Cognizant

DIGITAL BUSINESS

Seven Steps to Quickly Develop Software that Delivers Customer Value

These field-tested steps can help organizations make the required changes to speed software to market that meets continuously changing market needs.

February 2018

EXECUTIVE SUMMARY

Like many of their business and IT counterparts, application development organizations are at a crossroads. To deliver the types of software offerings that are needed to hone their competitive edge, they can't continue doing business as usual. And yet, in the face of day-to-day maintenance, integration and delivery challenges, many organizations are finding it challenging to adopt the new behaviors, processes and skills required to affect needed change.

What's become clear is that traditional waterfall-style development processes are too costly and slow to keep pace with the fast-moving digital economy. Waterfall development also doesn't support the business's need to maintain a laser-like focus on meeting ever-changing customer and user desires, which they need to do in order to fill the "sweet spot" of unfilled market requirements.

Delivering software features and functionality quickly enough to meet market demands requires shifting to a DevOps model, in which software development and operations are combined to speed time to market. Software changes must also be delivered continuously, with new features delivered as minimum viable products (MVPs), and fine-tuned over time.

Faced with the need to simultaneously master new technologies, customer needs, development tools and development processes, many organizations struggle to change. A Harvey Nash/KMPG survey showed that only 18% of respondents believe they are making very effective use of digital technology to advance their business strategies.[1]

In many cases, the difficulty lies in simply knowing where to get started. In this paper, we propose seven essential steps, based on our client experience, that can help application development organizations begin making the changes needed to deliver the right software at the right speed to drive competitive advantage. By following these seven steps, they will unlock value within their legacy applications, beat competitors to market with innovative products and services, and drive user adoption, loyalty and sales.

Delivering software features and functionality quickly enough to meet market demands requires shifting to a DevOps model, in which software development and operations are combined to speed time to market. Software changes must also be delivered continuously, with new features delivered as minimum viable products (MVPs), and fine-tuned over time.

Through their continuous knowledge transfer and communication, the developers can identify more solutions to problems than would otherwise be possible on their own, and develop a real partnership that reveals the importance of applying the human touch to their job.

STEP ONE: PAIR PROGRAMMING

Businesses can kick-start a new approach to software development simply by getting individuals together to collaborate in a paired development environment.

Pair programming involves two developers working together at one machine, with one programmer (the "driver") writing code while the other reviews it and considers its strategic architectural implications. For example, if the driver is coding a log-in screen, his or her partner might make recommendations to ensure the design is scalable to support the required 100,000 concurrent users. Through their continuous knowledge transfer and communication, the developers can identify more solutions to problems than would otherwise be possible on their own, and develop a real partnership that reveals the importance of applying the human touch to their job. When the empathy that develops in a partnership is also applied to customers, it leads to software that better meets customer needs.

As team members rotate among these roles, they gain more context into how each software module affects the overall backlog, empowering them to identify issues and offer resolutions or architectural changes. In addition to this heightened sense of ownership, the team also becomes more productive, as paired partners can step in for one another when fatigue sets in, creativity is blocked or an individual is otherwise unable to work to his or her full potential.

STEP TWO: CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY

Speed to market has become essential for keeping pace with business demands and maintaining relevancy. That's why continuous integration and delivery of code has become essential. The traditional approach of delivering and integrating code every 12 to 18 months just doesn't work anymore, as customer requirements will have changed by the time the code is delivered. Instead, businesses need to deliver and integrate code four, five, six times a day – and often more – to enable rapid feedback cycles and validate that what they're building makes sense to the customer.

Application developers also need real-time visibility into whether each iteration of code meets performance, scalability and security requirements. Through immediate feedback mechanisms, they can understand, in real-time, the effects of small code iterations on their software and the customer's use of it.

Continuous delivery and test-driven development (see Step Three) enable developers to work in small batch sizes and deliver changes immediately to obtain fast customer feedback. These techniques also help ensure that any updates designed to fix bugs won't be massively disruptive or fail halfway through.

The traditional approach of delivering and integrating code every 12 to 18 months just doesn't work anymore, as customer requirements will have changed by the time the code is delivered. Instead, businesses need to deliver and integrate code four, five, six times a day – and often more – to enable rapid feedback cycles and validate that what they're building makes sense to the customer.

Too much code transforms software from your biggest competitive advantage to your biggest liability.

STEP THREE: TEST-DRIVEN DEVELOPMENT

Testing processes need to occur earlier in the software development life-cycle and be executed more frequently. No longer can quality assurance be pushed downstream, when it's most costly to fix issues. Instead, organizations should adopt a test-driven development approach, which involves first writing the test code that describes the expected behavior of the implementation, and then running that test. If the test fails, developers make changes to fix the problem. Once the test passes, the code is refactored to remove extraneous code and make it as efficient as possible. Such refactoring is essential because the more code that's written, the more code that must be maintained over time, thus increasing long-term costs. In a DevOps world of continuous integration and deployment, too much code transforms software from your biggest competitive advantage to your biggest liability.

An important benefit of test-driven development is the fast feedback it provides. Changes can be quickly made and immediately validated so that any functionality previously committed to is not interrupted. This fast feedback acts as an important safety net, keeping the business from wasting precious time and effort on flawed code that will add to its maintenance burden later.

STEP FOUR: BALANCED TEAMS

A balanced team is a self-contained DevOps group that is interconnected with but independent of other groups in the organization. Its members, who rotate among various roles, use their different but complementary skills and perspectives to help each other reach a shared goal. The DevOps group values cross-disciplinary collaboration and iterative delivery, and ensures everybody's opinion is equally important. With the confluence of opinion, the team is able to best drive the process and product forward.

Six to eight team members is the ideal size, as larger teams tend to make it more difficult to accurately predict delivery times and development costs. They also tend to reduce satisfaction and efficiency due to poor communication. The team should include an anchor, product manager (PM), engineer and designer. The anchor stays closely aligned with the product manager in order to act as the principal engineer and serve as the point of contact for developers working on the product. The product manager's vital role is understanding what software to build. Fundamental tasks of the PM center on validating the user stories that describe a needed software feature from an end-user perspective, and writing and accepting stories. The PM also determines whether to "pivot or persevere" following the feedback from a minimum viable product (described in Step Six) by assessing whether users want a feature enough for developers to keep working on it, or if they should move on to work on another feature. By keeping the team small and structured, no voice or opinion goes unheard, and no task is too big to tackle.

Weekly retrospective meetings clear the air of miscommunication, and prep the team to resume work more efficiently the following week. They also help generate real action and drive improvements week over week.

STEP FIVE: FEEDBACK

A culture of teamwork and a shared sense of empathy among team members are critical components of a DevOps team. When everyone feels a shared responsibility for the code they're writing and the product they're building, a bug is just a bug. Errors are not any one person's fault but everyone's responsibility. While this is a big change from traditional software development environments, this approach makes it easier and faster to fix bugs.

Building teamwork and trust requires delivering the right kind of feedback in the right way. In addition to eliminating finger-pointing, this requires acknowledging the time pressures and priorities faced by different team members. It's also important to encourage all team members to report problems and suggest solutions. Research conducted by Google shows that "psychological safety," where all team members feel free to speak up and take risks, is one of the five characteristics of the most successful teams.[2]

A weekly retrospective meeting helps build such trust and is a critical self-improvement tool for a team. In these meetings, teams should discuss what went well, what was "meh" and what didn't work. "Positives" might include the fact that team members who usually work remotely were in the office and able to get face time with other team members. A "negative" might be the realization that a new development tool didn't deliver the hoped-for speed improvements, and the remedial action might be to try another tool in the next week.

Weekly meetings clear the air of miscommunication or misunderstandings, and prep the team to resume work more efficiently the following week. They also help generate real action and drive improvements week over week.

STEP SIX: BUILD-MEASURE-LEARN WITH MVPS

The build-measure-learn process involves building a minimum viable product, measuring its acceptance and use, and enhancing or adapting it as needed. This approach allows developers to build code in relatively small increments (as discussed in Step Two), based on a hypothesis of the customer or user need.

An MVP is a feature that is functional enough to generate customer feedback, with the understanding that it will be tweaked and improved over time based on that feedback. By delivering the first MVP very early in the development and product definition phase, businesses can reduce wasted effort and speed results by identifying which features users don't care about, and highlighting the ones that are most important.

One rule of thumb: If you haven't been embarrassed by your first MVP, you've probably waited too long to get your initial customer feedback, and have "solved" for more than was needed in the first product iteration. The point isn't to wait until you have an elegant user interface before getting it in front of users. The important thing is to get just enough to show the user or customer to ensure no time or effort is wasted on something that isn't needed or desired.

It may also take three or four MVPs, with iterations to each, before the product is ready for full deployment.

When businesses adopt a build-measure-learn approach, they avoid the risk of rushing to market with a "release early and often" approach or spending too much time, money and effort on software that users or customers don't want.

The Build-Measure-Learn Process

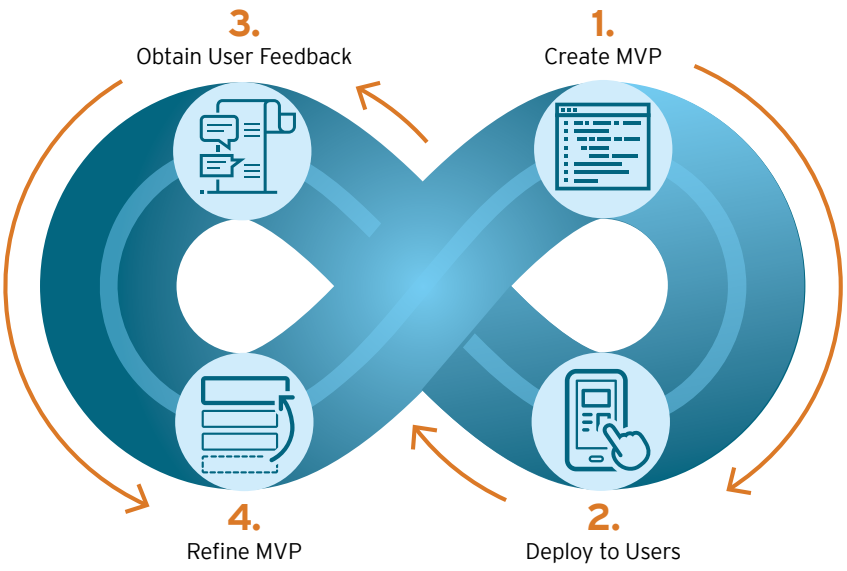

Figure 1

Idle time can also be reduced by treating any interruption (such as running a test suite) lasting more than 20 minutes as a “context switch,” which frees the developer to engage in an unrelated activity – whether that’s playing ping pong or air guitar or taking another story from the backlog.

STEP SEVEN: ELIMINATE WASTE

In addition to the competitive need for speed to market, cost is another reason that businesses need to use their development team’s time wisely. The talent gap is real when it comes to hiring and retaining skilled developers, which has effectively pushed up the cost of development.

This is why developer resources need to be focused only on what helps them meet their daily and weekly goals. One way to minimize developer distractions is to limit required meetings to three per week. The team should begin their day with the daily stand-up, capped at 15 minutes. Here, each team member tells the group what they did the day before, what they will do today and the obstacles, if any, they face, which reveals whether separate, longer meetings are required to resolve issues.

A second meeting is the daily iteration planning meeting, in which the team ensures it understands the stories designed by the PM. It’s also an opportunity for developers to understand their paired goals, and discuss how to mitigate identified roadblocks. The third is the retrospective meeting discussed in Step Five, in which the team reviews issues and plans course corrections. Outside of these three meetings, developers should be protected from any distractions.

Idle time can also be reduced by treating any interruption (such as running a test suite) lasting more than 20 minutes as a “context switch,” which frees the developer to engage in an unrelated activity – whether that’s playing ping pong or air guitar or taking another story from the backlog. By doing so, developers can refresh their energy, as the pace in a DevOps, continuous delivery and integration environment can be exhausting.

Because e-mail takes more time than face-to-face communication, organizations can also try tearing down cubical walls so all team members can easily see and talk to each other.

Making the needed changes in software development processes requires resilience and determination.

GETTING STARTED

Making the needed changes in software development processes requires resilience and determination. Even when organizations take the first steps, it may be tempting to turn back when inevitable roadblocks are encountered. For example, some developers may balk at working in paired teams with someone "looking over their shoulder." Others may find it difficult to graciously give, or accept, feedback about the quality or speed of their work. Some may resent having to learn new coding methods, development languages or ways of working. Or the fast pace of development, with instant and constant feedback, may be difficult or impossible for some developers to handle.

When (not if) such problems are encountered, businesses can't afford to give up. Instead, they should work through the challenges the same way that they iterate code: by continuously tweaking, testing, communicating and improving. By doing so, they will ensure that they stay relevant and future-ready as the world around us changes more quickly, and in more ways, than ever before.

FOOTNOTES

[1] "Navigating Uncertainty," Harvey Nash/KMPG, 2017, https://assets.kpmg.com/content/dam/kpmg/xx/pdf/2017/07/harvey-nash-kpmg-cio-survey-2017.pdf.

[2] Michael Schneider, "Google Spent 2 Years Studying 180 Teams. The Most Successful Ones Shared These 5 Traits," *Inc.*, July 19, 2017, www.inc.com/michael-schneider/google-thought-they-knew-how-to-create-the-perfect.html.

ABOUT THE AUTHOR

Brian Roche

Vice-President of Products & Strategy, Cognizant

Brian Roche is Vice-President of Products & Strategy at Cognizant Technology Solutions. In this role, he helps define and execute the company's product strategy, with a particular focus on building cloud-native applications that help customers accelerate their transition to the cloud and redefine their competitive advantage.

Brian has more than 20 years of experience in the industry. Prior to joining Cognizant, he held leadership positions at Dell EMC within the Office of the CTO, and was a Cloud Foundry Board member. He has led the adoption of DevOps with many teams and is passionate about building applications that are born in the cloud. Brian can be reached at Brian.Roche@cognizant.com I www.linkedin.com/in/brianroche.

Cognizant

ABOUT COGNIZANT DIGITAL BUSINESS

Cognizant Digital Business helps our clients imagine and build the Digital Economy. We do this by bringing together human insight, digital strategy, industry knowledge, design, and new technologies to create new experiences and launch new business models. For more information, please visit www.cognizant.com/digital or join the conversation on LinkedIn.

ABOUT COGNIZANT

Cognizant (NASDAQ-100: CTSH) is one of the world's leading professional services companies, transforming clients' business, operating and technology models for the digital era. Our unique industry-based, consultative approach helps clients envision, build and run more innovative and efficient businesses. Headquartered in the U.S., Cognizant is ranked 205 on the Fortune 500 and is consistently listed among the most admired companies in the world. Learn how Cognizant helps clients lead with digital at www.cognizant.com or follow us @Cognizant.

World Headquarters
500 Frank W. Burr Blvd.
Teaneck, NJ 07666 USA
Phone: +1 201 801 0233
Fax: +1 201 801 0243
Toll Free: +1 888 937 3277

European Headquarters
1 Kingdom Street
Paddington Central
London W2 6BD England
Phone: +44 (0) 20 7297 7600
Fax: +44 (0) 20 7121 0102

India Operations Headquarters
#5/535 Old Mahabalipuram Road
Okkiyam Pettai, Thoraipakkam
Chennai, 600 096 India
Phone: +91 (0) 44 4209 6000
Fax: +91 (0) 44 4209 6060

TL Codex 3412