Continuous Everything: Why Software Delivery Management Matters

Contents

3 Introduction

4 The Limits of CI/CD

5 What is Software Delivery Management?

9 How Software Delivery Management Brings Value

12 CloudBees and Software Delivery Management

Introduction

The widespread move towards an integrated DevOps approach to software engineering, building software fast, bringing it safely to market while combining it with maximum product adoption, is still fraught with bottlenecks and inefficiencies. It's not uncommon for a company to use more than 50 different development tools in an attempt to move faster and reduce errors, but each tool solves only one discrete step in the software development lifecycle while contributing to ever-increasing complexity. There is no shared data, no way to manage handoffs between teams and no single pane of glass to provide visibility into the entire process. In addition, the software organization often has no information about how the software fits into the overall business strategy or how it is used by customers.

This disconnect makes it hard for developers to build software that both delights customers and meets business needs. Continuous feedback loops are becoming the norm in development, yet getting feedback on how the new feature or application is being used or if it meets the intended business goal is elusive. Software Delivery Management (SDM) recognizes that software creation in the enterprise is a core business function, and extends the application feedback loop from development through customer adoption. The result: Software that meets the business needs, satisfies customers and drives continuous improvement in value and impact.

The Limits of CI/CD

There's no longer disagreement about the benefits of continuous integration (CI), continuous delivery (CD) and a DevOps culture. Adoption of CI/CD and DevOps gives companies the ability to deliver software faster, to react more quickly to market changes, fail faster, recover faster and generally iterate rapidly to produce higher quality, innovative software. Many companies practicing CI/CD and continuous deployment release updates to production as frequently as hundreds of times per day, instead of once a week or less; they go from commit to deploy in minutes or hours instead of weeks; recovery from downtime is measured in minutes, not weeks; and changes exhibit a failure rate of less than 15% instead of over 50% of the time.[1] Most companies recognize that embracing a DevOps culture is best practice. However, for all but the most tech-forward companies, the reality is messier.

In an enterprise setting, there are often a variety of development methods, tools and technology stacks being used to deliver a wide range of software to meet different needs, through different processes. Teams following DevOps "best practices" often do things completely differently from each other, even within the same company.

Even in those companies with a mature CI/CD pipeline, multiple daily deployments and a full, company-wide commitment to DevOps, there is often no end-to-end insight into the value stream — where products and features are stuck now or get stuck frequently, where delivery bottlenecks and inefficiencies slow down value delivery to end users. There is also a complete inability to understand how software affects business key performance indicators (KPIs). Does the application deliver the intended business result? Is the organization's net promoter score increasing or decreasing as a result of a new feature? Has it improved customer retention or click-through rates or conversions? Has an internal application actually improved efficiency? Do the intended users, whether they are internal or external customers, actually use it?

CI/CD helps us deliver better software, faster — but it doesn't ensure we are delivering the "right thing" or that the business need is being met. It doesn't pull together data and artifacts across the entire software delivery lifecycle from the many siloed tools an organization relies on, to provide a single overview with the contextual information that developers, product managers, operations teams, product marketers and support teams need. CI/CD doesn't provide the data needed to measure how well the software organization is creating value for the business — and without a way to measure, software organizations have no way to know if they are improving. You cannot foster the elite-level collaboration that should be an outcome of DevOps when there is no visibility across the stakeholders involved in delivering value.

Businesses don't just need speed and agility when it comes to software development — they need insight and intelligence so that the software has the right functionality, presented in the right way, to meet the business need it was designed to address. Unfortunately, in most cases the disconnect between tools, people and processes prevents the organization from performing at a high level. Developers are under constant pressure to deliver, but they are often isolated from the business motivations for the software they produce. They have no idea when a given feature is scheduled for release, or why. Once the application is finished, they don't get to see how users interact with it. This makes it harder to create an application that delivers on the business goals — and can be frustrating to the developers, who may not feel like their work contributes to the business in a tangible way or positively impacts users.

[1] *2018 Accelerate: State of DevOps Report*, DevOps Research & Assessment.

Software Delivery Management is a way to bring software development and delivery teams together, so that as software is developed there is continuous alignment across stakeholders driving the creation of business value.

On the other end of the spectrum, sales and marketing don't have a clear idea of when applications will be ready, how the features will actually work and how to promote them, while support teams struggle to get customer feedback addressed and incorporated into the next release.

At most companies, software developers, business leaders and other stakeholders have learned to live with this disconnect and work around it as best they can. But "as best they can" is often not good enough, at all.

And yet... software development is a critical part of modern enterprises. It makes no sense to isolate the development and delivery of software from other business goals. No business would keep the rest of the organization in the dark about the effectiveness of their key focus — why would we do so with the software delivery function? Indeed, as engineering organizations implement company-wide digital transformations, they've begun to realize CI/CD and DevOps only break down silos within (maybe) engineering. The need goes beyond engineering to other business units like product marketing and customer support as well.

Software Delivery Management is a way to bring software development and delivery teams together, so that as software is developed there is continuous alignment across stakeholders driving the creation of business value.

What is Software Delivery Management?

Software Delivery Management (SDM) extends the feedback loop to encompass the entire application lifecycle, from issue creation to end users interacting with the application. Just like DevOps breaks down the walls between the development and operations teams, SDM breaks down the walls between development, product management, UIX teams, documentation teams, support and product marketers. It allows them to communicate better, understand each others' needs and ultimately make software that isn't just bug-free, but also effectively addresses the business needs and creates value for the customer.

Software developers are empowered to incorporate feedback from the support team and marketing team, using their creativity and training to find ways to solve common customer challenges or shifts in the market. Instead of being asked to continually execute on someone else's idea, developers can start with a problem statement rather than a list of feature requirements. Here's how this might work, with and without Software Delivery Management.

Lack of SDM Strategy

Let’s say the CEO at ABC Company wants an application with three features we will call Alpha, Beta and Gamma. Without SDM, this would likely be written in a standalone document as a set of complex requirements and then sent to the development team. The development team would translate the requirements into a tracking system as user stories or features, but without a complete understanding of the business drivers. The fact that the CEO’s idea comes from data pulled from customer support tickets is completely obscured.

Unfortunately, the Alpha and Beta capabilities weren’t expressed clearly in the original requirements document, and even more was lost in the translation into stories. This gap goes unnoticed. The Gamma capabilities were implemented incorrectly. The project is released to customers, but adoption is lackluster. Instead of the hoped-for reduction in support requests, support teams get a flood of requests from customers who don’t understand the user interface or are having problems getting it to work.

Meanwhile, a competitor who had the same market challenge releases a new set of features. The competitor’s new capabilities aren’t exactly the same, but they meet the same customer needs. At first, the competitor also sees weak adoption, but is able to rapidly integrate the feedback from the sales and support teams into new releases. After fixing the confusing user interface and providing appropriate documentation for the product marketing team, adoption soars. The competitor wins industry innovation awards, and ABC Company loses customers, revenue and possibly even key developers who leave to work for the competitor.

What if:

- » Developers had known at the beginning why the new features were being developed, and could suggest tweaks that deliver the same business goals? Better yet, had they understood the market need, they may have been able to suggest an alternate approach that was even better.
- » Support and development had access to common data and connected processes, so they could collaborate?
- » Iterations were done for KPIs, such as user adoption?

SDM Strategy Adopted

A company following a Software Delivery Management strategy would involve the entire software organization, and related areas, when the data from customer support was being discussed, before Alpha, Beta and Gamma capabilities were settled on. Instead of a requirements document being thrown over the wall, development and operations would be able to engage in a conversation and correctly understand the business needs the features are being designed for. Each story assigned to a developer would be linked back to a concrete business goal with the associated customer, market research and expected outcomes. The developer would have the context and insights necessary to build and deliver the right functionality, as well as the connection to end users through the support organization needed to iterate on how well the feature meets customer desires. Support could weigh in with input they have received from customers about the new functionality. Finally, the field organization could share what they have been hearing and seeing in customer and prospect accounts they have visited.

When the conversation about business goals and customer impact includes development, product management, UIX teams, doc teams, support teams, product marketers and the field organization, everyone gets a more complete picture of how a feature is or is not meeting customer desires. Together, they can determine the best way to meet the business goals creatively and efficiently. If a feature isn't working for end users, that information is surfaced quickly and immediately integrated into future iterations.

Just as DevOps isn't all about automation tools, but about fundamentally reconceptualizing the way the development and operations departments work together, Software Delivery Management is about changing the way all functions in the company think about and collaborate to deliver software. Currently, management teams at many companies are non-technical and have a poor grasp on what goes on within software engineering and, while they know that software is important in the business world, they don't have a clear idea of how software impacts business KPIs. Meanwhile, software development teams are not part of strategic discussions and often don't understand how the business functions or what the business goals are, either overall, or for a specific project. As a result, they aren't able to work as collaborative problem-solvers, using their technical skills and domain knowledge to meet business goals.

This environment doesn't recognize how essential software development is in the modern business world, or that digital transformation is not just a buzzword — it's a requirement for competing in today's market. Software development is not the digital equivalent of submitting orders to a construction contractor or an extension of requesting a new desktop monitor from IT. It is a core business activity that should sit alongside sales, marketing, finance, product design and other strategic leadership in all modern enterprises. This is true regardless of whether or not the company's core product is related to software. Today, it doesn't matter if your company sells shoes or oil, your company uses software pervasively across all departments to ultimately build critical systems that drive business activities and create powerful customer experiences. Those powerful customer experiences create loyalty and a competitive advantage for your company.

Just as DevOps isn't all about automation tools, but about fundamentally reconceptualizing the way the development and operations departments work together, Software Delivery Management is about changing the way all functions in the company think about and collaborate to deliver software.

Intelligent Iterations

Software Delivery Management not only allows for better decisions and expectations at the beginning of the software lifecycle, but also extends the feedback loop throughout the software delivery process, through actual customer success — and back. This is an endless loop of input about what's working — and maybe most importantly, what's not — to drive continuous improvement in software quality and functionality. This reframes the role of software delivery in the enterprise, recognizing it as a key business driver. An iteration is not just doing a discrete build-test-deploy process to solve a technical problem or deliver a new piece of functionality. Instead, iterations are driven by analytics like:

- Customer downloads
- Customer adoption/uptake
- Increasing/decreasing conversions
- Support requests: What types of issues is a new feature introducing? Are they technical issues or related to user experience and design?
- Performance metrics from the perspective of the customer's experience

This holistic view of the software development lifecycle brings developers, support teams, product marketers, documentation teams and product management into the same feedback loop. This makes it possible to answer questions like:

- Does this feature meet the business need it was designed to meet?
- Does this feature meet some other need and/or opportunity that wasn't anticipated?
- Are there geographic differences in adoption rates? If so, can the organization engage customer success teams to figure out why?
- Can developers use feature flagging to perform A/B tests during production and deployment to see if small tweaks will improve metrics like user adoption, conversions or number of support requests?
- Does the success or failure of a feature point to ideas for new functionality in the future?
- Does the support team have all the information it needs to answer the questions it receives?
- Do production errors point to particular holes in our testing process?

Extending the feedback loop from development to sales, marketing and customer support teams makes it possible to deliver software that doesn't just work, but meets your customers' needs and drives competitive advantage.

How Software Delivery Management Brings Value

Development and operations teams talk a lot about speed and frequency: If we can deliver hundreds or thousands of times per day, we have a cutting-edge software development process. At the end of the day, though, software delivery is not just about speed: It is about delivering the best possible product — one that meets your customers' needs — as quickly as possible. Speeding up delivery of poor quality software or software that doesn't address the market need accomplishes nothing.

While Software Delivery Management still places a value on speed, the emphasis is on value delivery across the entire software organization through application of the four pillars of SDM: Common Data, Universal Insights, Common Connected Processes and All Functions Collaborating. The table below defines each pillar.

The Four Pillars of Software Delivery Management	
Common Data	The first pillar of Software Delivery Management is Common Data. Instead of key data being locked in silos of domain-specific tools such IT Service Management (ITSM), Source Code Management (SCM) and Help Desk software, it is extracted or linked. All stakeholders in software delivery — product management, development, operations and customer success — have access to the same data. Software developers can look at customer interviews; the product organization can track and preview features; and customer support can see where a fix for a customer issue is in the pipeline. The ability for stakeholders to access common data provides the context necessary to empower stakeholders and drive informed decisions.
Universal Insights	Common Data results in the ability to achieve shared and universal insights. Information related to software delivery is cross-linked and can be used to understand how to address customer and business needs; to better communicate with customers; and to better understand and drive outcomes for maximum business impact.
Common Connected Processes	When the software organization's processes are disconnected, miscommunication, mismatched schedules, missed dates and manual reconciliation are common-place and inhibit both value and speed. Conversely, when processes such as product planning, design, customer support, and software development and delivery are connected — and driven by common data and universal insights — rapid, continuous delivery of value becomes standard. Key stakeholders work at the same cadence towards common objectives, and collaboration is frictionless. Problem statements flow into research, which feeds development and delivery, which drive documentation and customer support.
All Functions Collaborating	Building on the other three pillars of Common Data, Universal Insights and Common Connected Processes, the software organization becomes part of the larger business structure and continuous collaboration among the participants in the software delivery process becomes natural. As a result, everybody gains transparency into the process and the larger goals. Instead of working against each other, they collaborate to creatively solve problems, drive value and satisfy customers.

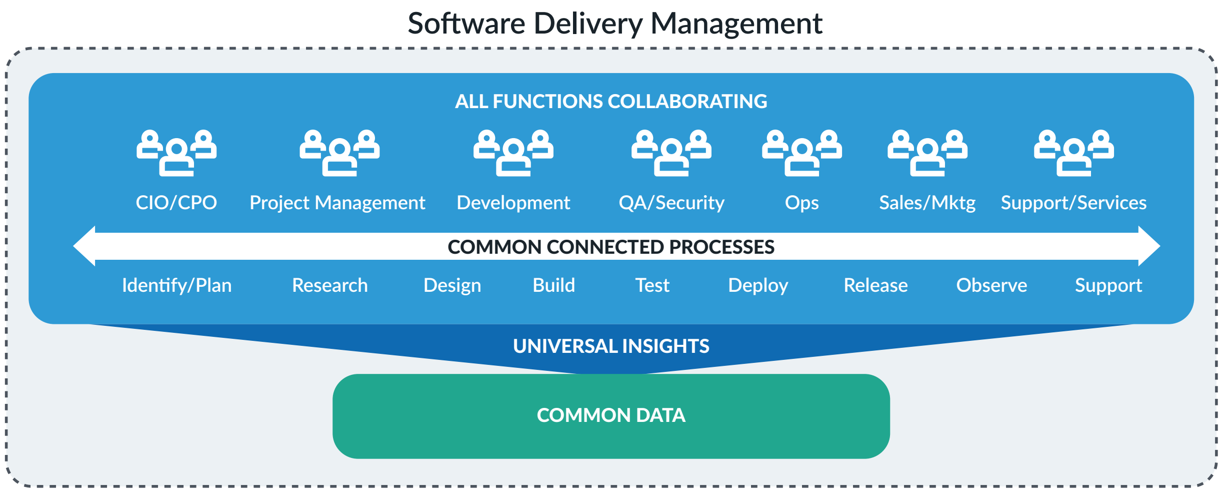
This approach still embraces and extends the automation established by continuous integration, continuous delivery and DevOps practices. The goal, however, isn't just to automate everything as a way to deliver instantaneously, but rather to free the creative humans who build software from low-value tasks and allow them to spend more time and energy on high-value, collaborative and creative work. And to give the entire organization visibility into the data required to make software drive better business outcomes.

In contrast to stereotypes about the lone, genius developer, companies get the highest value out of collaborative, cross-functional efforts. A DevOps culture is a step in the right direction, but true cross-functionality should involve collaboration among all stakeholders in the software organization, all the way through to customer success. When this happens, everyone has more time to focus on high-value work, from developers who no longer blindly create features to customer support specialists who are empowered to fix the root cause of customer frustration instead of repeatedly addressing the same concerns.

Integrated Feedback Loop

With Software Delivery Management, everyone, from the software organization through product marketing and customer success, has access to the same set of data and one unified platform for collaboration. Product marketers have a clear idea of how the feature will work, when it will be ready and which types of customers it's best for. Customer support teams have visibility into when new features will be released and can alert developers to patterns in support requests. As the feedback loop widens to include product marketing, documentation teams and customer support, developers get more valuable feedback that leads to intelligent iterations and better software. Developers also get the satisfaction of seeing how the features they build lead to concrete business results and customer adoption stories.

Everyone in the business will be operating from the same set of data, which is available in real time, in one dashboard that everyone has access to.



Source: CloudBees, Inc.

Figure 1: Common data sits at the heart for all functions to collaborate from one dashboard.

While part of the key to SDM is getting individuals with different areas of expertise to collaborate, access to common data means there is no interaction required to get visibility into the status of a software project at any time. Everyone can do their part without interrupting someone else's workflow. This allows for more seamless collaboration and communication across different departments and reduces the risk of miscommunication.

Once explained, most companies understand the value of Software Delivery Management, but very few are actually taking steps towards an integrated Software Delivery Management system. That's because of both a lack of tools available to facilitate the transition and because it requires a cultural shift potentially even greater than the transformation to DevOps.

Those companies that move towards implementing Software Delivery Management, however, will beat competitors over and over again when it comes to creating value with software. Here's why.

Software Delivery Management Changes the Game

Adopting Software Delivery Management means not sorting through customer data to deduce what the problem statement might be, then having non-developers create the requirements for a new feature to address those problems — then handing the requirement set to the software organization without ever discussing the original data.

Instead, customer support teams, business leaders and developers have access to the same shared database and same information about customer usage. They can work together to both identify the problem and brainstorm how software could be used to solve it. Everyone understands the trade-offs different potential solutions involve, and can decide collectively whether the expensive, time-consuming option is the only way to get the desired business results.

It's more likely the developers will deliver the right product the first time — and if they don't, they have enough information in the feedback loop to iterate until the software solves the customer need. Everyone, from developers to product marketers to support teams, has visibility into the process and knows when to expect new features or updates.

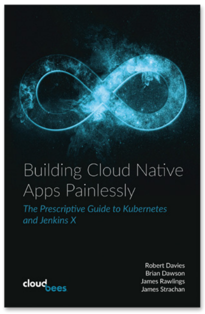
CloudBees and Software Delivery Management

Building a successful business — and maintaining and expanding that business in a sustainable way — is about coming up with novel ideas and developing products and services to meet customer demand. Software Delivery Management will help organizations reach that goal with better execution. CloudBees™ is already the leader in CI/CD and application release orchestration. With the CloudBees Suite, companies can extend the feedback loop from development into the customer support phase. CloudBees' solution for Software Delivery Management provides a way to orchestrate the entire software delivery lifecycle and tie together all the software development tools you already use, automating the repetitive parts of the process and providing a collaboration platform for the organization that is based on shared common data. This facilitates more high-value collaboration from the creative, highly-trained individuals that make up most software engineering teams, while providing transparency to other departments like marketing and customer support. Everyone in the business is on the same page and has visibility as a feature or application moves from research and ideation through development, production and deployment. The business has more insights into the process and therefore the ability to evaluate an application's success.

The end result? Software that engages customers and delivers on business KPIs.

Find out more about how you can participate in the CloudBees solution for SDM Preview Program.

Learn More


Building Cloud Native Apps Painlessly
The Prescriptive Guide to Kubernetes and Jenkins X
Robert Davies
Brian Dawson
James Rawlings
James Strachan
cloudbees

Download the eBook
Optimize DevOps with Application Release Orchestration

Read the Whitepaper
DevOps Performance: The Importance of Measuring Throughput and Stability

Read the eBook
Building Cloud Native Apps Painlessly – The Prescriptive Guide to Kubernetes and Jenkins X

Learn more
www.cloudbees.com/products

CloudBees Core is built on top of Jenkins, an independent community project. Read more about Jenkins at: www.cloudbees.com/jenkins/about

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