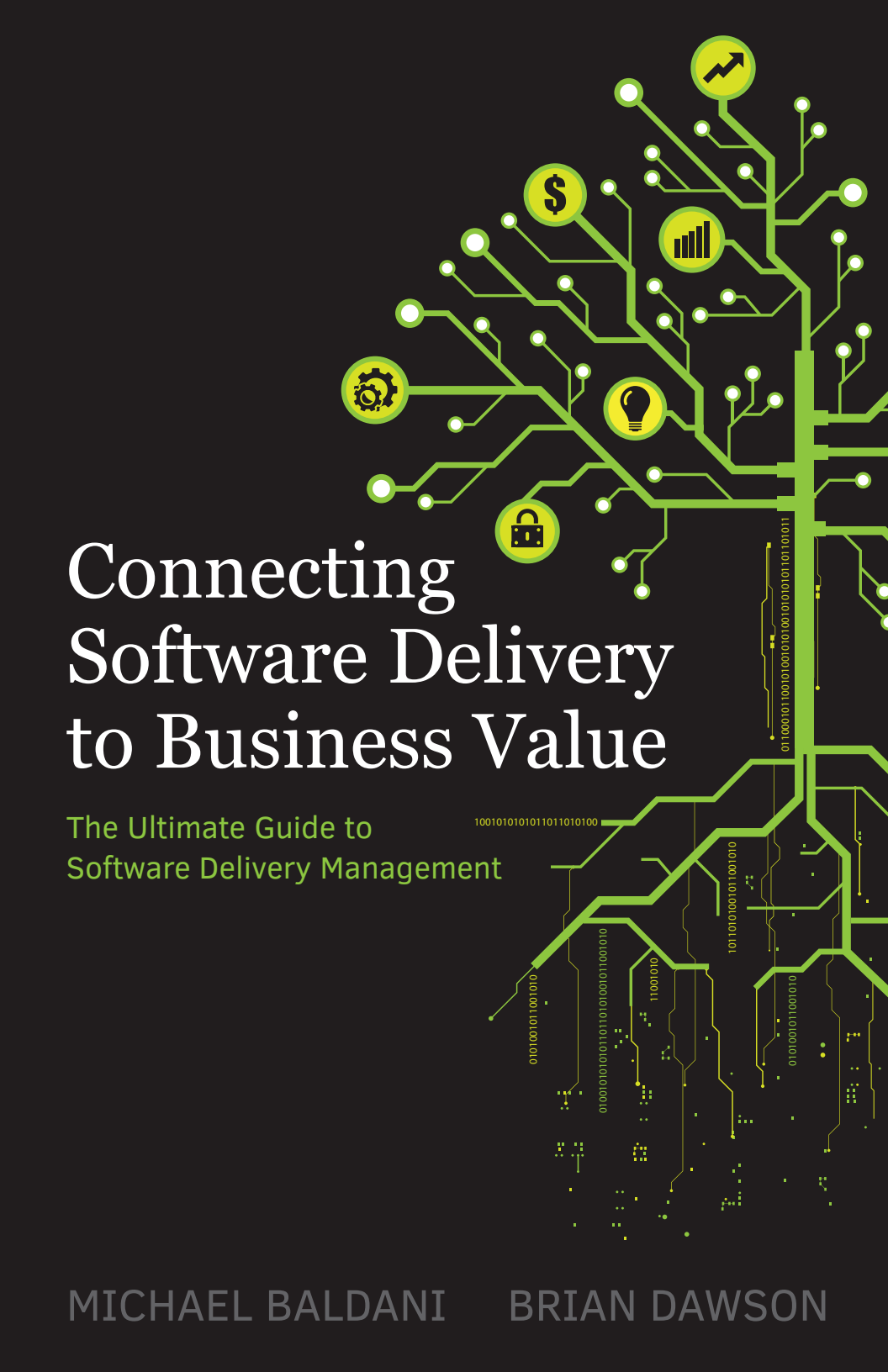


Connecting Software Delivery to Business Value

The Ultimate Guide to Software Delivery Management

MICHAEL BALDANI BRIAN DAWSON

Connecting Software Delivery to Business Value

The Ultimate Guide to Software

Delivery Management

Michael Baldani and Brian Dawson

Published by

Connecting Software Delivery to Business Value
The Ultimate Guide to Software Delivery Management

By Michael Baldani and Brian Dawson

Published by
Hurwitz & Associates, LLC
One Mifflin Place
Suite 400
Cambridge, MA 02138

ISBN: 978-1-949260-14-4

Printed in the USA by Hurwitz & Associates, LLC
www.hurwitz.com

Acknowledgments

This book has been a collaboration between the authors, publisher and editors. Some of the team members that helped make this book possible are:

Managing Editors:
Judith Hurwitz, Hurwitz & Associates
Daniel Kirsch, Hurwitz & Associates
Kiley Nichols, CloudBees, Inc.

Cover and Graphic Design: Kate Myers

Layout and Interior Design: Caroline Wilson

Table of Contents

About the Authors

Michael Baldani: Michael is a senior product marketing manager at CloudBees. He has spent the last 20 years in software marketing, and has recently been promoting and writing about data management, value stream management and DevOps analytics solutions to help business and engineering leaders overcome the challenges they face in their daily role.

Brian Dawson: Brian is the director of product marketing at CloudBees and a DevOps evangelist and practitioner with a focus on agile, continuous integration (CI), continuous delivery (CD) and DevOps practices. He has over 25 years as a software professional in multiple domains including quality assurance, engineering and management, with a focus on optimization of software development. Brian has led an agile transformation consulting practice and helped many organizations implement CI, CD and DevOps.

Chapter 1

The Challenges of Modern Software Practices

Inside

- Understanding the current state of software development
- Decomposing monolithic applications
- Gaining more value from DevOps
- Speaking the same language across your business
- Collaborating across the business

Companies are increasingly defined by the quality and functionality of their software. Software must have the ability to drive productivity and efficiency while supporting a positive user experience and customer satisfaction. The days when an organization could spend a year developing the next release of a product and introducing it to the market are long gone. Market demands, from both the customers and the competition, won't give any company the luxury of time.

Customers expect form and function - there is an expectation that software will include the latest innovations, to help them achieve the job at hand without getting in the way. This expectation requires companies to continually improve the features, functionality and usability of their software offerings. Relying on the success of yesterday's software offerings is dangerous. Nimble competitors are quickly creating more compelling offerings, more engaging experiences and software that solves customers' pain points. To be successful, businesses need to continuously innovate and improve their customers'

experience. Software has emerged as the fastest and most efficient way of doing this.

In this chapter, we will discuss some of the most significant changes in modern software practices along with the growing challenges in modern software development.

The Current State of Modern Software Development

Organizations have a wide variety of applications, ranging from legacy applications deployed on-premise to mobile, web, cloud, serverless and containerized apps. Over the last several years, businesses have adopted new development approaches to increase code quality, decrease errors and speed the development process. Many of these efforts have been focused on removing the barriers that often separate teams in various areas – for example: development, testing, operations and security.

Companies are increasingly adopting "cloud-first" initiatives, where new applications are developed, tested and run in the cloud. The cloud of choice could be a virtual private cloud or public cloud using one of the many options (Amazon Web Services, Microsoft Azure or Google Cloud, to name a few). The cloud gives teams rapid access to a scalable infrastructure with lower costs and labor. The economic benefits of the cloud are undeniable. In turn, a cloud-first approach requires a new model for software development and delivery.

The growing adoption of the cloud also changes both the development and deployment landscapes. In the case of hybrid cloud environments, where applications are hosted both on-premise and in the cloud, and/or multiple public clouds are used, managing across these varied

environments is challenging. Additionally, trying to move large, monolithic applications to the cloud is difficult and expensive. These legacy applications, which have many dependencies that are commonly poorly documented, were never developed with a distributed environment in mind.

Applications anywhere

This new era of application development means that software services can be developed and deployed virtually anywhere. Businesses want to have the flexibility to deploy applications on the platform or cloud that makes the most sense from a performance and economic perspective. They might, for example, deploy software on edge computing devices or on multiple cloud platforms, depending on customer demand and local regulations.

An application framework designed for change

With customer expectations continually increasing, businesses must approach software development differently. Adding a feature, improving interface design or enhancing security cannot require developers to fully redevelop the application.

Microservices enable applications to be developed as services linked together through lightweight interfaces managed within containers. Each service can be developed and deployed by independent teams. The microservices approach allows organizations to update portions of an application without requiring complex dependency management, full regression testing or a big bang deployment. In addition, services can be reused for different applications. For example, there is no reason to create a new authentication method for each application. Instead, reuse the authentication microservice that has already been tested and verified. If that microservice is

updated, it can more easily be implemented across all of the applications that utilize the service.

Operationalizing DevOps

In general, DevOps is a set of cultural principles focused on aligning the entire company, or at least the IT organization, around the shared objective of delivering quality software rapidly and reliably. The focus of DevOps is automation, efficiency and collaboration. Development no longer throws code over the proverbial wall to operations. Operations no longer throws it back when there is an issue. Instead, in a DevOps culture, teams work together to create quality software that is well-tuned for the environment where it will be deployed.

There are dozens of DevOps tools and software offerings, but at its core, DevOps is more about culture. A single tool, software package or simply moving to the cloud will not enable DevOps. Instead, companies need buy-in from every level of the organization.

The DevOps culture aligns well with the goals of successful companies. The most obvious benefits are an increase in the quality of software and faster delivery of new functionality. Performance, usability and security errors are identified earlier in the development process. In addition, taking advantage of automation helps to manage complex distributed environments. Most importantly, DevOps positions businesses to quickly respond to customer feedback and changing market conditions at the tactical level.

The emergence of DevSecOps

Despite the continued adoption of DevOps culture, some organizations still view application security as a totally

separate domain. Unfortunately, this means that security might be implemented at the end of the DevOps process as opposed to being embedded throughout the software delivery lifecycle. Keeping application security as a siloed activity can prove costly for two important reasons: firstly, corporate and customer data may be exposed through a breach, and secondly, releases can be delayed by months if security wasn't considered throughout the development process.

A solution to the problems created with a "security as an afterthought" approach is the practice of DevSecOps. In brief, DevSecOps is the process of integrating security into the software development process. DevSecOps begins with a change in culture, founded in ongoing learning to raise security awareness throughout the IT organization. Security must be viewed as a team effort. In addition, the organization may identify security-savvy people who can champion the change in the IT organization's approach to security. Security must not be thought of as a roadblock, but instead a key enabler in meeting business goals. In addition to cultural changes, tools can then be used to automate security testing, detect vulnerabilities early in the development process and even automatically halt deployment of a release if problems are found.

Increasing the Value of DevOps

This modern approach to software development combined with automation has helped some organizations achieve continuous integration (CI) and continuous delivery (CD). Through increased communications built into the expanding DevOps culture, organizations are experiencing shared decision-making, business goals and greater collaboration, which leads to effective CI/CD. Finally, organizations can start to eliminate the long backlogs, slow development processes and manual and error-

prone steps that prevent them from rapidly delivering reliable software - a key component to achieving the agility necessary to respond to changing customer needs and market conditions.

But DevOps is not perfect. While the development and operations teams are communicating with each other, they often focus on different criteria and different issues. Logs and reports from development tools are meaningless to the operations team. Likewise, the development teams are not familiar with tools used by the operations team nor with the data they produce.

Over the past several years, there has been a proliferation of tools to help teams improve the quality of their software practices. These tools have been designed to meet specific DevOps team goals. For example, there are specific tools for bug resolution, continuous integration, testing and performance management. DevOps organizations have put together complex toolchains that can be made up of a dozen or more open source and proprietary tools. Each of these tools has its own reporting functionality, its own interfaces and its own logging. There is no standard DevOps language to help these tools communicate with each other. The figure below shows an example DevOps tool chain. Each step of the DevOps process has its own tool(s).

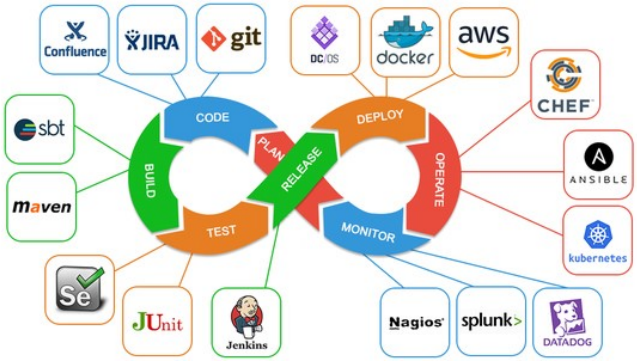


Figure 1-1: Example of a DevOps toolchain

The reality of the modern enterprise is that there are multiple teams, in functional groups with multiple technologies and in multiple business units, all using tools and processes purpose-fit for their teams' needs.

However, technology leaders then face a significant challenge: how do they integrate data and processes from these disconnected tools and teams? It is not realistic to have every participant in the Software Development Life Cycle (SDLC) familiar with each tool. For example, developers do not need to know the intricacies of a company's business intelligence tool. However, developers can benefit from the insights some of those tools provide. Similarly, operations can benefit from insight into the development work done within, for example, a continuous integration tool or testing tool.

Although it would make managing an environment easier, it is not always practical to have every team use the same tools or same process. Teams are most efficient when they can use the best tools for the job at hand.

The biggest challenges with modern software development stem from the fact that different teams are siloed and using different tools. These challenges include:

- **Lack of insight into the development process.** Managers are unable to quickly view the progress of projects. They can gain some insight during daily stand-up meetings, but managers must rely on others to understand the status of a project. Furthermore, identifying bottlenecks is difficult because of a lack of visibility.

- **Connecting software to business results**. The software organization is charged with developing, deploying and improving software. However, to be successful, the organization must tie their efforts back to business objectives. Yet the value of software is rarely tracked and developers seldom get business insight into their work.

- **Organizational structure bleeds into offerings**. Customers do not care about a company's organizational chart; however, disconnected processes can manifest in disappointing offerings. Customers want software that works; they don't want to know that a failure occurred because of a specific team. Therefore, teams from across functions need to work together on the common goal of delivering excellent software.

Expanding DevOps Culture Across the Organization

As we've discussed, DevOps is a culture-first movement to help development and operations teams collaborate more closely. While collaboration within the IT team is

important, in order to thrive, businesses need to collaborate across IT and business teams.

Organizational leadership is beginning to ask questions that are hard to answer. Today they expect answers - answers that can only be derived by gaining insights into software delivery status. Gaining visibility within business areas that previously didn't have that visibility can improve company results. The goal is to address opportunities to improve customer satisfaction and ultimately increase revenue. Some scenarios include:

- What if the marketing team could see the progress in the development of new software features and product offerings? Wouldn't it enable better outcomes if the marketing team could plan and align the timing of marketing campaigns with the actual development process?
- What if customer success managers could provide customers with real-time progress of feature requests without the need to track down multiple people within the development organization?
- What if organizations could connect software development budget with business outcomes? Are there KPIs that connect development with business results? This would provide intelligence to know what areas of software development are truly delivering value, based on meaningful KPIs.
- How can technology leadership streamline processes and gain insight into multiple projects without needing managers to manually consolidate reports and spreadsheets?

To be successful, businesses need to increase customer satisfaction. While moving from quarterly to weekly updates may sound impressive, frequent updates don't

necessarily translate to business value. If teams are still releasing poor quality software that users don't want to use, it doesn't matter that they are releasing it more quickly. What organizations really want is a connection between the DevOps efforts and their business KPIs. This has been difficult to achieve for most organizations due to a lack of intelligent integration between software delivery processes, the tools used within it and actual business outcomes.

A new way of managing software delivery is emerging to help businesses automatically capture, integrate and correlate disparate DevOps toolchain data and business data. This new approach to managing the entire software delivery life cycle is called Software Delivery Management. This book is intended to help modern enterprises quickly understand what Software Delivery Management is all about. A Software Delivery Management practice will enable software teams to leverage data from the DevOps tools they are using and to boost visibility into KPIs, enabling organizations to better manage both business and product success, while also enabling better visibility and collaboration.

Chapter 2

Understanding Software Delivery Management

Inside

- Recognizing the need for Software Delivery Management
- Understanding the goals of adopting a Software Delivery Management framework
- Introducing the four fundamental pillars of Software Delivery Management
- Explaining Software Delivery Management

Businesses are constantly changing and adjusting software offerings based on the needs of a variety of constituents. Many of these changes are incremental and barely noticed by users. Continuous delivery and DevOps seek to iteratively improve the user experience and add new functionality more frequently. However, it's imperative this is done without getting in the way of users and potentially driving them to choose other solutions. In this chapter, we will talk about what Software Delivery Management is and why businesses are starting to adopt this framework. In addition, we will introduce the four fundamental pillars needed to support a successful Software Delivery Management practice.

The Need for a New Approach to Managing the Software Delivery Lifecycle

Making frequent software updates to quickly respond to changing customer needs and shifting market dynamics is a drastic difference from waterfall development, where

there are typically one or two major product version releases a year. This new approach to software delivery created the need for continuous improvement, meaning small enhancements are made across each step of the software delivery lifecycle, to incrementally deliver better software, more frequently. This gets new functionality into the hands of users more quickly and also reduces the risk inherent in "big bang" releases.

Continuous integration (CI) and continuous delivery (CD) have enabled successful development organizations to quickly iterate on software and release frequent updates. CI helps businesses to engage in continuous improvement by having developers commit code more frequently. Continuous delivery is the practice of keeping software in a release-ready state at all times. CI/CD requires automation to assure quality, security and smoother deployments across the software delivery lifecycle. This approach gives businesses the ability to push reliable new software updates more frequently.

When businesses shift to a continuous improvement mode, teams from across the business need to operate on the same cadence. Aligning upstream and downstream functions (i.e. leadership, marketing, finance, sales, support) with development and operations, on a common set of goals and expected outcomes, is critical for success.

How can marketing know when a major new release is ready? How do business partners know when new product enhancements are generally available? Even more importantly, how do you connect development to customer satisfaction, revenue and, ultimately, business success?

Defining Software Delivery Management

Software Delivery Management is an emerging practice that aims to help businesses and their cross-functional teams better collaborate and align processes with software development, while also connecting and measuring development effort to business KPIs. Businesses have to reimagine their approach to software delivery as a new way to align teams (IT and non-IT teams alike) around the software delivery process and, ultimately, business change. While many organizations have adopted DevOps methodologies to accelerate application delivery and improve code quality, Software Delivery Management extends further to solve a greater challenge.

A Software Delivery Management practice enables efficient, continuous delivery of software across all teams, tools and technologies while integrating wider business functions. A successful Software Delivery Management framework links all teams and tools with a common data model and better visibility, to provide a platform from which it can all be managed and measured. This common data model enables unified processes, leading to improved collaboration across all teams. The end result is to measure and understand how software development and delivery effort translates into value for the customer and the business.

It's important to keep in mind that a Software Delivery Management practice does not revolve around a single tool. Software Delivery Management provides a framework for bringing together the data from multiple tools into a common platform. With access to a common data model, an organization's ability to effectively and efficiently manage the software delivery lifecycle is transformed. To achieve this, though, means everyone needs access to common data via common connected processes.

The end result is universal insights into data associated with software delivery and the ability for all functions within the organization to collaborate more effectively, based on that data.

Removing Silos with Software Delivery Management

As mentioned, Software Delivery Management links all teams and tools to a common data layer to enable better visibility for all. For many years, businesses have emphasized the need to break down silos and empower teams. However, one of the major hurdles to breaking down silos has been the inability for teams to share insights across multiple and disparate technology platforms.

With dozens of tools available for different business functions, it is nearly impossible for teams to collaborate if they are each deeply entrenched within a different technology platform. These challenges were highlighted in a 2020 survey conducted by Accelerated Strategies Group and commissioned by CloudBees. Eighty-four percent of respondents said the inaccessibility of information got in the way of their ability to do their jobs and/or make data-driven decisions. Respondents attributed this to organizational and functional silos impeding the free flow of information to IT practitioners and senior leadership.

For example, the sales organization will be very familiar with the company's customer relationship management (CRM) platform and all of its reporting capabilities. They likely have weekly, if not daily, meetings that are centered around metrics pulled from the CRM system. Likewise, the development team will have daily stand-up meetings in order to check on progress and track goals. You might see the problem emerging already. The sales

team is deeply engrossed in their own tools and reports while development teams understand processes based on the information and reports from their preferred tools. Getting these groups to collaborate on shared objectives is nearly impossible because they have very different roles – and they rely on completely different sets of data, from completely different tools. However, one is selling what the other produces - so they are, indeed, connected.

Adopting a Software Delivery Management framework allows teams from across the business to align their goals and resources based on insights derived from unified data. A successful Software Delivery Management practice provides teams the freedom to use the best tools for the job at hand, while also enabling them to collaborate across the business. Software Delivery Management is an imperative in the era of continuous integration and continuous delivery.

Understanding the Goals of a Software Delivery Management Framework

The goal of Software Delivery Management is to unify the process of software delivery and align it with business objectives. As already stated, a Software Delivery Management practice also fosters collaboration across teams in all functions of the organization. To implement Software Delivery Management, organizations require a Software Delivery Management platform to collect and hold unified data, provide insights for all and enable cross-team collaboration.

A Software Delivery Management practice implemented around a Software Delivery Management platform gives stakeholders the visibility into the software delivery process they need to make informed business decisions. Through role-based dashboards and reporting

capabilities, business stakeholders will be able to better understand the connection between DevOps, software delivery and business results.

The Four Pillars of Software Delivery Management

As we have mentioned, Software Delivery Management is not a single product or offering. A successful Software Delivery Management framework requires both technology and a new way of rallying teams around a unified mission. Software Delivery Management is based on four pillars or principles. These four pillars build upon each other and form the foundation for a Software Delivery Management practice. The four pillars are as follows:

1. **Common Data**
   A common data model is the foundational piece that enables Software Delivery Management. Teams across the entire organization must have the data from their line-of-business feed into a shared data model.

2. **Universal Insights**
   The common data model enables shared insights across teams. Everyone within the company is able to access and understand the same data, set specific goals for their areas based on that data, gain visibility into project status and measure outcomes.

3. **Common Connected Process**
   Once everyone is able to view the same data, it is easier for teams to work on a single, shared mission. Processes from different teams begin to be aligned as there is full insight into the project progress.

4. **All Functions Collaborating**
   Everybody across the organization is able to work together on shared goals. It is an iterative process of

continual improvement, where goals are established, work proceeds and outcomes are measured. Goals are then realigned based on market and customer feedback.

The next four chapters will go into greater depth on each of these fundamental pillars.

Chapter 3

Pillar 1: Common Data

Inside

- Understanding the importance of a common data model
- Defining a common data model
- Tackling the challenges of creating a common data model
- Setting the foundation for Software Delivery Management

The first pillar of an effective Software Delivery Management framework requires an effective data model. The ability to centrally manage key business data helps form the base that allows cross-functional teams to have universal visibility into the process and collaborate more efficiently. If you do not have this centralized data model, important information remains locked within domain-specific tools.

In this chapter we will discuss the central role of a common data model when planning your Software Delivery Management framework. We will discuss what it means to unlock siloed data from across specific applications and how this centralized data can serve as a rallying point for multiple IT and business teams.

The Need for a Common Data Model

It is typical for a single DevOps organization to use dozens of different tools to support different aspects of the software development and delivery lifecycle. Traditionally, these different tools have resulted in silos of data.

For example, teams working with Jira for bug tracking aren't necessarily going to understand the data coming from a configuration management tool like Chef or Ansible. Expecting DevOps teams to learn the outputs of dozens of applications does not make business sense and shifts their focus away from addressing customer needs. By providing a common data model that can decipher the context of all of this data, organizations enable all constituents - including developers, sales executives and marketers - better visibility into the software delivery process and allow them to effectively collaborate. The result is an organization focused on driving the adoption of their products and features to their customers, knowing how customers are using them and how the development effort translates to ROI for the business.

Defining a Common Data Model

A common data model is a data repository where data from all the business applications and DevOps tools can be ingested and made accessible in a consistent way. A common data model forms the foundation of a Software Delivery Management practice. This data strategy helps to unify data across all your development and business processes, providing teams the visibility they need to truly collaborate.

Traditionally data is locked within domain-specific applications and platforms. Software developers have little insight into the data that customer success teams have. Developers cannot easily look at customer reviews and feedback on recently released features. Time and effort is spent trying to find and understand these applications in order to extract that data from business applications such as CRM and other customer-facing tools. Likewise, customer success teams do not know when new features will become available or when the next update will be

generally released. If a customer has a problem when the new release goes out, the customer success team will need to reach out to a variety of colleagues who can track down that information and resolve the customer issue. Time is critical in order to keep customers satisfied.

So how does the common data model fit within your organization's business and an application model? Figure 3-1 helps to illustrate the inputs to the common data model. As you can see, data from across the company funnels into one common data model. It's important to keep in mind that this approach to common data doesn't just collect DevOps data; data from marketing, support, planning and other key teams are all also moved into the common data model.

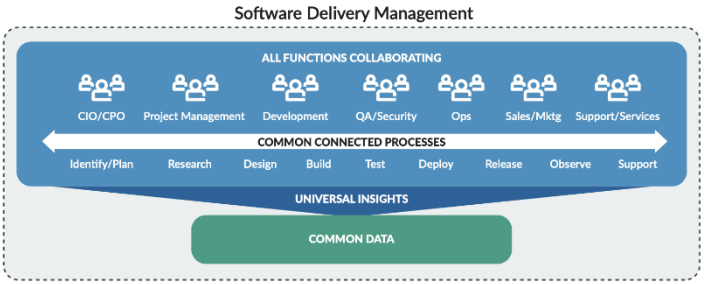


Figure 3-1: The common data model brings data together from across the entire business

In short, the common data model acts as the system of record to democratize and share data that can be accessed by cross-functional teams across the business. The ability to give teams access to key data from across business silos gives stakeholders the context to make data-driven decisions. Development teams have the context related to features customers want, while marketing and sales have visibility into when important features are ready for release. Neither group needs to guess because

they have access to the information. The common data model provides teams with the data they need in a context that they can understand.

Creating a Common Data Model

Creating a common data model requires thoughtful planning. There is not a universal language between business tools – there are few standards, even between various DevOps tools. The log data coming from a planning tool is going to look very different from data that results from application testing tools. To complicate matters, tools outside of the DevOps organization, such as marketing automation and CRM platforms, were never designed to be understood by DevOps teams.

The first step in creating a common data model is to understand all of the tools that are used within the software delivery process. This discovery phase doesn't just involve surveying DevOps teams. Instead, an organization needs to understand the role of different tools and how they work to form a software delivery tool chain. What are the key data points that these tools contain?

While building out a Software Delivery Management framework, organizations also need to understand the tools that teams outside of the DevOps organization use. For example, what are the critical marketing planning tools and customer service tools? What does the critical data look like within those applications?

As businesses identify key data points within each tool, they must form a plan to connect that data with their common data model. In addition, they must provide enough context to make that data useful for other teams. For example, the customer success team might constantly focus on "Customer Health Score" and "Net Promoter

Score," but those metrics have little meaning to other teams.

After companies understand their teams' most important applications and key data points, harmonizing data from various domain tools will emerge as a challenge. Simply coming up with a consistent term for "customer" may be a challenge. Some applications will call a customer a "client," or a "customer," while yet other applications might just identify them by "business name." Likewise, "build," "release" and "deployment" may all mean the same thing but will be labeled differently within applications. Further complicating the matter, a team might have a specific definition that differentiates the "build" and "release," but another team will have different definitions. How can you have a common data layer when teams within the same business are literally talking different languages?

Consistently defining data across applications may appear simple, but as your Software Delivery Management framework grows, you are going to need to think about hundreds, if not thousands of words that need to be consistently defined and understood by very different internal teams. It is especially critical to have a common understanding of business data no matter how that data is being used. For example, both the sales team and the accounting team need to have consistent data definitions.

Understanding Why a Common Data Model is a Core Requirement

Remember, a common data model is critical to creating the visibility needed for better collaboration in a Software Delivery Management framework. The establishment of a central data repository that contains key development, application and business data enables your

software delivery management team to achieve important business and technical goals. Without rationalizing and sharing critical data points and definitions, your organization will be forced to deal with fragmented and inconsistent data.

One of the primary benefits of adopting a Software Delivery Management framework is that it automatically captures and stores information in the common data model. As a result of this automation, it is now possible to enable:

- Universal insights across IT and business teams
- Connected processes and
- Collaboration between all functions across your business.

This foundational data model is critical because each tool within the DevOps tool chain has its own logging and reporting capabilities. To deliver insights across the entire modern software value chain, the system must bring together data coming from teams in every business unit across the organization.

Chapter 4

Pillar 2: Universal Insights

Inside

- Recognizing the need for data alignment
- Benefiting from the power of common insights
- Delighting customers by leveraging universal insights

Establishing a common data model is the first pillar when creating your Software Delivery Management framework– it sets the foundation for the remaining pillars. Democratized data from the DevOps toolchain provides the visibility to enable teams from across your business the ability to align their goals and share ideas. It also provides you with the ability to have a shared vocabulary so that teams can effectively communicate. However, data itself does not drive value.

Data needs context and needs to be understood by teams in order to help drive results. In this chapter, we will discuss the second pillar of a successful Software Delivery Management framework – universal insights.

Understanding the Needs of Different Teams

The insights that your Software Delivery Management framework will provide to different teams will depend on their role and what's needed to accomplish the task at hand. The needs of your management team will be very different than your application security team. Likewise, the way that different teams want to consume data will vary. For example, a sales team may want insights to be

built into a CRM platform, while other teams may prefer importing the data into their tool of choice. On the other hand, managers and executives may want easy-to-consume dashboards that are customized to display the KPIs that they care about most.

Defining Universal Insights

Universal insights provide the visibility that enable teams across all functions to understand how key software delivery projects are progressing. Universal insights can also provide data on how various features or releases are being used by customers in production. Finally, universal insights can provide visibility into the value returned to the business from various software projects. Dashboards, reports and ad-hoc data queries give employees the necessary information to make informed decisions, in order to align resources to drive the adoption of software products and features by customers and users.

Similarly, having these insights will help differentiate your business from the competition. Teams will be able to execute on projects more quickly, spot emerging user trends and respond to changing customer expectations when they share analytics into data that span across the organization.

The Power of Universal Insights

Gaining insight from data is not a new concept. Data visualizations, business intelligence, analytics and advanced analytics have been hot areas of investment for years, for businesses that want to create differentiation. However, in nearly every case, these data insight projects do not support the needs of all teams and functions across the business.

Giving all business stakeholders insights into the businesses and their specific jobs helps to align every team with a common set of business goals. The requirement for achieving the maximum business power from universal insights is twofold:

1. **Data must be presented in a way that is easy to consume**. For example, Dev and Ops teams may want to view data related to development progress and security scan thresholds, while sales and marketing leaders will want sales and revenue data related to a software product to be presented within the CRM system.
2. **Data needs context to provide maximum value**. Without providing additional meaning, data insights might be interesting but will not drive success. For example, DevOps teams should receive sales and revenue information, but tying that data directly to applications and development efforts gives teams actionable information.

The impact of this insight will improve collaboration across business and technology teams, support and operational groups, resulting in better business outcomes and more satisfied customers.

Even executives who may not ordinarily get involved in low-level details should be able to explore data and potentially intervene if they see an emerging problem. For example, because of experience, leadership may recognize a problematic pattern before its consequences become clear. They may, in effect, keep a project from falling behind schedule, prevent its going over budget or save a promised feature that is critical to an important customer by reallocating resources.

How Universal Insights Drives Business Success

Let's look at the example of an employee in the role of a customer success manager for a SaaS vendor. The job of the customer success manager is to drive adoption of the software, keep the customer happy and ultimately expand use of the software within the account.

A key customer has several feature requests of varying priorities. The client outlines the requirements with the customer success manager and the need to have the software work within the company's current business processes. The client is committed to fully migrating to the vendor's SaaS platform if the software can be tailored to the company's needs.

The success manager would normally have little insight into the progress of the requests. Before bi-weekly customer calls, the manager needs to send out emails, reconcile spreadsheets and track down a half dozen co-workers in order to provide the customer with an update. This process is not only time-consuming, but it also gives the manager no real-time insight. Information that was provided to the customer was at best several days old.

Because of these challenges, the SaaS vendor adopted a Software Delivery Management approach. Now the manager is able to track requests and feedback from across her clients, on an up-to-the-minute basis. The manager also has information that empowers her to have meaningful conversations with development teams so they have the context about the client's requirements and challenges the new feature will solve. Additionally, the manager always has real-time insights into feature progress and status. The manager can confidently assure the customer that its desired feature is on the way. This type

of clear communication with customers drives customer satisfaction, increases renewal rates and decreases churn. One of the company's universal principles - putting the customer first - has been demonstrated to its clients. This type of client-vendor relationship, powered through a Software Delivery Management practice, is how the SaaS vendor is working on differentiating itself from the competition.

Chapter 5

Pillar 3: Connected Business Processes

Inside

- Seeing the importance of business processes
- Understanding the need to align processes
- The value of connecting processes

In the previous two chapters, we discussed the requirements to create a common data model and to gain needed access to shared universal insights. While those are necessary steps as you advance your strategy to create a Software Delivery Management framework, connecting business processes is where your business will begin to gain significant value.

In this chapter, we will discuss the importance of business processes and how integrating business processes from across your organization can drive value. It's important to remember that the first two pillars - a common data model and shared universal insight - set the foundation for connecting business processes.

The Importance of Aligning Business Processes

Well-designed and well-executed business processes that are aligned across an organization's functions can differentiate your business from your competitors. They enable you to provide a unique and valuable product or service to your customers. Reimagining business processes is how born-in-the-cloud startups have disrupted industry

stalwarts. If you look at unicorn startups, they surely rely on technology, but at their foundation are business processes that make it easier for customers to conduct business. The technology enables the transaction, but technology alone is not the differentiation. Likewise, well-aligned business processes that connect the various functional groups in an organization, along with incorporating industry and customer experience, are how established companies can and should quickly respond to evolving customer expectations.

It's likely that you haven't considered the different business processes that exist within your organization. You may think of just a couple – like closing a deal (which includes everything from prospecting to closing, payment and product delivery) or onboarding new employees.

However, there are dozens of business processes entrenched within each function of your organization. Business processes range from product and customer research to marketing, sales and support services to finance and more. Each business process is supported by software – whether it is a complex, customized customer relationship management platform or a well-established Gantt chart.

Some of these business processes are easily executed by a single person within a team. That individual might not even consider their task as a defined "business process." For example, accounts payable received invoices, checks their validity, confirms the account and distributes funds. On the other hand, other processes require an entire department, or even multiple departments, to coordinate efforts.

Let’s take the example of a major software release. Think about the number of people and departments involved in the processes of bringing this release to market. Everyone from executive leadership to development, operations, marketing, sales and legal must be involved. Although you may have an overall process to ensure a timely release of the update, each department will have its own business process to execute the needed work. Many of the department-level tasks are supported with function-specific software packages and individually adopted best practices.

Connecting Business Processes to Meet Customer Expectations

When departments depend on their own disconnected tools, that's when communication, deadlines and product quality all suffer. In many cases, it’s as if different parts of an organization are speaking different languages. At the end of the day, this disconnection results in poor customer experience and satisfaction. Customers don’t care if a feature request is held up in testing or debugging, and they don’t care if an update is ready but marketing is finalizing written materials – they simply want the enhanced capability.

Connecting and aligning the cross-functional business processes is an essential element for establishing your Software Delivery Management framework. Without connecting processes, it is impossible for different parts of the organization to know how a project is progressing. While it might be okay for a small, team-level process to be siloed, departmental and, more importantly, business-wide processes must be connected and aligned. If you are trying to establish a common goal across your business, you cannot rely on independent processes within each team.

Let's continue the example of a major software update. You may have biweekly leadership meetings between the development, marketing and sales organizations to discuss the progress and customer requirements. Sales plans will begin to go into process well before the release is generally available. However, things change, and with disconnected departmental processes, alerting teams to those changes can be delayed by weeks. You may have teams spending time on recently irrelevant work or, worse yet, you may communicate inaccurate information to partners, customers and prospects.

In addition to project change, organizations are constantly tweaking business processes in a constant search for better efficiencies, cost savings, improved services – things that make your business successful. Connecting business processes allows you to execute this change across the organization in a predictable and planned manner.

The Business Value of Connected and Aligned Processes

The value of connecting and aligned business processes is likely becoming clear – it is the foundation for efficient and effective collaboration. While the first two pillars of a Software Delivery Management framework align teams on data and the understanding of that data, this third pillar brings cross-functional alignment into everyday business practices. In short, teams have the right information, and they know what they are expected to do, how to do it and when. By connecting and integrating processes throughout the organization, your teams are unified around a common goal and a pathway to achieve that goal. Ultimately, unifying business processes leads to faster responses to customer requirements and better client outcomes.

Chapter 6

Pillar 4: All Functions Collaborating

Inside

- Creating the foundation for successful collaboration
- Satisfying customer needs by working together
- Seeing the value of collaboration

The final pillar of creating a Software Delivery Management practice is to achieve departmental collaboration across your organization. Collaboration is one of the most important aspects of achieving market success. It is clear that connecting software delivery with other departments through unified data drives better collaboration. This level of interdepartmental teamwork means that everyone is working with the same data and has real-time insight into software releases. This collaboration is the keystone to accelerating time-to-market and faster time-to-revenue. The ultimate beneficiary of working together are your customers, who will receive better support, higher-quality updates and faster turn-around on feature requests.

In this chapter, we will focus on the final pillar of a successful Software Delivery Management framework – all functions across your business collaborating.

The Journey to Collaboration

Many businesses have long strived to improve collaboration between departments. A business leader may hire

consultants, conduct workshops and rethink their organizational structure in an effort to increase collaboration. The key to successful collaboration is two-fold: you must have teams fully embrace a common goal, and you need to implement technology that makes working together easier.

In the previous three chapters, we discussed the first three pillars to build a successful Software Delivery Management framework. These building blocks help organizations ready themselves for meaningful collaboration that will benefit the bottom line. The following is a brief explanation of how the first three pillars help organizations collaborate across teams:

- **Common Data**: The common data model brings together data that is typically held within domain-specific tools. This data is normalized so those insights can be gleaned.

- **Common Insights**: Giving teams access to critical business data in a way that is easily consumed allows them to make fact-based decisions. In addition, providing analytical insights into tools and processes that are normally held within different departments gives teams the ability to collaborate more intelligently.

- **Connected Processes**: By connecting business processes that are typically held within separate departments, teams are able to execute more quickly and eliminate redundancies.

Standing up each of these pillars requires teamwork. For example, consider the process of creating a common data model. Each team needs to work together to share and understand the different tools they use, the types of data that are important to them and how they can contribute their data to a shared repository. In some cases, teams

might be reluctant to share their data – they are accustomed to being judged on their team performance and don't want to spend time on a shared effort. In addition, teams might be concerned that their data will be misused or misconstrued, or that the shared data will not reflect their reality. To overcome these challenges, it's important to align everyone on a common set of goals and provide metrics that can show how their efforts contribute to those goals.

The Business Imperative of Collaboration

Every business unit is able to achieve some level of collaboration between departments. As you create your Software Delivery Management practice, the value of that collaboration will accelerate. In addition, a Software Delivery Management practice will help bridge the gap between software and business teams. Software and business leadership are able to look at the same data and have enough context to understand data that is typically siloed within functional tools. Business leaders are also not dependent on software leaders for updates and reports. Instead, business teams are able to get up-to-date visibility into the software process and progress in ways that they can understand.

Creating Positive Customer Outcomes

Delivering transparency between different parts of your business means that teams can focus on customer and business growth. Let's look at the example of bringing a new capability to market.

Some departments will already have a cadence and familiarity with teamwork - for example, sales and marketing or development and operations. However, think about the benefits of marketing working closely with product

management. What if marketers had the ability to see in real-time the progress of a new release? Marketers could better time their efforts and tailor their campaigns to match the evolving efforts of the product team. In addition, if customer success teams have insight into product development, they can share, as appropriate, those insights with customers. Furthermore, by collaborating across departments, customer feedback can be better incorporated earlier in the development process. Creating a platform that encourages collaboration across development, operations and business has the potential to transform the way organizations create synergy and context to support the need for flexibility and agility.

Chapter 7

Five Steps to Establishing a Software Delivery Management Framework

Inside

- Understanding your current state
- Aligning stakeholders
- Achieving buy-in from business and technical teams

Creating your Software Delivery Management framework is not an overnight project. The journey to rethinking your software value chain requires careful planning. To help you think through your approach to Software Delivery Management, in this chapter we share five steps to get started.

Step 1: Align Teams on Software Delivery Goals

Before you can plan your future, you have to understand the goals, requirements and expectations of each constituent within your organization. It's important to remember that the goal of a Software Delivery Management framework is to help align all teams on a common set of goals; therefore, you must understand the challenges that different teams face. It's important to bring together team leadership and begin by asking these questions:

- What are the common bottlenecks that slow down projects?

- What do other teams within the organization not realize about your processes? And what information would help them?

- How do you ensure that customers are happy and that their needs are met?

- How do you collaborate with teams in different parts of the business?

- Do you have the structure that allows you to move quickly?

By understanding where you are and what's working you can understand what you need for the future.

Step 2: Build a Common Data Framework that Can Adapt to Business Needs

The first pillar of creating a Software Delivery Management framework is to create a common data model. As a part of your planning process, you must begin to understand the types of data that are important for different teams within your organization. It's important to remember that this data model must be able to adapt as teams utilize new tools and rely on different types of data. As you begin to plan your common data model, you must start to understand the following:

- What types of data does your team manage – for example structured, unstructured, machine data?

- Do you have a single source of truth within your data or does your team need to reconcile data across several sources to normalize it?

- If you had access to additional data from different teams, what would be the most important information?

- Do you currently have data on how your work directly drives customer success?

- Do you have a standardized process for incorporating new tools and data sources into your data model?

Step 3: Giving All Teams the Insight They Need

Although some teams may be reluctant to share their data with other parts of the organization, the common data model provides the foundation to provide universal insights. Providing analytics-based insights doesn't mean publishing generic dashboards and reports. Teams must receive data that is both easily understood and helpful. To deliver useful insight that can improve the way teams work together, you should start by asking the following questions:

- How can data insights be embedded into a team's current workflow?

- Do your existing tools have plugins and interfaces that allow insights to be pushed down to users?

- What are the missing data points that teams need to respond to customers (internal and external) more quickly and with greater precision?

- Who, within your organization, possesses both the business context and technical know-how to lead your universal insights efforts?

- Are you providing data that is actually improving business results? How are you measuring the impact of your efforts?

Step 4: Understanding and Connecting Processes Across the Business

As you advance your Software Delivery Management practice, you need to begin to connect the way teams work. For some teams, processes may have already begun to integrate. For example, many successful companies have built DevOps organizations to connect development and operations processes. However, other elements of the software development chain still live in isolation. As you begin to connect business and technical processes, you need to think about the following questions:

- How does each team approach a new project?
- Are there common bottlenecks that can be eliminated by integrating a process with another team's workflow?
- Are there processes that are duplicated multiple times by different teams?
- How can you involve downstream processes earlier in a project to increase speed-to-market?
- How can teams that regularly interact with customers infuse customer feedback throughout the software delivery process?

Step 5: Drive Collaboration Across the Business

The ultimate goal of creating your Software Delivery Management practice is to drive collaboration across your business. Achieving departmental collaboration will not happen overnight. As you look to increase collaboration across teams, it is important to think about these questions during the process:

- What teams are open to collaborating and where can you see fast and meaningful results? For example, aligning product management with marketing can yield significant results.

- As you begin to break down team processes, are there obvious points of collaboration? It's likely that marketing and sales are already linking processes as they approach customers and prospects.

- What technical barriers exist for collaboration? Do you have the tools in place to allow for collaboration between functional teams?

- Do you have processes in place to measure the impact of collaboration? How are you defining successful collaboration?

Chapter 8

Best Practices for Successful Software Delivery

Inside

- Rallying teams to drive change
- Align goals across teams
- Improving processes to drive better business and customer outcomes

Creating a Software Delivery Management framework and executing on each pillar requires changes to both corporate culture and the way teams create and evaluate processes. In this chapter, we break down best practices into two major categories: change management and continuous improvement. It's important to remember that your Software Delivery Management practice will be an ongoing effort to improve collaboration, customer outcomes and, ultimately, the bottom line.

Best Practices for Change Management

Change is difficult, and even the most well-meaning teams within an organization might be resistant to it. Change management is an approach to planning and executing a transformation of your organization's goals, processes and technology. Creating a successful Software Delivery Management practice requires you to think about how you will promote change management across teams. Organizations that are already seeing positive results through their Software Delivery Management efforts have used the following three change-management best practices:

- **Incremental change,** rather than a rip and replace approach. It's important to change team processes gradually. You need to recognize that teams have created procedures based on their experience and expertise.

- **Quick wins** help to build enthusiasm for change. By exhibiting the value of increased collaboration and its impact on business results, you will get buy-in. For example, can you easily connect product management activities with customer feedback received by customer success teams?

- **Institutional alignment** helps rally everyone around a common set of goals. Although each team might be evaluated based on functional considerations, it's important to connect each team's role to organizational goals.

Best Practices for Continuous Improvement

Customer requirements, market expectations and competitive offerings are always changing. Therefore, to establish long-term success, the business must plan for continuous improvement. Continuous improvement is a practice of ensuring that business and technical processes and practices are as efficient and effective as possible. Along with other processes, your Software Delivery Management framework must evolve and improve. As you apply continuous improvement methodologies to your Software Delivery Management practice, it's important to keep in mind the following three continuous improvement best practices:

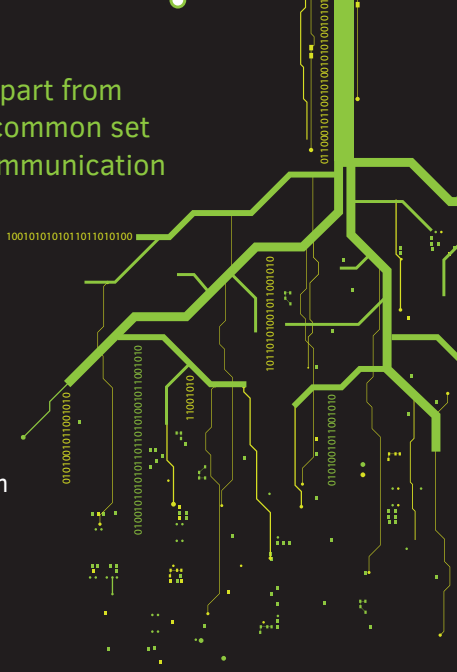
- **Evaluate toolsets** on an ongoing basis to make sure teams have access to best-of-breed tools

while eliminating tool redundancies. In addition, continuously ask teams what types of new data would be helpful for them and find ways to deliver insights from existing tools.

- **Measure the impact** of your Software Delivery Management practice. It is important to review key performance indicators (KPIs) to understand how process changes are affecting the business. At the same time, think about whether you are capturing the right KPIs and how you can share success across teams.

- **Rethink processes** based on your evaluation of KPIs, team and customer feedback. Adopt a regular cadence to evaluate business and technical processes to find efficiencies and areas where improved collaboration will positively impact customers.

Outstanding software sets businesses apart from their competition. Aligning teams on a common set of goals and increasing visibility and communication across teams are the keys to creating excellent applications that meet customer expectations.



Successful businesses are rethinking their software delivery process through the adoption of a Software Delivery Management framework. This new approach gives you the technical underpinning to enable collaboration throughout your business. Prepare your teams to collaborate around common data and a shared set of goals that are focused on customer and business success.

This book will explain:

- How insights can be delivered with context to make them actionable
- The business impact of connecting business processes
- The best practices for succeeding with Software Delivery Management

AUTHORS

Michael Baldani is a senior product marketing manager at CloudBees. He has spent the last 20 years in software and SaaS solutions to help customers overcome the challenges they face in their daily roles.

Brian Dawson is the director of product marketing at CloudBees and a DevOps evangelist and practitioner with a focus on agile, continuous integration (CI), continuous delivery (CD) and DevOps practices. He has over 25 years as a software professional in multiple domains including quality assurance, engineering and management, with a focus on optimization of software development. Brian has led an agile transformation consulting practice and helped many organizations implement CI, CD and DevOps.

CloudBees.

CloudBees, Inc.
4 North Second Street | Suite 1270
San Jose, CA 95113
United States
www.cloudbees.com | info@cloudbees.com

Published by Hurwitz & Associates

$14.95
ISBN 978-1-949260-14-4
51495>
9 781949 260144