BCG

THE BOSTON CONSULTING GROUP

Developing Software at the Speed of the Cloud

The Boston Consulting Group (BCG) is a global management consulting firm and the world's leading advisor on business strategy. We partner with clients from the private, public, and not-for-profit sectors in all regions to identify their highest-value opportunities, address their most critical challenges, and transform their enterprises. Our customized approach combines deep insight into the dynamics of companies and markets with close collaboration at all levels of the client organization. This ensures that our clients achieve sustainable competitive advantage, build more capable organizations, and secure lasting results. Founded in 1963, BCG is a private company with 82 offices in 46 countries. For more information, please visit bcg.com.

BCG

THE BOSTON CONSULTING GROUP

Developing Software at the Speed of the Cloud

David Mark, Mike Quinn, Saurabh Shah, and Sanjay Verma

May 2015

AT A GLANCE

Cloud developers are revising the rules for writing software and breaking new barriers in releasing and updating code. Traditional software teams can learn from cloud development teams about unleashing creativity and satisfying customers.

Smashing Functional Silos

Cloud teams have end-to-end responsibility. Development, testing, operations, data instrumentation, and operational analytics all report to a common leader.

Living Software

Cloud teams continually update their code. To accommodate this dynamic process, software needs to be modular and loosely coupled.

Embracing Automation

Cloud teams invest heavily in automation to improve efficiency and quality.

Connecting with Customers

Cloud teams also invest heavily in metrics that provide real-time customer insights that allow teams to build software that meets customers' exact needs.

Clouds may move across the horizon in slow motion, but they are the jet rockets of software development. In today's mobile, ubiquitous, and instantaneous world, cloud teams are running far ahead of traditional teams in writing and releasing code. The cloud has also enabled these teams to become innovative and efficient, and to deepen their ties with customers.

Traditional software teams have a lot to learn from cloud teams about unleashing the creativity of their code writers. In fact, so do any companies that build software-enabled products and services. In today's landscape, in which software is embedded into everyday objects, that means virtually all companies.

After working with world-class cloud teams, we uncovered four principles that guide how they operate. These principles can help more traditional teams modernize their software-development practices.

- *Smashing Functional Silos.* Teams have end-to-end responsibility. Development, testing, operations, data instrumentation, and operational analytics all report to a common leader, creating a single point of accountability.

- *Living Software.* Many teams still treat each software release as a singular event and then move on to the next big thing. Cloud teams instead launch services with the explicit understanding that they will be continually updating their code. To accommodate this shift, software must be modular and loosely coupled.

- *Embracing Automation.* Traditional software development has too much downtime and manual checking and testing. Cloud teams invest heavily in automation to improve their efficiency and consequently the quality of their software.

- *Connecting with Customers.* For cloud teams, the customer is not an abstract concept but a real-time and constantly changing composite of actual behavior, usage, and preferences. These teams have invested heavily in metrics that provide real-time insights unavailable through more traditional customer-feedback methods, such as focus groups. These insights show teams how customers are using their products and services, allowing the teams to build software that meets customers' exact needs and even delights them.

For cloud teams, the customer is not an abstract concept but a real-time and constantly changing composite of actual behavior, usage, and preferences.

Collectively, these principles push decision making down into the organization, enabling software developers to own their code, understand their customers, and continually improve their service. They also eliminate the traditional trade-offs that

have long marked software development. Cloud teams are innovative *and* organized; they are agile *without* sacrificing quality.

Venture capitalist Marc Andreessen likes to say that software is "eating the world." Companies that fail to adopt these principles risk being eaten by those that do.

Smashing Functional Silos

Software has traditionally been developed sequentially, with the waterfall serving as a rough metaphor for its progression. Separate groups conceive, design, build, test, put into operation, and maintain software, with each group waiting for the previous group to complete its work.

This setup is fraught with high transaction costs. Participants can spend more time sitting in meetings and managing handoffs across organizational boundaries than writing and testing code. Disputes among these groups are often discovered late in the game and have to be resolved by senior executives.

Cloud development organizations are flat. Many of their functions report to the same manager. In addition, engineers in cloud teams often develop, test, deploy, and maintain their own software or service.

In this setup, individual contributors have a better sense of how their decisions affect the overall development and release of software, so there are fewer slowdowns and do-overs. As one cloud manager put it, companies "need to have a single throat to choke for each service."

Cloud development organizations are flat. Many of their functions report to the same manager.

Leadership Structure

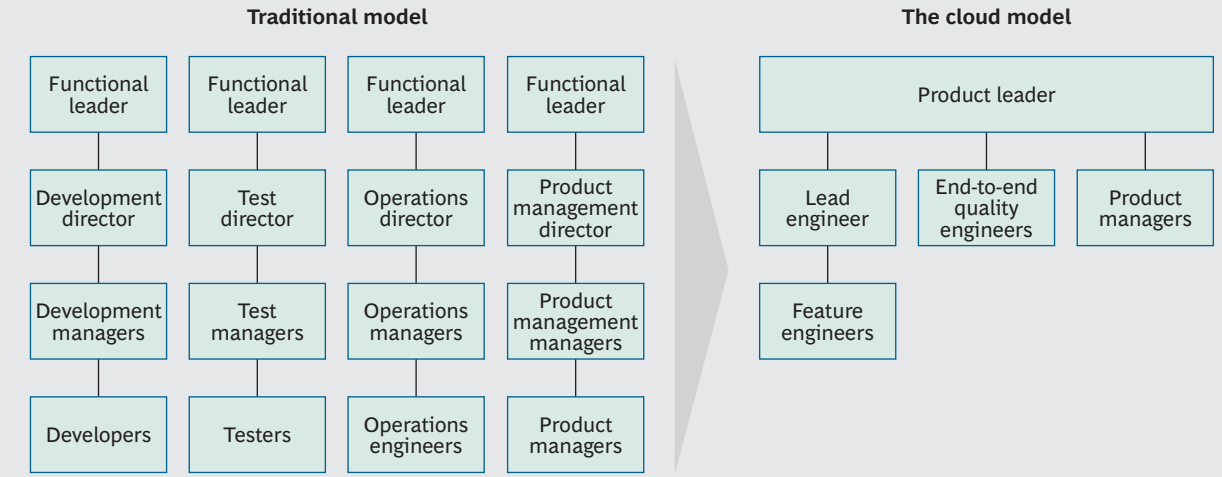
In many traditional software organizations, executives lead specific functions or disciplines, such as development or testing. An unintended consequence of this type of system is software complexity. To paraphrase Conway's law, software mirrors the organizational context in which it is created. Software takes on the mishmash that results from trade-offs, handoffs, poor communication, and competing organizational power bases. Most perniciously, there is little sense of ownership. In fact, a static plan, which is often out of date shortly after it is written, determines the outcome, rather than intimate customer connections. The final decision maker is much closer to the CEO than to the development teams.

In cloud development organizations, executives tend to head cross-functional product teams, rather than functional silos. All the functions required to deliver the product or service report to them. (See Exhibit 1.) In addition, these leaders are responsible for revenue realization and the overall approach to marketing, sales, and channels. When they need to adjust to changes in market or customer demand, they have the authority to marshal resources without time-consuming and wasteful negotiations with other parts of the organization.

New Roles and Responsibilities

Conventional wisdom and practice dictate that software teams separate development, testing, and operations functions. To use a reference from *Top Gun*, develop-

EXHIBIT 1 | Cloud Organizations Eliminate Silos



Source: BCG research and analysis.

ers are akin to "Maverick," the creative risk-taking leader played by Tom Cruise, while testers fall into the background as "Goose," his reliable sidekick. This division of labor was ostensibly created to promote accountability for each type of activity. Instead, it caused management overhead.

Conventional wisdom is wrong. Cloud teams have developed a software engineering role that is responsible for not only writing but also testing and deploying features. The idea of relinquishing operational control to traditional developers can be daunting, so cloud teams build automated guardrails to ensure that testing and deployment are of high quality despite democratization of control. They have found that the productivity and quality gains of this organization design typically outweigh the risks. After all, who better to fix the code than the person who helped write it?

As developers take on more of a traditional testing role, a new "end-to-end quality engineering" role has emerged. Engineers in this role continually replicate and test the customer experience to ensure that the user interface, speed, response time, and overall quality are delighting customers. This role is crucial as customer environments become more complex and fragmented. Software teams need to ensure that their software works on public clouds, private clouds, on-premises servers, desktops, mobile devices, and multiple operating systems. The end-to-end quality engineers have this responsibility before, during, and after launch.

The product manager's role is also fundamentally changing at cloud companies, as software engineers assume more responsibility for scheduling and managing development. The product manager is no longer responsible simply for "hitting a date" but also for the business and operational success of the program. He or she has to take on a more strategic, analytical, and technical role. The product manager defines hypotheses and features that can be tested, prioritizes their development, and continually monitors actual usage of those features. This manager is responsible for

informing and training sales and marketing personnel, setting up partnership programs, developing a pricing framework, and monitoring pricing realization.

The role of data scientists is also becoming more prevalent at cloud companies. Their sole responsibility is to interpret the incoming stream of information. Unlike developers, they are not biased in favor of specific features and serve as honest brokers to determine which emerging trends will have the biggest impact. (See Exhibit 2.)

Living Software

To build cloud software, it's not enough simply to create new organization structures and roles that facilitate continual development and constant updating. Companies also need a software architecture that allows cloud teams to move quickly and independently.

A cloud "product," in fact, often consists of literally hundreds of microservices. Cloud development organizations create small teams—usually, just 10 to 15 members—that are responsible for a specific software module or service. The teams are able to release their software independently. If a service needs a larger team, then it probably has not been broken into small enough component parts.

Team Composition

All software teams are not staffed equally. Teams follow the same principles, but their composition will vary significantly depending on the type of software under

Exhibit 2 | The Shifting Roles of Software Development

	Traditional model	The cloud model
Product manager	• Oversees schedule and project management • Acts as proxy for customer	• Acts as product owner • Prioritizes features and services • Defines product strategy and differentiation • Encourages customer engagement and usage analysis • Acts as a bridge to marketing
Customer experience engineer	• Designs user interface, often late in the cycle	• Owns holistic user experience, beyond just user interface, across products and features • Engages early in the development cycle
Developer	• Writes code • Performs limited testing • Has limited customer interaction	**Feature engineering teams** • Write and deploy code • Define architectural design • Own both unit and functional tests • Oversee schedule and project management • Release the microservice or feature when ready **End-to-end quality engineering teams** • Validate customer scenarios • Run horizontal testing and validation during production
Tester	• Tests features • Has limited customer interaction	
Release engineer	• Ensures that releases are tracked • Tests the final bits to ensure high quality • Manages distribution of code	
Data scientist		• Analyzes usage data to identify trends and insights across features and services • Has a pure data focus without feature bias

Source: BCG research and analysis.

development. In our research, we found that software can be grouped into four archetypal categories on the basis of the complexity of customer interactions and who (the company or the customer) controls software deployments.

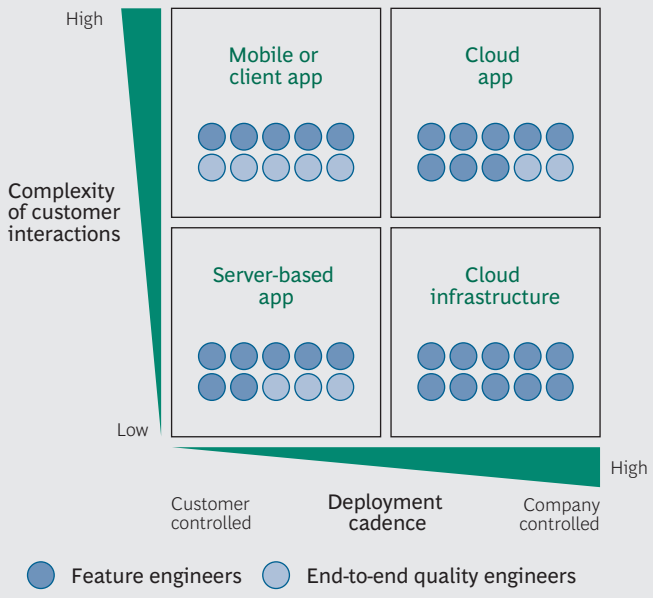
A common benchmark of software development teams is the ratio of traditional developers—whom we call "feature engineers"—to other members of the team. As Exhibit 3 illustrates, these ratios can differ greatly depending on the category.

When a software team internally controls an application or a service, such as consumer cloud software, developers can directly write, launch, track, and adjust code without engaging with the customer. This reduces the need for end-to-end quality engineers, as illustrated in the boxes on the right side of the exhibit.

At the other extreme, when the customer controls and deploys the application or service, as illustrated in the boxes on the left, feature engineers have less ability to modify the software after launch and must address a more fragmented user base. Accordingly, end-to-end quality engineers are necessary to serve as a proxy for customer needs and to monitor deployments.

This model applies to the individual microservices that compose a product, not to the product itself. A single product will frequently have services that fit within all four categories. For example, the team launching an app on the iPhone will have a very different composition from the team responsible for launching the app's underlying server infrastructure.

EXHIBIT 3 | The Composition of the Development Team Varies



Source: BCG research and analysis.

Modular Architecture

To support such decentralized team structures, cloud development organizations build loosely coupled software modules that interact seamlessly with other modules. The loosely coupled modules typically sit on top of stable infrastructure software. This approach allows developers to work flexibly—adding, replacing, and changing individual modules while maintaining high quality and reliability.

In this environment, teams are no longer forced into large-scale monolithic annual or quarterly release dates. Without the need to manage complex software integration, cloud teams can release features as they become ready. Features not ready for prime time are "toggled" off in order to control what the customer sees while preserving code integrity for easier version control.

Embracing Automation

Automation promotes speed in cloud software development, allowing teams to launch frequently and independently while maintaining high-quality products. It is often what separates the best software teams from the rest of the field.

Cloud teams invest heavily in tools to ensure that bugs and glitches are caught quickly. With a single click, developers' code can be checked in, tested, and deployed. Automation serves as a safety net that notifies developers of small problems before they become large. These systems are not cheap, but they are critical for teams that want to maintain quality.

Automation promotes speed in cloud software development, allowing teams to launch frequently and independently.

New products are generally released as "canary builds" that are staged so that only a small proportion of customers receive an update. If the software encounters issues, the automated system can quickly roll back deployment without affecting the entire customer base. If the software works as expected, it is automatically rolled out sequentially to larger subsets of users and regions until full coverage is achieved.

Automation improves the efficiency of software development teams by reducing the time they spend in manual tasks such as check-in, testing, and deployment. This downtime can reach 20 to 30 percent—or more than one day a week when developers are not using their creativity.

Connecting with Customers

The most successful software developers are not sitting in a silicon tower, imagining what customers want. They have an intimate understanding of the features that customers use and the roadblocks they encounter. These teams invest heavily in systems that generate real-time usage metrics, and they constantly monitor this data. Operational dashboards are generally updated every minute, while business dashboards are updated daily. The code from these teams is written in a way that allows detailed data and usage collection.

It should not be any surprise that cloud teams don't rely on "gut feel" to make decisions. They test a hypothesis and measure the result through frequent iterations

and A/B testing that compares usage of identical software except for one variation. At cloud organizations, even marketing, documentation, and sales approaches are tested in this way.

Cloud teams are also democratic in their dissemination of information. Test results and other operational and usage metrics are available for anyone in the company to see, opening the door for innovative thinking and cross-pollination of ideas.

Cloud teams are also democratic in their dissemination of information. Test results and other operational and usage metrics are available for anyone in the company to see.

All the Pieces Matter

In the television show *The Wire*, one of the main characters, a detective, frequently says, "All the pieces matter," meaning that every element of police investigations—every wiretap, witness, and piece of the puzzle—is significant. All these principles matter, too. To achieve breakthroughs in your software development, you need to have all four principles in place.

Carefully instrumented code allows cloud companies to track usage and performance in order to understand the market's reaction to their software. The way the code is written actually brings them closer to their customers. Modular software speeds the reaction time of developers. To respond quickly to customer needs, they can release new code without waiting for the other modules to be updated. Automation accelerates development time and efficiency, permitting developers to release their code rapidly, rather than waiting for other teams to catch up.

Collectively, usage metrics, modular software, and automation lay the groundwork for development teams to have end-to-end responsibility for their code, which improves innovation, customer responsiveness, and employee engagement.

Traditional software teams may not need to adopt a pure cloud model. But they should try to become speedier and more innovative in their development activities.

It's not just talent and legacy systems, structures, and processes that are preventing companies from fully adopting a cloud development model. It is also mind-set. Teams have to be willing to break down the historical divide between development and testing and to expand the roles and responsibilities of the software engineers. They must view testing as an ongoing function, not an afterthought. They must be willing to make significant investments in training, automation, and data-capture-and-retrieval systems. (To see how your team measures up in cloud development activities, see the sidebar, "The Transformation Journey.")

A company's success ultimately rests with its leaders. You need to challenge conventional ways of working and delegate decision making down into the organization. You need to give your best software developers a reason to be excited about going to work in the morning and staying late at night. The closer you come to adopting the cloud development model, the more success you'll have in achieving your goal of innovation and speed.

THE TRANSFORMATION JOURNEY

This way of working under the cloud development model is a significant departure from tradition. The principles are straightforward, but the transformation journey is not simple. Companies must change across four dimensions.

Principles. Communicate a focused vision to the development organization based on the principles of the cloud software-development model.

Metrics. Set up well-defined targets and timelines to reach them. The exhibit below outlines a set of leadership metrics that can serve as a barometer of your progress toward your goals. Your performance and incentive structure needs to be consistent with these metrics. For example, as developers take on greater testing responsibility, the quality and speed of their output may decline. However, this setback will be short-lived if you have the right metrics and monitoring infrastructure in place.

People. Redefine roles for developers, testers, and product and operations managers, and create a flatter organization structure on the basis of products and services rather than functional silos. Exhibit 3 can help you create the optimal team composition.

Technology. Automate check-in and deployment, create loosely coupled architecture, enable code sharing across teams, and write instrumented code that facilitates real-time analysis.

The goal of the cloud development model is to enable faster development and more powerful customer connections. By aligning your organization's principles, metrics, people, and technology, you can change the trajectory of your business and your relationship with customers.

The Best Cloud Teams Monitor Metrics

Customer connection	• *Engagement:* Measured by high user growth and low churn • *Usage:* Success is defined as first or second place in market share • *Customer Experience:* Defined by high customer-satisfaction levels, low levels of crashes, and strong performance on service level agreement
Data-driven decision making	• *Data Access:* Ease of data access across the team • *Experimentation:* All engineers can create a feature A/B test • *Data Availability:* Operational metrics available in less than a minute; business metrics available in less than an hour
Fast code releases	• *Cloud Services:* Continuous, often daily deployments • *Mobile or Store Apps:* Monthly releases, daily internal releases • *On-Premises Services:* Quarterly feature updates, monthly service releases
Engineering system	• *Build Time:* Occurs in seconds • *Check-in Time:* Takes less than five minutes • *Level of automation:* 100% • *Code Access:* All engineers can review and check out code

Source: BCG research and analysis.

About the Authors

David Mark is a senior partner and managing director in the San Francisco office of The Boston Consulting Group and the global leader of the technology sector. You may contact him by e-mail at mark.david@bcg.com.

Mike Quinn is a project leader in the firm's Detroit office. You may contact him by e-mail at quinn.mike@bcg.com.

Saurabh Shah is a project leader in BCG's Los Angeles office. You may contact him by e-mail at shah.saurabh@bcg.com.

Sanjay Verma is a partner and managing director in the firm's San Francisco office. You may contact him by e-mail at verma.sanjay@bcg.com.

Acknowledgments

The authors would like to thank the executives who agreed to be interviewed in the preparation of this report. These conversations enriched our understanding of cloud software and sharpened our conclusions. We would also like to thank our colleagues Thomas Krenik, Jérôme Moreau, and Daina Paulikas for their help in conducting the analysis and strengthening the arguments. Finally, we thank Astrid Blumstengel and Amanda Provost for marketing support, Mark Voorhees for writing assistance, and Katherine Andrews, Gary Callahan, Kim Friedman, Abigail Garland, Sharon Slodki, and Sara Strassenreiter for their editorial and production support.

For Further Contact

Please contact one of the authors if you would like to discuss the insights and conclusions of this report and how they can be applied to your company.

To find the latest BCG content and register to receive e-alerts on this topic or others, please visit bcgperspectives.com.

Follow bcg.perspectives on Facebook and Twitter.

5/15

BCG

THE BOSTON CONSULTING GROUP

Abu Dhabi
Amsterdam
Athens
Atlanta
Auckland
Bangkok
Barcelona
Beijing
Berlin
Bogotá
Boston
Brussels
Budapest
Buenos Aires
Calgary
Canberra
Casablanca
Chennai
Chicago
Cologne
Copenhagen
Dallas
Detroit
Dubai
Düsseldorf
Frankfurt
Geneva
Hamburg
Helsinki
Ho Chi Minh City
Hong Kong
Houston
Istanbul
Jakarta
Johannesburg
Kiev
Kuala Lumpur
Lisbon
London
Los Angeles
Luanda
Madrid
Melbourne
Mexico City
Miami
Milan
Minneapolis
Monterrey
Montréal
Moscow
Mumbai
Munich
Nagoya
New Delhi
New Jersey
New York
Oslo
Paris
Perth
Philadelphia
Prague
Rio de Janeiro
Riyadh
Rome
San Francisco
Santiago
São Paulo
Seattle
Seoul
Shanghai
Singapore
Stockholm
Stuttgart
Sydney
Taipei
Tel Aviv
Tokyo
Toronto
Vienna
Warsaw
Washington
Zurich

bcg.com