Modern Software Development

It is all about quality and speed

Organizations across different sectors and industries are going through an increase in digitization efforts. The technology function often struggles to keep pace with the ever-increasing scale of new demands in today's business environment. Modern product management, engineering, and delivery discipline are needed to be able to keep up and face these challenges. To ensure quality across the value chain a 'left shift' mindset is needed.

drs. Jos van Brummelen
is a senior manager at KPMG Digital Enablement in the Netherlands.
vanbrummelen.jos@kpmg.nl

Tom Slenders MSc
is a senior consultant at KPMG Digital Enablement in the Netherlands.
slenders.tom@kpmg.nl

Organizations have massively adopted agile frameworks and methods to further integrate business and IT functions

INTRODUCTION

Organizations across different sectors and industries are going through an increase in digitization efforts. The technology function often struggles to keep pace with the ever-increasing scale of new demands in today's business environment. After all, it must keep up with market speed; address rising expectations for ease and availability of technology by the business; take advantage of new opportunities and evaluate issues of risk.

Modern product management, engineering, and delivery discipline are needed to be able to keep up and face these challenges. Organizations have massively adopted agile frameworks and methods to further integrate business and IT functions. Transformed organizations have become collaborative, cross-functional, highly automated, innovative, self-managed, and productive by adopting a set of practices such as value stream mapping, Design Thinking, Lean, Agile, and DevOps. These practices are no longer mere buzzwords; they are proven principles and mindsets that empower organizations to realize value through improved performance, profitability and market share.

When software product development is at the heart, establishing fit-for-need software development processes are key in staying ahead of the game, as well as being able to continuously deliver high-quality products, at low risk.

This article describes the way modern product-driven organizations structure and further integrate their end-to-end software delivery processes.

The ever-changing world of the IT function

The business world is under increasing demands for agility and speed. Advancements in technology have enabled the way we do business and allowed some to gain competitive advantage. However, business is still not always getting the desired speed. There are several reasons for this fault.

One reason is that consumer behavior, both from the external market as well as internal stakeholders, has changed over the past five to ten years in most industries. Customers today are accustomed to highly available, constant-release, high-value applications that are consumable anywhere, anytime and on any platform. This consumer buying behavior drives organizations to adapt their strategies, both internally and externally. The way most corporate technology is delivered is insufficient to be able to catch up with these advancements. Some reasons for this are the large amount of technical debt (legacy systems), monolithic systems that are difficult to build upon, and applying a slower waterfall approach to software development. Furthermore, long cycles of annual financial planning processes and siloed organizational functions prevent adopting a customer centric approach for product design. We also see that traditional technical solution development no longer solely belongs to the IT function within organizations.

Organizations are facing an increasing dissatisfaction with the lack of speed and flexibility in their IT functions, to the point where business consumers go elsewhere. Especially as the barrier to entry to acquire and deploy their own solutions from Cloud or SaaS providers is much lower. This availability means that businesses in organizations bypass the IT departments more often and start implementing their own decentralized IT solutions (so called shadow IT).

Finally, as the business dives into solution development outside of the central corporate IT function, organizations are beginning to understand the ripple effects of today's fast-evolving technologies and the quality impact of poorly controlled releases. These include implications to architecture, data management and privacy, access control, but also effects in areas like financial planning and supplier management. As these elements come into play, the overall risk to the enterprise grows.

To an increasing extent, it will become a balancing act between speed and risk while also putting the customer at the center of the value chain. A more holistic view of technology development is needed. This change in focus requires a solid end-to-end product development process supported by all functions of the organization.

A holistic end-to-end view of software (product) development

Regardless of which framework or method is being followed, product development – in the end – is about creating value for the end customer. Empathizing with the end user and creating a deeper understanding of the end user is not always the number one priority however when it comes to developing new software products.

For the last few decades, the focus around new IT developments were primarily on specialization and improving operational excellence. This scenario is where the traditional waterfall processes excelled: how – from an IT perspective – to design the best possible architecture and realize the most optimal design. But also – given the agreed upon design – how to make the actual software development process as efficient as possible for the developers and testers involved. Only after development was done, the user came into play again to test the actual result. The service support and maintenance teams came last in line. These teams typically had a completely different focus: keeping the running environments as stable as possible while keeping costs low. Most of the time, due to budget cuts, they needed to keep things up and running at a lower cost every year.

Even though concepts like Lean Product Development and Design Thinking had existed for years already, these approaches were mostly used in the very early stages of the design process; well before the actual software design phase started.

With the establishment of the Agile Manifesto [Fowloo] in 2001, the product development mindset started slowly changing to an incremental approach. From this period onwards, an agile approach (like Scrum) has become the new normal. This first wave of agile software development practices was mainly focused on the internal alignment between business team members and IT functions. As a result, software development was more focused on creating Minimum Viable Products (MVP's) in short iterations, leading to a value thinking mindset and a shorter time to market.

When multiple teams needed to be involved for creating a releasable end product, the Agile scaling frameworks (like the Scaled Agile Framework, Large Scale Scrum or the Spotify culture) gained popularity, also focusing more on the so-called Lean-Agile mindset. All these frameworks put more emphasis on creating actual product-driven organizations, whereby the identification of the value streams in the organization (a concept that Lean already introduced in the early days) became a main driver for the new target operating model of organizations ([Cool18]).

At the same time, all these models had not yet fully fulfilled the promise of maximizing the efficiency of a project. Further integration across development and operations teams was still a hurdle to overcome. This led to the creation of a philosophy called DevOps. The main goal of DevOps is to unite software development with infrastructure operations into a single operating model through collaboration and communication, emphasizing frequent and fast software deployments. Four main aspects of DevOps that are at the heart of this philosophy are collaboration, automation, measurement, and monitoring.

DevOps is often used as an umbrella term to describe software development approaches with the aim of increasing the pace of software development processes

> Organizations are facing an increasing dissatisfaction with the lack of speed and flexibility in their IT functions

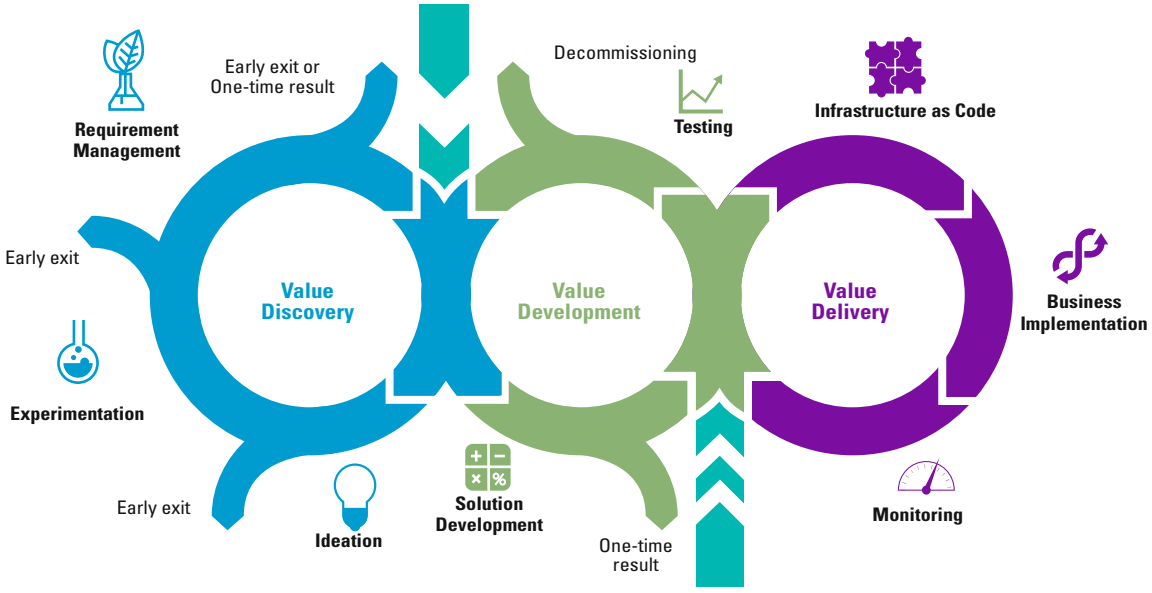

Figure 1. A typical modern delivery approach for software development.

and improving software quality. DevOps has been referred to as many different things among which a movement, a philosophy, a (development) practice, a mindset or a culture. Seen from a development practice perspective, the focus was always primarily on IT-related matters. It was about breaking down the traditional silos of IT agility by integrating engineering, testing, and operations into full stack teams, automating large portions of the value chain, and creating a culture of collaboration focused on customer outcomes.

More holistically, DevOps can be described as a movement with the goal of becoming "better" and "faster". It brings the word "continuous" to the center stage of agile development and is, in fact, adopting a continuous everything model that, in a way, is fixing the promises that agile (frameworks) once made about delivering customer value faster.

While many frameworks for agile software development exist, these frameworks do not cover the full extent of agile and DevOps combined. For many organizations, it is the successful combination of these approaches that will in the end lead to success. In the next paragraphs, we describe a typical modern software delivery approach to be able to speed up the delivery process, also ensuring the delivery of continuous high-quality products.

Better, faster and more transparent: a typical modern delivery approach for software development

Modern delivery approaches for software development focus on the entire value chain, combining a mixture of Design Thinking, Lean, Agile and DevOps practices. The combination of these concepts provides a full scope of the project; sometimes referred to as "from concept to cash": from initial idea up until gathering data about the end product in a live environment, whereby development is done in short iterations and product releases are done frequently. For some organizations, this means delivery every few weeks, but for an increasing number of companies this means releasing new functionalities to the end clients, multiple times a day.

In Figure 1, a typical approach to modern software development is shown. The actual solution development can be segregated into three different iterations that are intertwined. During *Discovery*, the need of the customer is identified. These needs are then implemented in the *Development* iteration, after which during *Delivery* the value is actually delivered to the client. All three phases consist of short iterations to keep the feedback cycles as short as possible to ensure the value delivered still matches the actual need of the client. Dependent on the ambition level, this flow of activities can take a couple of weeks, but in certain occasions only a couple of days.

During **Discovery**, the focus will be on the identification of the actual need. Then, the short- and longer-term

roadmap is identified by selecting and prioritizing the right ideas and client wishes. In the early stages of product development, aspects like ideation and experimentation play a major role, empathizing with the (end) customer and collaborating and testing new ideas and insights. Design Thinking and rapid prototyping techniques are most of the time valuable sources for inspiration. Sometimes, as part of the architectural runway, small technical experiments will be executed as well to research technical feasibility of the possible solutions identified.

When the product is already in a more mature state, the focus of this stage will be more about identifying smaller, incremental improvements that can be taken from the actual use of the product. Collecting data about the actual use of features (see *Delivery*) can be a very valuable source for this.

During **Development**, the focus will be on the actual development and testing of the proposed functionalities. As organizations strive to deliver more frequent, this requires a high level of automation of the development pipeline to be able to achieve both speed as well as ensuring quality. Therefore, development teams are constantly working on optimizing their development factories to enable automation. To ensure speedy development, making use of Continuous Integration practices are key. Build automation ensures that source code created by multiple developers is integrated correctly. Test automation in these pipelines is the only solution that can provide the risk mitigations that are needed to deliver new functionality at speed. To ensure that security standards are met, many case aspects like security testing are also highly automated.

Value Delivery

The purpose of **Delivery** is to actually deliver the result to the end user. In doing so, deployments need to be automated as well to increase their speed and quality. A straightforward and repeatable deployment process is important for what is called Continuous Delivery. This means that infrastructure changes are considered part of the delivery process. This concept is often referred to as Infrastructure as Code (IaC), the management of infrastructure (networks, virtual machines, load balancers, and connection topology) in a descriptive model, using the same versioning that a DevOps team uses for source code. This consistency is not only beneficial for speed, but also provides more reliable deployments and thereby also improving compliance.

After software has been deployed, measurements are executed in the live environment to proactively monitor the application performance as well as to gather information about the actual usage by the end users. All this information can be used to detect issues as early as possible, or can be fed back to the Product Owners to better adapt the product to the needs of the users.

By integrating significant parts of the development process and by creating small cross-functional teams, most organizations today are better capable of delivering business value much faster. Also, as a result of the smaller deliveries, the risk of failure is reduced as the number of different components affected is generally much smaller. Because there is also a lot of focus on rollback scenarios in case of any mistakes or errors during development, the smaller deployment size also has a positive effect on the time for recovery. The combination of these results in lower risk of releases and promotes courage within teams to experiment more often. This in turn empowers the *Discovery* phase which creates a positive continuous feedback loop.

To be able to come to a good end-to-end process, it is key to have the right foundations. On the one hand, this will require a well governed and agreed upon process that allows room for continuous improvement and continuous learning. On the other hand, aspects like culture and attracting and developing the right talent are indispensable, a prerequisite for success. Figure 2 outlines all the detailed elements of this lifecycle.

Ensuring quality across the value delivery chain

As discussed in the previous section, there are a lot of different aspects to keep in mind that can influence the quality of the actual output. Keeping the goals of becoming "better" and "faster" in mind, there are a couple of aspects that, from a software quality perspective, are relevant:

- In an agile world where development is done iteratively, it becomes a necessity to automate significant parts of the workflow to be able to deliver iteratively

> An important aspect in any organization is to detect possible issues in the software development lifecycle as early as possible

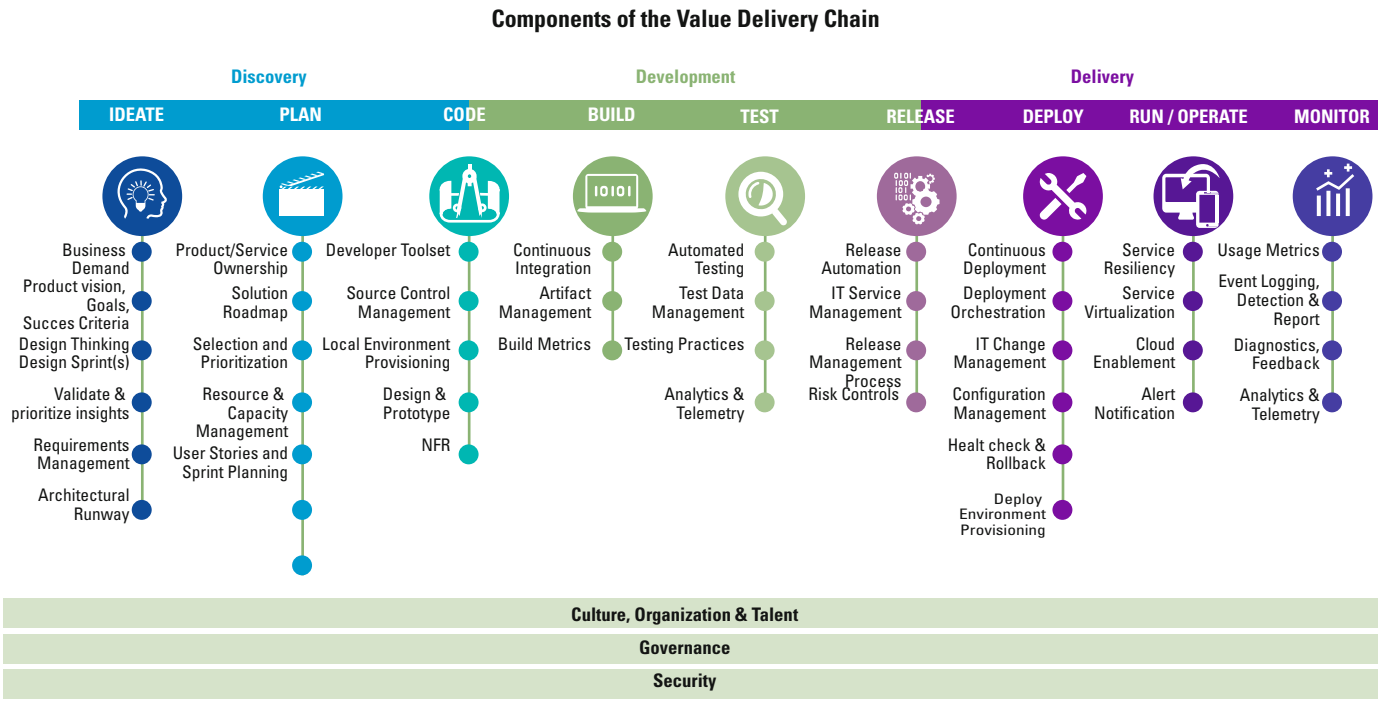

Figure 2. Typical elements of the lifecycle.

and this importance intensifies when the release frequency is increased.

- Ensuring quality means taking an end-to-end view of the entire software development lifecycle. Starting from gathering the initial requirements to learning about the actual product that is running in a live environment. This quality focus also means considering that security, audit, and compliance processes need to be an integral part of the delivery cycle.
- Learning from past experiences and making optimal use of the feedback loops of inspect and adapt (sometimes referred to as the Plan, Do, Check, Act cycle of Deming) becomes an important measure to continuously emphasize on not only process related matters, but also puts focus on inspection and adaptation of the actual end product. This requires a culture of shared responsibilities and ownership (sometimes referred to as a "we build it, we run it" mentality).

Adopting a 'left shift' mindset for ensuring better quality at speed

An important aspect in any organization is to detect possible issues as early as possible in the software development lifecycle. This aspect is sometimes referred to as a 'shift to the left' approach. The 'shifting to the left' metaphor dates to the 1990s, as software development organizations realized that the traditional/waterfall approach to software development often led to poor quality software and expensive fixes. An important root cause of these issues was that the actual testing was done far 'to the right', so very late in the life cycle. Especially for testing integration elements as soon as possible this mindset provided value. By moving things to the left, meaning to start testing as early as practical, software quality has many times improved significantly.

As an initial concept taken from the testing world, the shift left approach became a must do in any agile software development project. The shift left approach has been embraced further and became a major driver of integrating operations into the life cycle. Only by automating big parts of the operational activities, real speed can be achieved, and quality levels can be guaranteed. By integrating development and operations even further, the development teams were able to release more frequently.

From our point of view, the shift left mindset can be taken much further, already in the early stages of design. Many industry studies show what happens when a team does not fully understand the business needs and the associated requirements. In fact, some estimates show that more than 35% of software projects fail due to poor requirements or the lack of proper user involvement ([Huss16]).

Bringing integrated approaches like Design Thinking into the overall process means that not only controlling the quality of the actual work that is delivered can be included into the value delivery chain; "garbage" is also not allowed in. Starting off with a good ideation and prototyping phase provides more certainty than the development of the right functionality. The risk of human

error can be reduced greatly by automating the work to be done, including release and deployment of software products to the infrastructure environment.

Similarly, security can be considered part of the shift left mindset, which is sometimes called "DevSecOps". Now, security is considered a shared responsibility and should therefore be part of the skills of a DevOps team. Security aspects are part of the requirements, which leads to a system which is "secure-by-design". Lastly, security should be measured in an automated fashion: from a developer writing his code, which is being scanned for vulnerabilities, to production being scanned for attacks. This way, security is incorporated throughout the whole development lifecycle instead of the end, where it traditionally resided.

By taking a system view (looking end to end), integrating significant parts of the workflow and measuring the actual outcomes, software quality can be increased significantly.

However, this shift is not only about tooling and processes. They have very limited value unless they are paired with a culture which stimulates and fosters collaboration and continuous improvement, and which is focused on a shared responsibility for all different aspects within the project development cycle. Product-focused organizations can become more successful when the focus is not just on a single development team, but aligned at all levels of the organization.

Making the actual shift to deliver higher quality software products at a much higher speed is not something that will happen overnight. It requires a culture where constant focus is put on experimentation, learning, and adaptation to the actual needs.

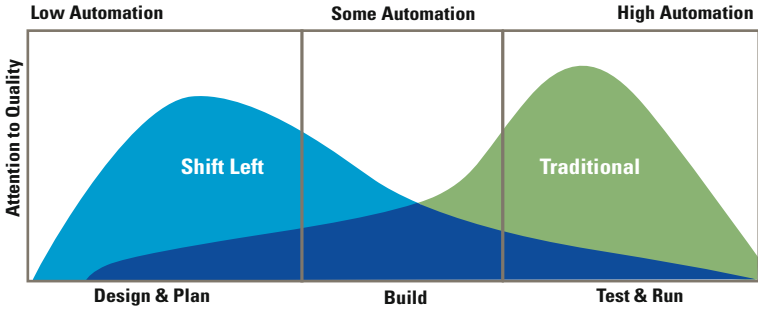

Figure 3. The 'left shift' mindset.

CONCLUSION

Forward looking organizations understand that Agile & DevOps are about shifting the value stream closer to the business. As discussed in this article, the value delivery stream needs to be considered as a whole to be able to fulfill the promise of delivering good quality software products at a constant high pace.

The shift left mindset to quality focuses on solving issues as early as possible in the software/product development lifecycle. Taking a left shift approach to quality not only means fully automating the delivery pipeline, but also already starting in the early phases before any code is written to validate concepts, hypothesis, and prototypes with future end users. Integrating these aspects into the value delivery chain, as well as paying continuous attention to all quality aspects during development and operations, is key to delivering good quality software products.

Integrating these elements into the overall process can be a main driver for improved quality, risk reductions, and faster speed to market. Furthermore, it can even improve quality of life for employees. By putting efforts in both automation and process improvements, the business can focus on speed, flexibility and quality at scale.

References

[Cool18] Just Coolen, Tim de Koning, William Koot and Victor Bos, *Agile transformation of the (IT) Operating Model – Cross-industry observations and lessons learned*, Compact 2018/1, https://www.compact.nl/articles/agile-transformation-of-the-it-operating-model/, 2018.

[Fowl00] Martin Fowler and Jim Highsmith, *Manifesto for Agile Software Development*, The Agile Manifesto, https://www.agilemanifesto.org/, 2000.

[Huss16] Azham Hussain, Fazillah Kamal and Emmanuel Mkpojiogu, *The Role of Requirements in the Success or Failure of Software Projects*, ResearchGate, https://www.researchgate.net/publication/308972993_The_Role_of_Requirements_in_the_Success_or_Failure_of_Software_Projects, 2016.

About the authors

drs. Jos van Brummelen works as a senior manager for KPMG Digital. He is specialized in the implementation of (agile) software development frameworks and methods.

Tom Slenders MSc works as a senior consultant at KPMG Digital. He has multiple years of experience as a software developer and specifically focuses on advising clients about their software delivery methods and frameworks.