuous delivery of small software upgrades.

Capabilities can be developed in three stages: simplify, scale, and sustain. Companies should create high-quality units of code that are tested early and often, pursue automation opportunities in the highest-value areas of the company, and move technologies and products to cloud-based virtualized environments.

delivery and DevOps. One international travel company facing disruption from online competitors was able to introduce new features to its website more quickly and efficiently by moving software and systems to the cloud, fully automating testing and provisioning of servers, and rolling out a one-click software-deployment process. In another case, a large financial institution is using a continuous-delivery approach to streamline its product testing and go to market quickly with new digital services; processes that once took days to complete now take minutes.

But while a range of companies are experimenting with DevOps and continuous delivery, few are capturing their full value. Legacy IT systems, antiquated technologies, complex system and project-management processes, and uncoordinated actions by disconnected teams are undermining even the most determined business leaders.

In our experience, the companies that are implementing these software-development approaches most successfully have avoided investing large sums of money to move their entire technology infrastructure to the cloud in one "big bang." Instead they have adopted a more deliberate approach, building their capabilities step by step and funding further changes and process improvements through the resulting savings and efficiencies. In this article, we consider the three main stages for building capabilities in continuous delivery and DevOps—simplify, scale, and sustain—as well as the cultural changes required to reap the most value from these lean approaches.

Implementing DevOps and continuous delivery

For most companies, software development and rollout involve a period of iteration, a period in which all changes are "frozen" (that is, no more development is allowed), and a period during which all software

What are continuous delivery and DevOps?

Continuous delivery accelerates the time to market of new software features through an investment in automation of testing, deployment, and infrastructure. This differs from the traditional approach, which often encourages the organization to defer new features and fixes into a single "big bang" release. Continuous delivery dramatically reduces the incremental cost for each software release; smaller releases happen early and often to provide more timely feedback from customers to the product team.

DevOps is a cross-functional approach that integrates development and IT operations into a product-oriented culture. The DevOps culture embraces automation, monitoring, and sharing to enable continuous delivery. Historically, development and operations teams have worked in silos and often with opposing priorities. DevOps promotes joint ownership of a technology product by integrating these groups into a single functioning team to improve response times. DevOps is largely a by-product of the cloud revolution, which created a view of "infrastructure as code" that has blurred the lines between coders and infrastructure teams.

changes across the organization are integrated into one package and reviewed for weeks by a large team of testers. In parallel, a separate team sets up the supporting technology infrastructure, a process that can take a few weeks. The release is deployed only after all these steps are completed. Even with this lengthy process, buggy software often results.

Given this difficult undertaking, most companies will release major internal software changes only a few times a year. But, as the company examples cited above suggest, DevOps and continuous-delivery models can help streamline the development of digital products and services and improve time to market. We have identified three main success factors for adopting these lean approaches.

Stage 1: Simplify

Companies need to create a "single source of truth" for all software: one repository for storing, versioning, and tracking all source code. The mainline version of code can then be accessed quickly and reliably. For this approach to be most effective, developers must submit code changes frequently to the repository, which reduces the size of the code to be reviewed by peers as well as the complexity of merging parallel code changes.

Simultaneously, the organization should be committed to developing high-quality code—units that are modular, simple, and easy to maintain. Adopting "test-driven development"—the process of writing automated tests for code before actually writing the code itself—is an important step. Specifically, teams should focus initially on testing simple and discrete units of code, eventually expanding to more complex, integrated functions. To ensure that developers adhere to quality guidelines, the best companies perform complexity analyses, which measure the integration and logic factors associated with units of code and have a well-honed peer-review process to identify poorly written code so it can be corrected quickly. Google's engineers, for instance, send their code to a large distribution list across the company, and randomly selected peers review the code and post comments to the entire list.

Setting up this first stage does not require significant resources. We have seen companies make progress simply by reassigning one or two talented developers, implementing a handful of automation programs, and relying on vendors for monitoring and support. In many cases, companies will already have the technologies (such as source control, test control, and test automation) in-house and can supplement them with good, open-source software. One international hotel company consolidated its sales and catering systems by moving to a single version-control repository, integrating software code twice a day, and insisting that developers write automated tests for new units of change in their code. As a result, the company was able to reduce its time to market with new software by about 25 percent.

Stage 2: Scale

It can be a long and expensive task to scale up and build out fully automated IT systems that have a mix of modern and legacy technologies. Focusing on the highest-value automation opportunities is the most productive way forward. Customer-facing components—for instance, mobile applications and e-commerce websites—are usually the highest priority for most companies; they generally also experience the highest rate of change and innovation. To support these opportunities, developers should incorporate into their IT architectures sophisticated automated testing, such as verifying an end-to-end customer journey. They should also use performance tests that aim to measure the system under load or stress, and security tests that seek to measure its resilience against malicious attacks. Every "merge" of new code into the mainline source code should trigger these tests and the deployment of the latest code to low-risk test environments.

An important by-product of this approach is transparency into the status of the mainline code and the overall quality of the code base. The automated reports can offer managers insights that can help them continually improve planning, resource utilization,

and even product and service quality (for example, by measuring conversion rates for use of specific products and services) and issue resolution (for instance, by identifying root causes of technology failures).

One large corporate bank moved to continuous delivery as part of a larger initiative to digitize its customer processes—aiming, for instance, to simplify corporate lending and customer onboarding. The company sought to automate the deployment of new software to test environments. Full-time testers worked shoulder to shoulder with developers to manually test the more complex transactions as a complement to the simpler transactions. Once a feature was completed, it was transferred to the test environment in just minutes. Any bugs identified were returned to the developers to be fixed that same day. Through its use of a continuous-delivery model, the bank was able to develop fully tested software releases within one week compared with the six weeks it had taken previously. The overall quality and predictability of those releases were vastly improved as well.

Stage 3: Sustain

While companies can often gear up to change their software-development processes in one big burst, this all-hands-on-deck approach is rarely sustainable—hence the appeal of continuous delivery (exhibit). However, the pursuit of continuous delivery needs to be easy for staffers to follow and ingrained in the culture to maintain its value. This means systematically moving technologies and products to cloud-based virtualized environments, with software automation in place to fully control the server and technology infrastructure, scale, and quality. One international consumer-goods company migrated its software to the cloud so it could give developers self-service access to production-like environments for testing, delegate the

Exhibit

Continuous software delivery can increase companies' speed to market with high-quality digital products and services.

	Traditional software delivery	Continuous software delivery
Time to delivery	**Internal software release once every 3–6 months**	**Internal software release multiple times a week (or daily)**
Quality and testing	**Manual testing of up to 50% of software releases** performed by large teams	**Automated testing with more than 80% coverage** requires limited human intervention to validate
Software deployment	**Manual deployment of software** can take 30–50 individual steps; limited success	**Fully automated deployment of software** one-click process can launch more than 50 steps at a time
Monitoring and support	**Reactive software-monitoring** issues, downtime reported to users in hours, days	**Proactive software health–monitoring** issues, downtime reported to users in seconds, preventive actions taken at defined thresholds
Infrastructure setup	**Infrastructure setup can take 3–4 weeks or longer**, manual and highly error-prone process	**Automated provisioning of new infrastructure** setup and configuration in less than 10 minutes end to end

Source: McKinsey analysis

execution of thousands of automated tests to "virtual servers" that take seconds to run, and adjust servers automatically to support spikes in traffic based on seasonality and multimarket demands.

Migration of software to the cloud gives companies an opportunity to simplify their IT architectures and introduce new technologies (middleware, for instance) and capabilities (A/B testing, for example) without affecting legacy systems.[1]

Companies can take advantage of cloud-based environments and use automation extensively—some companies even choose not to release features if at least 80 percent of the code is not covered by automated testing. Failures can be rolled back in seconds, allowing fixes to be made without putting significant parts of the business at risk. Developer work flows are also automated to allow one-click software releases in seconds, not hours. Organizations can continuously release software to production with limited reliance on human intervention, and changes can be verified and released with a high degree of confidence.

As a company's continuous-delivery capabilities mature, it can shift its focus from automating development work flows to improving system scalability, resilience, and disaster recovery, as well as optimizing the monitoring and logging of systems to better track customer and system performance. Product teams can learn and make decisions quickly based on data that affect the customer experience and performance.

It's about more than technology

To move toward DevOps and continuous delivery, companies cannot focus solely on revamping tools and technologies. They must treat the initiative as a change program requiring an updated culture. The CIO and other business leaders must commit resources and time that go far beyond forming committees and attending more meetings. Shifting to this way of working requires a new organizational model, though which particular one depends on a company's specific situation. Some organizations have set up a model similar to a center of excellence, in which a separate DevOps team is created and used to embed capabilities inside teams, build and expand capabilities, set standards, and capture and disseminate best practices. Other organizations embed DevOps champions within application-development teams. Whatever the model, in our experience, a successful transformation requires the following ingredients:

Be clear about the change, and set high aspirations.

Senior leaders should create a compelling vision of where the organization needs to go, provide resources for a change program, lay out a road map, and put in place clear measures of success. A change program should articulate a new operating model for how teams work together, including who has decision rights and what process checks are needed. A road map should include the assignment of people with specific responsibilities, a plan for building capabilities, and the identification and sequencing of architecture and tool changes. Goals should be bold but specific—for instance, reducing time to market from months to days, or rolling out releases in seconds, not hours. One logistics company set a high bar—to transform its entire technology delivery process within a year—and communicated it widely.

Create incentives that are aligned with business outcomes.

The incentives associated with the successful use of DevOps and continuous-delivery approaches need to be aligned with designated business outcomes—for instance, revenue targets, conversion rates, customer-engagement scores, product-quality metrics, and time-to-market figures—and should be shared by the entire team. Too often we see that the development group and business have different incentives—the latter is focused on revenues while the former is worried about delivery times. Similarly, there is often a disconnection between the development group and the operations team, which will necessarily be fixated on up-time and ticket management. Incentives should motivate and reward complete teams rather than individual functions.

Create a 'single team' mind-set.

To deliver on the promise of DevOps and continuous delivery, coders and infrastructure teams must work in lockstep. That requires infrastructure, operations, and developer team members to work side by side using the same agile software process of iterative planning, development, and testing. During a 12-month transformation, the CIO of a technology service provider moved 500 full-time employees in operations physically closer to the engineering team to enable agile software development. The company also trained IT employees in new delivery methods. This allowed the organization not only to become more efficient but also to significantly reduce the time it took to roll out service changes to end users.

Build a continuous-improvement and data-driven culture.

It's not enough to release great software code once; high-performing companies do it continually. That requires teams to collect and interpret data and customer feedback, adapt code, and redeploy in near-continuous cycles. To do this quickly, teams need transparency and easy-to-understand metrics so they can understand exactly how the program is performing. One bank created such transparency by integrating critical site traffic, customer-performance metrics (such as number of visitors and digital-sales figures), and site-operations metrics into an online dashboard that was projected on 60-inch LCD screens across the room where the digital team was located. The visual metrics made it easier for teams to spot early indicators of lagging performance; errors were reduced by 25 percent over a few months.

Build the right capabilities.

As infrastructure and operations teams shift from executing manual steps to writing software code to automating processes, new skills and experience will be required. Some of the more important ones include the ability to write software code and test automation; experience with new tools, programming languages, and development-work-flow automation; experience with agile development; and collaborations with software-development groups and business. Organizations that were "born digital" may be able to look in-house for people with these skills, but most companies are being forced to look outside for help.

Speed and innovation are the hallmarks of a successful company in the digital age. To get there, companies should consider adopting an agile-software-development culture that goes beyond delivering one-time results to continuously adapting and growing over time.

[1] For more, see Oliver Bossert, Chris Ip, and Jürgen Laartz, "A two-speed IT architecture for the digital enterprise," December 2014, mckinsey.com.

The authors would like to thank Steve Jansen for his contributions to this article.

Satty Bhens is a digital partner at McKinsey Digital Labs and is based in McKinsey's New York office, **Ling Lau** is a digital manager at McKinsey Digital Labs and is based in the San Francisco office, and **Shahar Markovitch** is a principal in the Tel Aviv office.