The Battle for Speed Is Won in the DevOps Arena

MARCH 07, 2022
By Andrew Agerbak, Jon Brock, Kaj Burchardi, Steven Alexander Kok, and Arne Weiner
DevOps is critical for software-intensive businesses. Here are five principles to ensure that you can attain the necessary velocity.

For software-intensive businesses, the fast, effective delivery of new offerings is not only a driver of competitive advantage but also a critical capability for long-term survival. This reality has prompted many companies to make substantial investments in DevOps, which includes practices, processes, and tools to improve the quality and speed of the software development and integration life cycle. Unfortunately, so far these investments have not often translated into tangible results that satisfy internal stakeholders and customers.

In our work with clients, we have seen firsthand the challenges companies face in getting the most out of their DevOps investments—and how these efforts to improve the software-delivery operating model can succeed. The primary problem is that most companies have too many silos and handoffs in the development life cycle (between customers and users, business, development, infrastructure and operations, and security) to attain the necessary velocity in their software development and integration. To help companies invest effectively in DevOps, overcome these challenges, and improve speed to achieve
outcomes, we have synthesized our experience into five key principles for success.

# Make Speed the Primary Metric

Broadly speaking, most companies invest in DevOps for three main reasons: speed (sometimes framed as time to market), cost, and quality. But in our experience, speed is the most critical factor because, if done right, the other two will follow. Increased speed brings with it enhanced agility and efficiency (which together reduce costs) and the ability to iterate and improve quickly (which also boosts quality, as long as you have the right testing and feedback processes).

Speed, or time to market, is sometimes too narrowly defined as time to deployment (meaning the time it takes to deploy a new feature in the market). A better definition incorporates the time needed to achieve outcomes. There are three core tenets of speed:

- **Creating outcome-oriented persistent teams that own these outcomes over time and are responsible for the associated assets end to end.** When a team is involved long term and held accountable, it’s more likely to focus on longer-range goals such as improving product delivery speed.

- **Developing principles and guardrails designed to increase productivity and ensure that problems don’t get baked into solutions early on.** This requires introducing security, quality, and reliability engineering practices earlier in the development life cycle. Other principles and guardrails include putting architecture governance in place to keep code up to date (for example, by certifying the life cycle of a component so that no component goes too long without refactoring), choosing which tools to standardize
To leverage these tenets and improve cycle time, companies must also provide tools and services that minimize waste during development (such as simplifying toolchains and adding self-serve DevOps capabilities). In addition, it’s critical to enable and incentivize developers with a variety of training and clear DevOps development practices (including test automation, proper branching strategy, and feature toggles). Moreover, companies should streamline the operating model to maximize the quality and speed of feature builds. All of this requires effective, outcome-oriented governance.

Focus on Continuous Learning and Improvement

DevOps should not be siloed in the tech function. It’s an end-to-end effort that needs to include both tech and nontech participation. The goal is to improve business outcomes by integrating continuous learning and improvement in daily operations across all functions and stakeholders.
involved in the software development life cycle (SDLC). Continuous improvement is critical for any company to compete at speed and at scale—just as a Formula One pit crew must constantly improve the car if the team is to have any hope of winning the race.

“DevOps should not be siloed in the technology function. It’s an end-to-end effort that needs to include both tech and nontech participation.

To apply continuous learning to agility, efficiency, and quality, the company should elevate the importance of certain objectives and key results (OKRs):

- Speed to improve business outcomes
- Costs spent on unnecessary or misaligned technology development
- Delivery quality, such as fewer bugs, lower security risk, and greater customer satisfaction

A good way of gauging whether a company’s commitment to continuous learning and improvement is adequate is to study the budget. In our experience, 10% to 20% of the overall technology budget should be allocated to continuous improvement of delivery productivity. These DevOps investments need to focus on three main areas:

- Increasing the levels of automation across the SDLC (build, test, deploy, release, live application monitoring and management)
- Boosting the reusability of components and standardization of interfaces across the architecture
• Refactoring the architecture to make it more modular where necessary to improve speed to outcome

Achieving this type of focused investment, in our experience, is very challenging if a company is still working with a project-centric delivery model. That’s because the projects are typically funded for scope, so there is limited incentive to improve the delivery model. But if companies do not commit enough of their change budget to these goals, they will not achieve the envisioned speed to outcome necessary to compete in the digital realm.

Consider DevOps a Transformational Undertaking

DevOps done right is far more than a technology implementation involving processes and toolchains: it’s a transformational endeavor, involving the whole technology operation (and often the business), to improve the entire delivery model. The implications are felt far and wide.

Organization. The company must decide how best to support and increase DevOps capabilities. That means determining what to federate, by making sure that individual development teams have the right resources; and what to centralize, by ensuring that the central function has the capability to support individual development teams as they improve their methods of self-serving and self-managing their assets. The key concept here is to establish DevOps as a platform, which means that central teams are not a shared service. Instead, they focus on creating capabilities and services that individual development teams can use to boost their productivity (and ultimately cycle time). As a result, the boundary between application development and operations should
eventually break down and be owned within the teams: “you build it, you run it; you break it, you fix it.”

To a large extent, using DevOps to improve delivery productivity requires that companies manage the context of application development, management, deployment, and monitoring (instead of individual projects). By “context,” we mean all aspects that affect development, including management and leadership practices, the operating model, architectural guardrails, and tooling. Managing the context also allows for a more effective transfer of ownership and accountability so that it’s as close as possible to application development and operating teams, which in turn accelerates decision making—and fulfills the principles and promises of agile.

Process. Companies need to simplify their control framework and governance to reduce handoffs and inefficient touch points. Instead of checking and testing code themselves, employees in the DevOps environment should manage highly automated machines (which rely on many critical third-party assets) that provide these controls. For example, the site reliability engineering (SRE) model has clear rules, governance, technical, and operational practices to oversee the overall quality of the technology stack. (See the sidebar, “A Global Bank Establishes SRE.”) The challenge is to consistently manage the required controls at scale in a highly fragmented application stack, including overall quality (such as SRE models), security (the “Sec” of DevSecOps), as well as continuous monitoring of the infrastructure and live applications (AI for IT operations [AIOps], for example).

---

A GLOBAL BANK ESTABLISHES SRE

BCG worked with a global bank to stand up a site reliability engineering (SRE) function, establishing several small teams that
could apply software engineering to infrastructure and operational problems. SRE teams use engineers with software expertise to perform work historically done by production management groups.

The bank’s SRE teams, each with different types of infrastructure and development experience, interact with their environment—production, development groups, testing teams, users—while applying seven key principles: simplify and modularize, measure everything, degrade services gracefully, embrace risk, set service-level objectives, automate, and respond to failures systematically. These principles help the SRE teams and other stakeholders streamline their production management efforts.

To help embed SRE culture into the bank’s teams, BCG designed a curriculum that addressed two broad areas: the hard skills necessary to define how engineering teams design and interact with platforms, and the soft cultural skills required to embrace risk and break organizational silos. Critically, the bank committed to investing in these teams and this curriculum to improve reliability over the long term.

Applications and Infrastructure. Companies need to push development into a technology stack that is modular, can be automatically deployed, and has high levels of automated quality controls (such as those offered by today’s modern cloud providers). This requires strong coordination and engagement between the development and operations teams to design and manage the application and infrastructure stack. The development teams take responsibility for choosing and designing the stack, whereas operations teams focus more on providing the stack and its elements as a self-service capability to development teams.
Services and Capability Sourcing. It’s important to manage the evolution of third-party services. Take, for example, when a company outsources the management of on-premises infrastructure or application support and maintenance. In these cases, the company and outsourcing partner need to develop a plan to gradually automate these services—and put the contractual terms and incentives in place to make this happen instead of only squeezing them for cost.

Don’t Leave Data Behind

DevOps is not just about code; it’s also about data. Orchestrating the workflow to manage complex AI and machine-learning data pipelines requires the same rigor as managing code. Moreover, good test data is crucial to ensure the effectiveness of quality control across different environments—including production, development teams, testing teams, and users. But in our experience, data often doesn’t get the proper attention in DevOps efforts focused on configuration management, testing automation, and continuous software development.

“DevOps is not just about code; it’s also about data.

It’s important to align improvements in data management with DevOps and architecture efforts to ensure end-to-end improvements in speed. (For example, even when choosing commercial off-the-shelf software packages that use their own data stores to operate, companies need to control their own data and not allow the vendor to become the master repository for customer data.) Without this alignment, it’s difficult to manage test data across different environments consistently. And if data
teams are forced to resolve these issues separately, the delivery model will fragment.

DevOps guardrails and practices also have an impact on data. For example, developers are sometimes uncertain about what data needs to be logged for performance-monitoring purposes. Their response is often to become overly cautious about what data to log—in some cases, overloading the services and network, creating latency and potentially instability issues.

This is why it’s essential to consider early on how the end-to-end stack (architecture choices, data management, and tools to manage the stack) will work throughout the development life cycle and into live production.

**Invest in People and Partners**

There is a general scarcity of strong engineering talent (especially with modern DevSecOps and cloud capabilities). Even when companies can find a good outside candidate, they rarely find someone who deeply understands their business context, history, and how to drive holistic changes in the company’s operating model. So, in addition to battling for the right external talent, companies should invest in the people already working for them, focusing on training and empowerment.

In our experience, having good DevOps practices attracts talent, while the lack of an effective SDLC can drive attrition. We worked with one client where developers saw their code go live in as little as a week and another client where it took a year. Not surprisingly, the former was able to hire and retain talent much more easily.

Companies also need to create incentives so that vendor partners will invest in continuous improvement—especially among the vendors supporting run services and managing legacy on-premises infrastructure.
Unfortunately, these contracts are often managed primarily for cost. Companies keep pressuring their vendors’ margins, reducing their partners’ incentives to invest or co-invest in model improvements.

The service model itself can also discourage vendors from making these improvements. For example, a vendor providing testing services has no inherent incentive to automate these services. Similarly, if a partner provides application maintenance and support, it’s likely to make more money when the stack functions poorly. And if a vendor contracted to run the infrastructure is paid on the basis of usage, it won’t be inclined to facilitate a transition to cloud services or application rationalization since these will reduce usage.

Given these financial realities, companies need to think creatively about how to design contractual incentives to drive co-investment in continuous improvement of the stack. The goal is to progressively create a team that can manage the increasing (and sometimes poorly understood) complexity of modular architectures and the plethora of services and capabilities that come with moving to multicloud infrastructure landscapes.

The delivery chain is only as fast as its slowest link. That’s why corporate transformations can’t treat new ways of working in business and IT as separate endeavors. Software-intensive businesses that implement agile without a mature DevOps-centric operating model may discover that two-week sprints in business can be hamstrung by nine-month software release cycles. DevOps is a way to bring business and technology together to truly boost speed to outcome, with the always-cherished consequence of reducing costs and increasing quality.
Authors

Andrew Agerbak
Partner and Director
London

Jon Brock
Partner and Director, Technology & Digital Transformation
London

Kaj Burchardi
Platinion Managing Director
Amsterdam

Steven Alexander Kok
Partner & Associate Director, Technology & Digital Transformation
London

Arne Weiner
Platinion Agile Director
Platinion – Cologne

ABOUT BOSTON CONSULTING GROUP

Boston Consulting Group partners with leaders in business and society to tackle their most important challenges and capture their greatest opportunities. BCG was the pioneer in business strategy when it was founded in 1963. Today, we work closely with clients to embrace a transformational approach aimed at benefiting all stakeholders—empowering organizations to grow, build sustainable competitive advantage, and drive positive societal impact.

Our diverse, global teams bring deep industry and functional expertise and a range of perspectives that question the status quo and spark change. BCG delivers solutions through leading-edge management consulting, technology and design, and corporate and digital ventures. We work in a uniquely collaborative model across the firm and throughout all levels of the client organization, fueled by the goal of helping our clients thrive and enabling them to make the world a better place.

© Boston Consulting Group 2024. All rights reserved.
For information or permission to reprint, please contact BCG at permissions@bcg.com. To find the latest BCG content and register to receive e-alerts on this topic or others, please visit bgc.com. Follow Boston Consulting Group on Facebook and X (formerly Twitter).