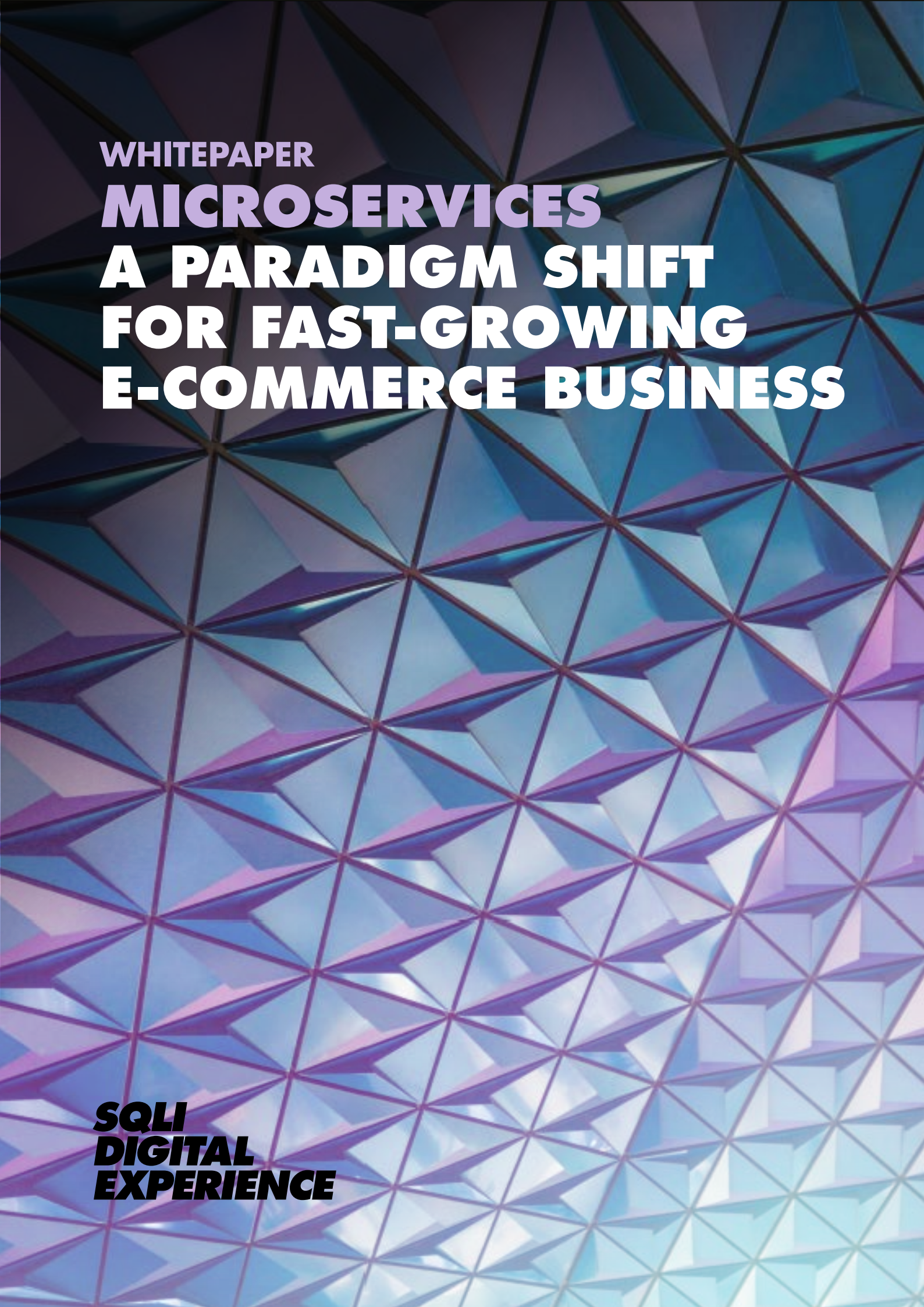

WHITEPAPER
MICROSERVICES
A PARADIGM SHIFT
FOR FAST-GROWING
E-COMMERCE BUSINESS
SQLI
DIGITAL
EXPERIENCE

MANAGEMENT SUMMARY

Speed is today's leading competitive differentiator.

Facing changing customer demands, the rise of the mobile web and ever shorter innovation cycles, merchants need to make sure to have both an organizational as well as a technical structure that allows for agility and speed. While monolithic e-commerce applications still drive most of the biggest retailers' sites worldwide, they have also become bottlenecks for innovation.

Companies such as Netflix, Amazon, Uber, eBay, SoundCloud and Google already rely on an architectural principle called the evolutionary architecture. This architecture supports incremental, guided change

as a first principle across multiple dimensions. Perfectly suited for the evolutionary architecture are microservices: small and independent micro applications that address specific business functionality. Microservice are for software like a car factory is for a car manufacturer; instead of one big car-making machine there are many small machines that create car parts and assembly.

Compared to the monolithic approach where one application containing all business logic, a flexible network of microservices handles all complexity. Developers can work on small functional chunks rather than having to understand many millions of lines of code, resulting in better quality as well easier testing and adapting.

Using examples by Netflix, Gilte Group and Zalando, we will show how brands benefit from switching to microservices and give practical guidance on the evaluation and introduction of this methodology.

Mark Blockhuys CTO at SQLI

CONTENTS

WHY MICROSERVICES?

CENTRALIZED TEAMS AND MONOLITHS ARE EVERYWHERE

Ever wondered how Netflix can stream high-definition movies and shows over what seems like limited bandwidth? It wasn't always this perfect. In 2008, a single missing semicolon corrupted a major database and brought down the entire website for hours. This incident1 kick-started its move from monolith to the cloud. The company needed an architecture that would allow Netflix to be available 24/7, fast and scalable. Netflix's approach was to break down services into smaller services. That would result in more managing, but would also prevent losing the whole system all at once. Netflix's new architecture paved the way for their expansion of services to over now 190 countries.

In the digital age, fast-growing and fast-moving organizations need to find both an organizational and a technological structure that allows for agility and speed. The challenge is to anticipate change, especially for the IT department.

SYSTEMS FOLLOW ORGANIZATION

When it comes to team set-up, e-commerce IT departments traditionally follow a centralized structure with experts organized around technology tiers. So in practice, a database professional might work on checkout in one week, and on product search in another – both two completely separate parts of the application. Such a jack-of-all-trades approach usually does not allow deep and specialized knowledge to evolve and often gives way to mediocrity. And when a project requires the involvement of experts across several technology tiers, this results in massive communications overhead. This causation is described by the principle of Conway's2 Law, as dubbed by Fred Brooks:

> **"Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations"**
>
> – Melvin Conway

WHY E-COMMERCE APPLICATIONS ARE MONOLITHS

Conway's Law implicates that a centralized team structure will result in a centralized application. If there's an invoicing department for example, there will be an invoicing system. So when the first e-commerce systems were launched at the turn of the century, they inherited the monolith paradigm of contemporary software, such as ERP solutions. E-commerce followed the zeitgeist and reflected the technological status quo of the time. Most people accessed the Internet through CRT monitors and the mobile web seemed light-years away. Also, the capability to connect systems via standardized APIs – something we take for granted today – was not available as yet. As a result, infrastructure components had to be

created from scratch and added to the core of the commerce solutions – where they still are today as considerable technical debts. Successively, e-commerce software developed into monolithic, full-stack suites, assuming some ERP functionality as well as other parts of the business logic such as WCMS or CRM.

ARCHITECTURE FIRST

So, what did Netflix do to curb Conway's Law when it moved to the cloud? It embraced architecture before organization. Netflix created 30+ separate engineering teams capable of independently developing and implementing new services on a variety of new platforms. This allowed the company to build and test new services with a relatively low- risk, through which Netflix became more adaptable to change.

ADAPTING TO CHANGE

As technology and consumer behavior keep evolving, so does your business focus. Especially in recent years, global commerce has grown exponentially and there is no sign of this development ending anytime soon. Customers worldwide are also becoming more demanding, looking for inspiring shopping experiences on all channels and on all devices. Personalization, high service levels and fast delivery are just a few of the challenges merchants are currently facing. The question always seems to be what will be next. How can organizations anticipate these changes successfully?

Some things may change by an organization taking the initiative, other things will change involuntarily. There are two kinds of change your organization has to deal with:

BUSINESS-DRIVEN CHANGE

This covers new revenue models, disruptive competitors, emerging channels, changing customer needs, market regulation and product innovation. These change the business requirements and use cases you are addressing with your architecture.

ECOSYSTEM CHANGE

The technical domain shifts and evolves as programming languages, libraries, framework tools and operating environments change. Docker, for example, helped to bring container technology to the masses. Consequently, this revolutionized the way we use computer resources.

Change can have a very strong impact on your architecture because of the so-called **Dynamic Equilibrium**. A term typically used in chemistry and other physical sciences, it also holds true in software architecture. Dynamic Equilibrium means that when you change a finely-balanced condition, you'll require continuous adjustments to maintain a stable state. Just like Yin and Yang, the software universe is dynamic rather than static. In a constant state of flux, any snapshot of your software architecture will differ depending on the moment it's taken.

WHY MONOLITHIC APPLICATIONS SLOW DOWN INNOVATION

In this fast-moving, fast-growing market environment, the shortcomings of monolithic applications and an outdated software architecture become painfully apparent, especially to ambitious merchants around the globe. As monoliths continue to grow, there is a high degree of software complexity that brings the risk of turning into a black-box that nobody wants to take responsibility for. Teams are usually structured according to their individual functions, such as frontend, backend or database, which results in any project taking a great deal of time. And as the amount of transactions grow every year, scaling efficiently becomes a big challenge.

So, quite understandably, companies facing the need to move faster and be more innovative are trying to find ways around these restrictions. Rebecca Parsons, Patrick Kua and Neal Ford from Thoughtworks3 came up with a fitting concept.

EVOLUTIONARY ARCHITECTURE: THE ANSWER TO CHANGE

By its very nature, architecture is what you want to change the least since architecture is the hardest and most costly to change. That's why changes

in the ecosystem can create real problems for software architecture, especially enterprise. The more things change, the less possible it becomes to do long-term planning because predictability gets lost. How can you have a five-year plan for your architecture when it can be easily overthrown by some unfathomable innovation? The answer: to build an architecture with change and scalability foremost in mind, called the evolutionary architecture. The guiding principle is to support guided, incremental, non-breaking change along multiple dimensions. Microservices fit this bill perfectly.

WHAT ARE MICROSERVICES

MICROSERVICES: SPECIALIZED, LOOSELY COUPLED SERVICES

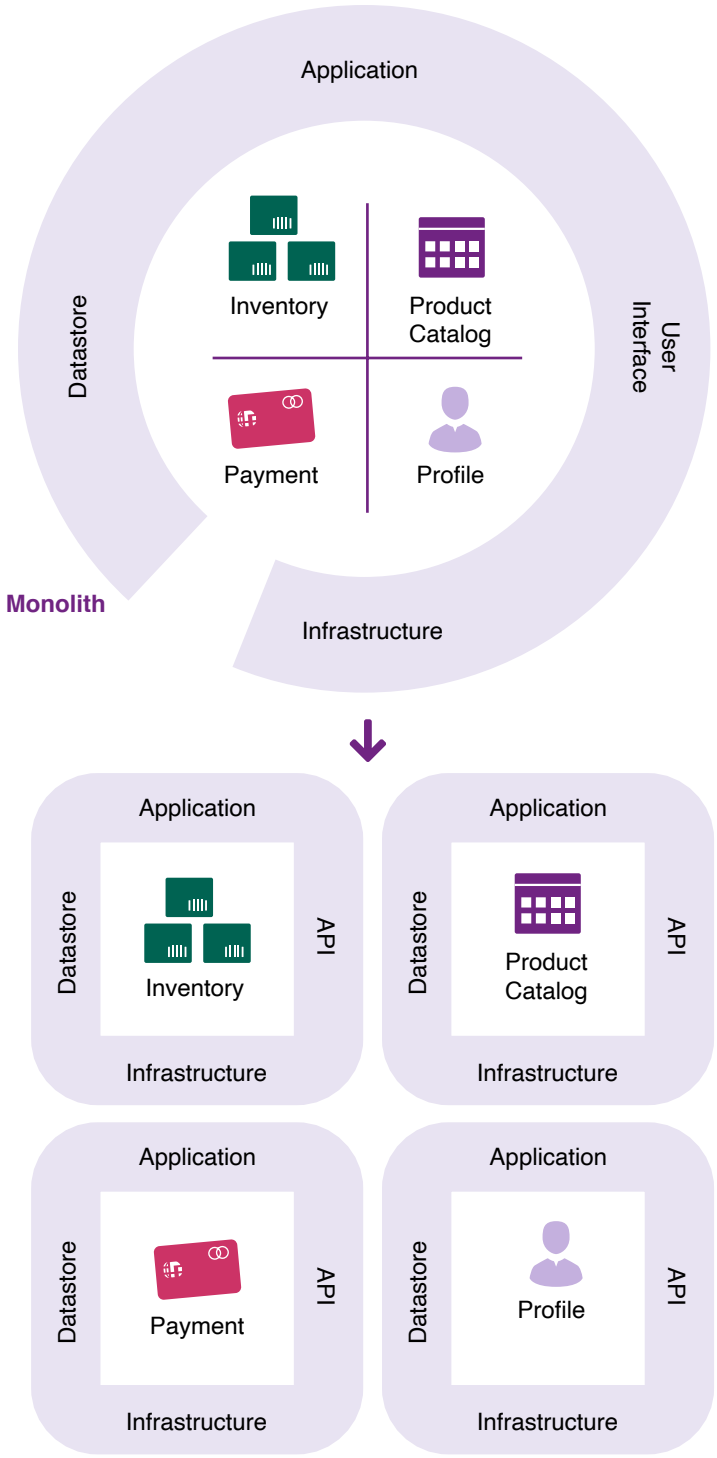
Microservices are often described as the enablers of the e-commerce of tomorrow. Encapsulating each business capability into individual services that interact, microservices offer specialization, speed and reliability.That allows for rapid anticipation to customer needs, which is exactly what e-commerce today demands. But the microservices architecture is no silver bullet: it comes with a trade-off.

WHAT ARE MICROSERVICES?

The microservice-based approach encapsulates each business capability into individual services that interact with each other. Microservices are small, standalone applications that are designed, deployed and managed individually like Lego bricks. This means you can use the language most efficient for each service. You can add specific functionality like a shopping cart, payment and search functions. Working independently of each other and loosely coupled, each microservice can be replaced individually without impacting the rest of the application.

“There is very little centralized management. Because microservices are independently deployable, we can start to break the dependence on technology stacks as well. We are no longer tied to technology stacks.”

– Rebecca parsons



As Rebecca Parsons, CTO at Thoughtworks, explains in this video4, microservices come with **smart endpoints and dumb pipes**. As a design principle, this approach favors basic, time-tested asynchronous communication mechanism (APIs) over complex integration platforms. APIs allow the focus to be put on the microservices by dedicated teams. Product owners, marketers and process managers manage their own piece. Thus, the workload shifts from IT administrators to end-users. After initial development and implementation, all IT has to do is to keep things running smoothly.

4 https://www.thoughtworks.com/

THE 6 BENEFITS OF MICROSERVICES FOR E-COMMERCE

1. REDUCED SOFTWARE COMPLEXITY

The scope of a single microservice's functionality is limited. That makes maintaining and updating so much easier. You only have to care about messages from other microservices that you subscribe to (inputs) and your API that can be called (outputs). For example, at Gilt Groupe, the first microservice that was put in production was designed to only output a small marketing message on a per- user basis

2. SPECIALIZATION

Microservices allow for choosing the right tool for the right task. Each microservice can use its own language, framework or ancillary services best suited and preferred by the team using it. For services requiring extensive background calculation, developers might choose to use Java for example. Whereas others might rely on more lightweight technologies such as PHP or Ruby.

3. DECENTRALIZED RESPONSIBILITY

Dedicated teams take full responsibility for "their" microservice. This leads to smaller codebases, which help developers to focus and have a closer connection with end-users. This leads to better motivation and more clarity. As Amazon's CTO Werner Vogels5 famously put it: "You build it, you run it". And so you own it.

4. FASTER TIME TO MARKET

Building a microservice requires a cross-functional development team that works independently on their project. This reduces synchronization efforts between teams and allows significantly faster deployment. For example, the product service team can work on its own and makes adjustments and deployments as it sees fit – independently of what the payment team is currently working on. As a result, it also becomes easier to test individual services and establish a continuous delivery workflow.

5. INCREASED RESILIENCE

A business application made of an array of microservices has no single point of failure. If one service no longer responds, this does not automatically break the whole application. For example, you can continue streaming movies on Netflix even if the search is down.

6. HIGHLY SCALABLE

Microservices are small and work independently. This means it's easier to scale them vertically and increase the overall performance of the whole business application. Without having to scale the entire application, you can scale up a single function or service. You can deploy business-critical services on multiple servers for increased availability and performance. If, for example, an application requires a lot of calls to an inventory service, this service can be scaled individually – without wasting resources on scaling other parts of the application that receive much less traffic.

5 https://queue.acm.org/detail.cfm?id=1142065

MICROSERVICES ARE NO SILVER BULLET

While microservices does address a lot of monolith issues, it doesn't result in development and maintenance complexity magically vanishing. Instead of a classic, horizontal structure, microservices require a cross-functional structure with vertical teams that work independently. Organizations will need to have the right infrastructure and tools in place to orchestrate and monitor their microservices architecture. The following seven challenges come with choosing for microservices:

1. DECENTRALIZED DATA

Each microservice has its own data store and governance. Multiple databases and transaction management require extra attention to prevent losing overview of all the deployed microservices. This happened to Netflix, which had to adapt to the DevOps model to manage hundreds of microservices.

2. TESTING

When testing a microservices- based application, you'll need to confirm and test each dependent service. This makes integration testing and end-to-end testing more difficult and also more important, since one failure can cause something a few hops away as well.

3. DEPLOYMENT: ESPECIALLY

In the initial set up, deployment requires attention. You'll need to think about how services are rolled out and in what order. Investment in automation will be necessary.

4. MONITORING

A centralized view of the whole system is critical to pinpoint bugs. At scale, monitoring microservices quickly becomes a challenge as remote bugging is not an option across dozens or hundreds of services. That's why logging, containerization and visualization are best considered from the very beginning of an application's lifecycle.

5. CROSS-CUTTING CONCERNS

Each microservices has to deal with a number of cross- cutting concerns like logging, metrics, health checks, externalized configuration etc. It's likely you didn't anticipate these fully at design time. You could accept the overhead of separate modules for each concern. Or, alternatively, encapsulate these concerns in another service layer through which all traffic gets routed.

6. HIGHER COSTS

Microservices come with a higher workload for operations, deployment and monitoring. This comes with higher operational overhead. As services communicate with each other, the resulting high number of remote calls can lead to higher costs associated with network latency and processing. And with each microservice deployed independently and requiring its own runtime environment and CPU, there is also a higher resource demand.

And with microservices relying heavily on messaging, certain new challenges will arise. Without using automation and advanced methodologies such as Agile, communication can be hard. This calls for DevOps tools such as CI/CD servers, configuration management platforms APM tools for network management. You need a common container orchestration system, multiple tiers of load balancing, and service discovery to make sure that the services are deployed correctly and to monitor whether they work together as planned. Companies already using these tools will find starting with microservices is easy. If these extra requirements need to be adopted however, this can be a challenge for smaller organizations. But neither of these issues should be leading your architectural approach: the focus should be on the business first.

WHEN DO YOU CHOOSE MICROSERVICES

STARTING WITH MICROSERVICES

With advocates like Netflix, Airbnb, Google, Disney and Amazon, microservices have built momentum and mind share. With the benefits and challenges of the microservices architecture clear, the next question is the matter of when.

It should be clear by now that there is no guaranteed best choice when comparing monoliths and microservices; opting for one of the two always involves a trade-off. Your own business context – evaluated against the pros and cons – is essential to deciding if you should start with monolith or microservices. The specific needs of your organization should be leading in your architectural approach. That will help you to clarify and cut down on any unnecessary complexity. When starting with microservices, you should clearly understand why you want to do microservices.

SCENARIOS FOR MICROSERVICES

For a complex, evolving application with clear domains, the microservices is a logical choice. Especially for the development of complex e-commerce projects, microservices offer a clear advantage over monoliths, since reduced functionality leads to less software complexity which in turn makes developing new features easier. Typical starting points for the microservices architecture are:

CLEAR DOMAINS

During the designing phase of the microservices, their boundaries should be clear so can align them with the business capabilities. In Domain-Driven-Design, this is also known as the bounded context. This allows you to use the language most efficient for each domain.

EXTREME REQUIREMENTS

Microservices can deliver maximal agility, speed and scalability. Applications with extreme requirements are best served with a microservices architecture that can cater specifically to each service.

EXPERTISE IN MICROSERVICES

To start with microservices, you'll need a team with some experience in microservices or distributed design. DevOps and containers experts are advised since these concepts are tightly coupled with microservices.

COMPLEXITY

If you plan a large application with multiple modules and user journeys that lead to high complexity, a microservices architecture is the best fit. It makes scaling and adding new capabilities much easier.

AMAZON'S MOVE TO MICROSERVICES: THE API MANDATE

Back in 2002, Jeff Bezos, CEO of Amazon issued a mandate that would have enormous repercussions for the company's future: he requested that all teams be required to expose their data and functionality through service interfaces and only communicate through these. Also, these services had to be set up in a way that they could be used by external developers. This mandate has since been known as the API Mandate6. As a result, the company built an infrastructure that would allow them to scale their online retail business and provide a hugely successful platform to third-party developers and businesses.

WHO BENEFITS FROM MICROSERVICES?

HOW NETFLIX MOVES FASTER THAN ITS COMPETITION

Availability wasn't the only reason for Netflix's move to the cloud in 2009. The company was also branching out to platforms like Xbox, Wii, mobile, and was expanding globally with the number of users growing very rapidly. Netflix couldn't build data centers fast enough to keep up. To meet the increased demand for services and fulfil its potential, scalability was key.

"When we said we were going to move all of Netflix to cloud everyone said we were completely crazy," Netflix cloud architect Adrian Cockcroft7 shared at a GOTO conference. But by December 2011, Netflix had successfully moved to the cloud, breaking up their monolith into hundreds of fine- grained microservices.

The microservices architecture allowed Netflix to greatly speed up development and deployment of its platform and services. Netflix became one of the first big corporations to exist completely in the public cloud. And to prove that its all-in public cloud strategy is not only for unicorns like Netflix, the company open sourced many of the tools, technology and components used through the Netflix Open Source Software Center (Netflix OSS8). This platform provides an accessible on-ramp for applications to move to a cloud native architecture.

"Gilt Groupe relies on microservices for their unique business model"

The Gilt Groupe offers a flash-sales e-commerce site for luxury brands and lifestyle goods and was founded in 2007. Based in New York, the website aims at selling highly discounted merchandise to its members with an estimated global net sales of $580 million in 2018. Each day at noon, Gilt sends

6 https://medium.com/slingr/what-year-did-bezos-issue-the-api-mandate-at-amazon-57f546994ca2
7 https://www.youtube.com/watch?v=BeNrVI2_nyI
8 https://netflix.github.io/

out a blast of emails and other notifications to its more than 6 million members, informing them of discounted products with only limited stock. As a result, they see a massive traffic spike following these send-outs, resulting in a stampede of visitors to their site. From a technical point of view, one of the major goals for Gilt was to find a software or an architecture that would be able to handle the shift of this traffic dynamic.

After relying on software monoliths based on Ruby on Rails and Java for the first years, Gilt decided to move towards a microservice-based architecture in 2015. Today, they are running more than 250 different services that create the individual pages for their different sites and form the basis for their inventory and logistics.

The Gilt Groupe reported9 the following benefits to this approach: lessens dependencies between teams resulting in faster code to production, allows lots of initiatives to run in parallel, supports multiple technologies, languages and frameworks, enables graceful degradation of service and promotes ease of innovation through 'disposable code'.

"Zalando builds a fashion platform on top of microservices"

Founded in Germany in 2008, fashion retailer Zalando is now active in 17 countries, has generated revenues of more than € 5.4 billion in 2018 and sees on average more than 300 million page visits per month. In 2014 the company has started an initiative dubbed **radical agility**: changing their organizational structure to form small, autonomous teams and create software using cloud and microservice principles. One of the first projects following the radical agility approach was to exchange their fashion store frontend, as Software Engineer Dan Parsa explains10: "Our team — seven engineers and a UX/UI designer — decided to replace 'Jimmy', our monolithic shop application, with microservices built by multiple autonomous teams. Under Jimmy, all of the Shop teams shared the same code base, lacked true ownership or the freedom to make decisions, and couldn't move quickly because of Jimmy's slow startup time and our overly complicated deployment processes. Radical Agility promotes microservices to get around monolith-generated problems."

This strategy allowed the Zalando development teams to bring new features into production without depending on others. Although there still needs to be communication between the teams in order to create a seamless frontend design and a homogeneous user experience, relying on microservices has increased the level of innovation at Zalando. And since microservice are polyglot by nature – they can be built with any given technology – the fashion retailer is able to attract new talent easily.

9 https://www.infoq.com/news/2015/04/scaling-microservices-gilt/
10 https://jobs.zalando.com/tech/blog/from-jimmy-to-microservices-rebuilding-zalandos-fashion-store/?gh_src=4n3gxh1

HOW DO YOU START WITH MICROSERVICES

WHAT STEPS ARE REQUIRED TO INTRODUCE A MICROSERVICES ARCHITECTURE?

A growing number of fast-growing, fast-moving organizations rely on a microservice-based architecture. And while you can't mention microservices without mentioning Netflix, there are many other brands which apply a microservice-oriented approach to address the challenges of their digital transformation.

STEP 1: DEFINE STRATEGIC GOALS

In a first step, get some clarity on your strategic goals and how these benefit from this new approach. Broadly speaking, there are three objectives you could aim for:

1. GAIN AGILITY

For customer-centric initiatives it becomes increasingly important to get new features out quickly, so customers can benefit directly and remain loyal. Instead of showing something fresh every quarter, aim at deploying updates at least daily.

2. SCALE WITH GROWING TRAFFIC AND BIGGER TEAM SIZES

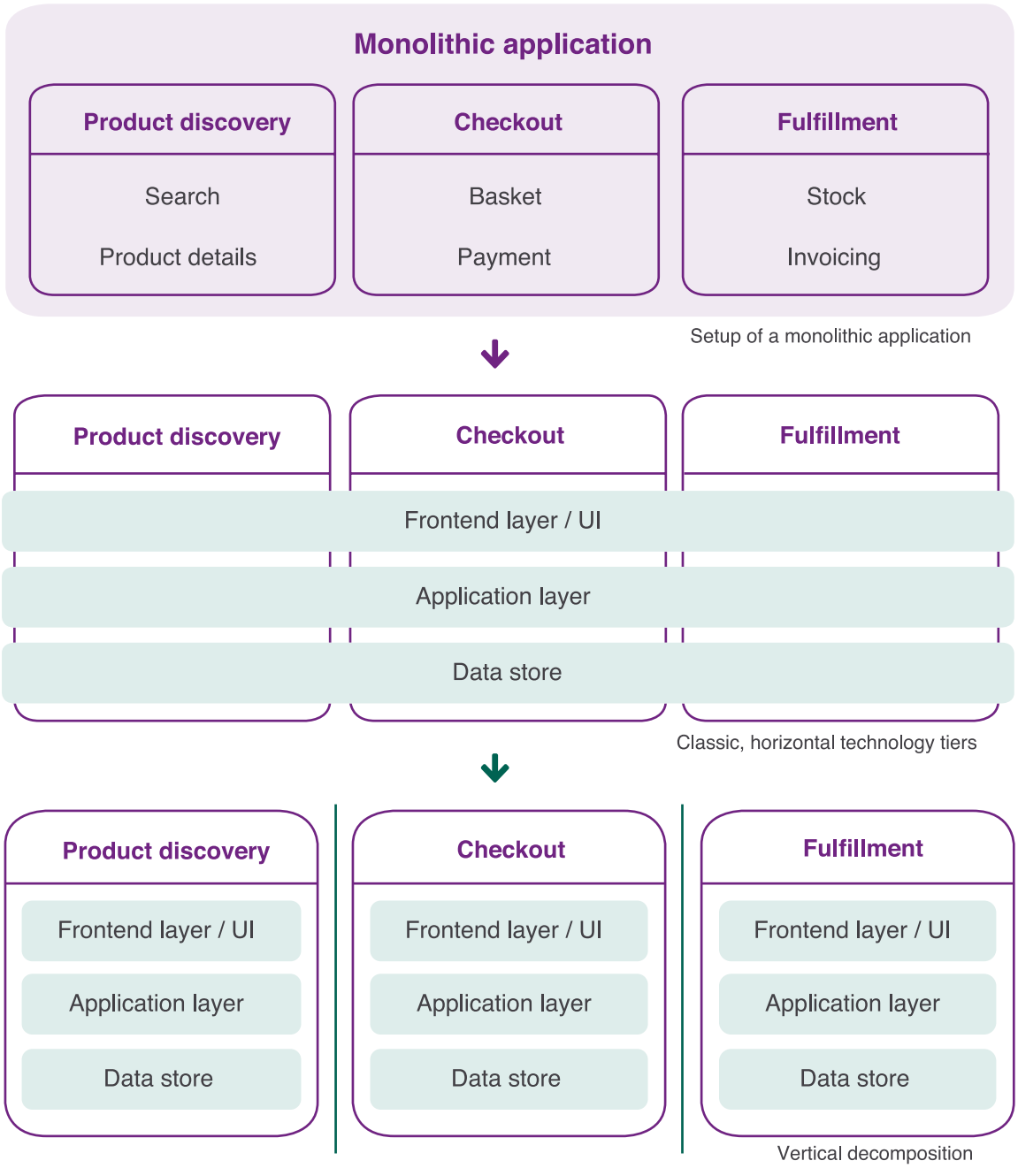
In a similar fashion, you need to make sure to prepare your infrastructure for the anticipated growth. On one hand, scaling refers to the software being resilient enough to handle large amounts of traffic – on the other it describes the architecture's ability to allow for growing teams and parallelized tasks.

3. MITIGATE VENDOR LOCK-IN

In order to manage the risks of relying solely on one technology for business-critical processes, you might consider developing your own microservices for these areas and thus regain technical independence from any third-party vendor.

STEP 2: DEFINE BUSINESS DOMAINS ACCORDING TO THE CUSTOMER JOURNEY

Next, analyze your current software ecosystem from the perspective of the customer journey it creates. Usually, there is an evolved, highly individualized monolithic commerce application in which very different business functionalities are encapsulated. Try to define the boundaries between the various stages of the customer journey and which services are needed for the respective customer experience. The process of discovering a product, for instance, is completely separate from the checkout and fulfillment logic that happen later on in the course of the journey. Then, within the bounded context of product discovery, find smaller functional units that can work independently from one another, such as search or product details.



STEP 3: REVIEW CURRENT SYSTEMS AND MATCH DOMAINS TO GOALS

In this step, look at each domain individually and determine how well the respective parts of the application match the business objectives. Use a simple, traffic-light system to specify if the objective is already fulfilled (green), if some work needs to be done (yellow) or if the objectives cannot be met at all (red). For instance, the organization could focus on the shopping cart, think about what should be possible with it in the future (e.g. multiple locales, multiple channels, mobile-friendly, etc.) and then assess what the current status looks like. This simple method also helps prioritize the necessary effort. And as a side-effect, using this threefold classification makes it easy for non-technical staff to better judge the challenges at stake and follow the progress.

STEP 4: FORM NEW TEAMS AROUND DOMAINS

A fundamental part of a successful microservices infrastructure is to align teams to the services and make them autonomous. Ideally, each team owns one (or more) microservices and is completely responsible for what they are and how they perform. Traditionally, teams are organized around horizontal technology tiers, such as frontend or data storage. Here, developers do their work across all business domains and might be working on search functionality in one week, and on checkout features in the next. In contrast, microservices are planned and built as self-contained, isolated units which do not share either code or database. As a result, a given application will be cut into vertical parts, resulting in fully functional parts that contain a specific part of the overall business logic. Organize your teams according to these vertical parts, making sure that each team has a broad skill set – involving specialists for UI, backend development and operations – so that they can work autonomously.

STEP 5: SOFT MIGRATION

The decentralized nature of microservices is a major advantage for their introduction into a company's existing IT landscape. They can take over processes formerly assumed by the monolith – one part at a time. Rather than making the transition with one fell swoop, teams can gradually introduce more microservices to the picture.

A good starting point for an e-commerce business is to think about an alternative way to store and deliver their catalog data. Implement a product data microservice parallel to the existing legacy application and update its data regularly. As a result, this new product service can serve data to other clients, such as internal search or price comparison sites and successively take over responsibility from the original application – to a point, where products are not handled by legacy software anymore.

WHAT STEPS ARE REQUIRED TO INTRODUCE A MICROSERVICES ARCHITECTURE?

1. START SMALL – EVOLVING TAKES TIME

“Don’t switch from a monolithic architecture to a microservice architecture all at once. If you have a monolithic server, you probably have repositories, deployment tasks, monitoring and many other things tightly set up around it. Prioritize the things that need to be understood, and do the smallest thing that teaches you the most. Do that over and over again. With this incremental approach you will quickly take the risk out of the transition.”

Steven Coymans, Practice Lead Commerce at SQLI

2 THE DOMAIN-DRIVEN DESIGN MEANS CLOSE COLLABORATION

“DDD is useful for both monolithic and microservice architectures, but there is a natural correlation between service and context boundaries that really helps microservices. For software architects to really understand the different business domains, close face-to- face collaboration of the business and technical teams is required. Together, parties must define the domain, iterate through the definition and focus on the business problems. Only when a microservice has a correctly bounded context, it’s self-contained for the purposes of software development.”

Mark Blockhuys, CTO at SQLI

3 ORGANIZATIONAL SHIFT – BUILD YOUR TEAMS AROUND MICROSERVICES

“Implementing the microservice architecture design isn’t simply a technical decision. It’s expensive and as a long-drawn out project, the impact goes beyond the in-house development team. This transition needs commitment from the senior management, and not only financially. The development team will need to be organized differently: separate, cross-functional teams that tackle different microservices and are empowered sufficiently. That requires both a cultural and a management transformation.”

Ramy Salama, Technical Lead at SQLI

SUMMARY

Businesses that have a clear customer-centric strategy require a flexible and scalable software infrastructure in order to be successful in a disruptive marketplace. The evolutionary architecture in the shape of microservices is the next logical step. Introducing a microservice methodology – as complex and startlingly it may seem at first glance – brings agility and speed to your organization. Luckily, from a technical point of view, you do not have to start from scratch.

To capitalize on microservices, your organization needs to be capable enough to organize and manage it. Required tooling and infrastructure are offered by providers like commercetools, a leading e-commerce solution that brings microservices, API, and cloud together. Their cloud-based microservices solution contains 300+ commerce APIs that can be used individually. With ready-made commerce blocks you can easily create or supplement your own infrastructure. That enables true business agility and e-commerce success.

However, technology is only one side of the equation. In order for this strategy to be successful, it requires a long-term effort in terms of the organizational structure as well. By breaking down business functions into independent apps, you allow new features to get to market very quickly because you remove dependencies between teams. Only by enabling them to create, maintain and run their own microservices, do you have the opportunity to scale your business and stay atop the food chain in the next decades.

ABOUT SQLI

At SQLI, we craft digital experiences that define the success of great brands and companies. Being around since the ‘90s, we have a proven track record in e-commerce and creating digital experiences. We do this by engaging customers and driving business, based on a sound technology.

SQLI.com

ABOUT COMMERCETOOLS

Commercetools, the next-generation software technology company that offers a true cloud commerce platform, provides the building blocks for the new digital commerce age. An agile, componentized architecture improves profitability by significantly reducing development time and resources required to migrate to modern commerce technology and meet new customer demands. It is the perfect starting point for customized microservices.

commercetools.com

Do you want to know more about evolutionary architecture and microservices?

READ MORE