THE STATE OF THE ART IN AGILE SOFTWARE DEVELOPMENT

"A point of view for the financial services sector"

Notice:
This document is interactive and clickable. Use the latest version of Adobe Reader to open and click on the hyperlinks to navigate to reference material.

TABLE OF CONTENTS

01 EXECUTIVE SUMMARY

For a bank or an insurance institution, technology is "the" **business**. Now, more than ever, customers expect intuitive user interactions in real-time, anywhere, and on any device, every time they communicate with a financial **institution**. To address rising customer expectations and ever-growing competition from digital-native firms and FinTech start-ups, financial institutions must increase their digital maturity. **They** must be able to nimbly plug-and-play new services and components (into their operations) to meet changing customer demands, and the new skills and mindsets required.

For many companies, the move towards becoming a technology **business** is a long and complex journey. After all, it requires significant changes to the existing IT landscape. The IT portfolio of a thriving technology business must be nimble, easy to use, and built on top of cloud-native capabilities - predicting and adapting to ever-changing demands and requirements. However, we understand that applying a new application blueprint is far from straightforward. Apart from FinTech startups, older technologies are a common feature in the core systems of most companies.

This means that digitally mature financial institutions must have a state-of-the-art software development life cycle that is fully integrating with their business, covering not just technology, but also people and process-related aspects.

There are three key characteristics to consider if this is to work:

1. **People**
   - Align engineering teams to value streams, focused on client product delivery
   - Embrace open- and business-first-connected policies
   - Take all business, technology, innovation, and collaboration skills into account
2. **Process**
   - Create and operate a business- and technology-connected life cycle
   - Establish an integrated end-to-end process covering everything from strategy to operations
   - Ensure all quality- and compliance-related aspects at any point
3. **Technology**
   - Create Communities of Practices that understand the impact of new technologies and how they can be leveraged to improve product feature velocity and quality
   - Make use of DevOps, DevSecOps, and Continuous Integration / Continuous Development (CI/CD) approaches
   - Address legacy through rationalization and reducing the technical debt

This point of view paper, created by Capgemini's experts will detail the key elements of a sustainable and scaled agile software development life cycle (SDLC) focused on the financial sector. We hope the insights in this point of view paper help you to drive technology business transformation in your own organization.

Manchester, UK, April 2021

Gunnar Menzel, CTIO Europe North & Central

Melbourne, AUS, May 2021

Sudhir Pai, CTIO Financial Services

DevOps Definitions

DevOps is a way of collaborating and industrializing practices, across people, processes, and technologies while leveraging highly automated approaches to deploy solutions that evolve as fast as a business needs them to.

DevSecOps Definition

Extending the definition of DevOps, security is integrated across the entire lifecycle to incorporate practices and automation tools to address security-focused non-functional requirements.

Agile Definitions

In a software development context, agile was first coined in 2001, (http://agilemanifesto.org/) and mainly focused on the development cycle, covering all aspects of it – from gathering requirements to code testing execution.

Agility Definition

Agile is an organization's ability to sense and respond to change both adequately and in due time.

Waterfall

- A sequential software development process, where progress flows steadily toward the conclusion through the phases of a project(s)phases of a project(s)

Waterfall + DevOps

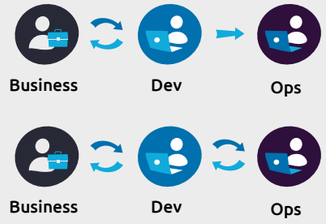
- Improvement of time frame for testing and deployment
- Limitation of error risks

Agile

- Acceleration of time frame for upstream phases: design, developments, testing
- Focus on added value for the business

Agile + DevOps

- Acceleration of TTM via actions on the whole delivery cycle
- Improvement of quality via industrialization and automation of manual tasks



> "Being agile means constantly being adaptive to change.
> It constantly raises the question of what we can do better to succeed."
>
> **2017, RALPH KIENZLER,**
> **Head of Group Accounting,**
> **Landesbank Baden Württemberg**

return to table of contents

02 PEOPLE

Moving to agile ways of working involves people change; as our report titled '**How to turn agile a company**' outlines, it "is not turning a traditional **company** into a digital pure player but transform the way the company innovates and delivers products and services, to face its challenges in the digital market, in accordance with the company's DNA". From a software development perspective, this means focusing on empowerment, culture, and skills.

return to table of contents

2.1 EMPOWERMENT

Scaling agile successfully requires a high-trust environment. By **empowering teams** to make decisions independently, without centralized approvals, we simplify governance, foster experimentation, innovation, and reduce delays through better product development flow and throughput. The delegation of decision-making occurs only when the team knows what its responsibilities are. Therefore, agile teams in market-leading organizations make and deliver short-term commitments as increasing predictability is achieved when teams deliver on small, short-term feature commitments rather than **long-duration requirements**.

To maximize decentralized decision-making, agile market-leaders focus on:

- Gaining management support for distributed decision-making, based on predictable feature outcomes and a constant team capacity
- Communicating company strategies and values clearly and comprehensively
- Training employees so that they are comfortable taking on new responsibilities
- Training middle-management to gain the skills required in the new environment
- Extending trust to external vendors through agile contracts, Making them partners in the journey towards agility

Scaling agile effectively does require some central authority to govern the transformation. The transformation will require the creation of a Center of Excellence (CoE) that includes cross-functional leaders who will be responsible for defining the vision, principles, framework, strategy, and overall success metrics. This CoE will help steer the organization in a common direction and drives cultural change.

The multidisciplinary teams in a CoE include not just developers and testers, but also security, legal, and compliance to foster conversations "early and often". With a industry regulations, it is critical to balance speed and control, as well as being sensitive to risk. As an example, two years on from the program's launch, **Barclays** has doubled its throughput and halved the time to market for new **features.**

Our research reveals that the CoE is responsible for:

- Design of the agile playbook tailored to the specific needs of the enterprise
- Setup and rollout of training to engineering teams in agile practices
- Creation of change agent networks to provide feedback from the field
- Design, on-boarding, and support of DevSecOps (CI/CID) pipelines across target technology stacks
- Displaying leadership commitment and ambition to change ways of working
- Building and housing the analytical capabilities to identify engineering good practice across the enterprise that can be harvested for institutional continuous improvement

Agile market leaders gauge transformation success by leveraging qualitative information and feedback. This ensures that their agile practices deliver value throughout the organization. To ensure this, it is important to identify the right people for the right roles when building an agile CoE within any organization. This process side-steps any potential pitfalls that arise from creating a new silo and enables the measuring of any actual impact the center is having within the organization.

2.2 CULTURE

Innovation is critical if financial institutions want to stay competitive. Therefore, organizations must foster a culture of psychologically safe experimentation and collaboration to encourage innovation across their companies.

In the latest Capgemini Research Institute report: **Agile@Scale** we discovered that 82% of agile market-leaders say culture and mindset are their biggest obstacles to scaling agile effectively. For example, a leading global bank executive noted: "People want their minimum viable product (MVP) to be such a huge product that they forget what MVP means."

MVP = is a version of a product with just enough features to be usable

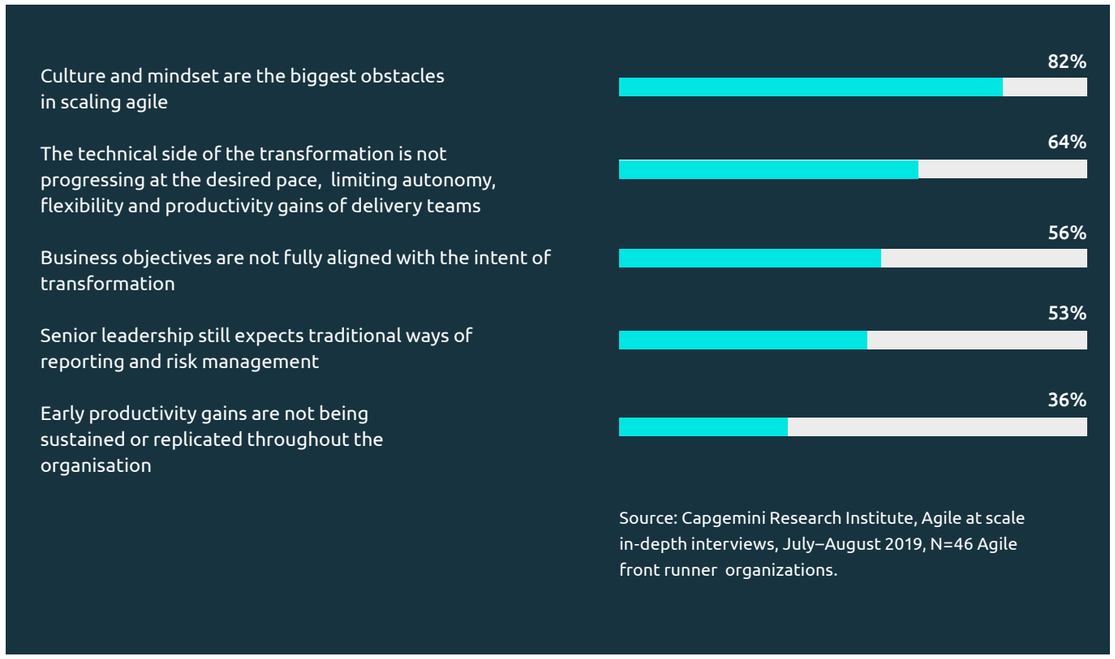


Figure 1: key challenges faced by agile front runner in scaling agile

return to table of contents

We see that while large financial institutions are also undergoing transformations, digitally-native financial startups hold on to existing products and application portfolios, most of which are still located on-premise. Product support is spread across loosely integrated teams, business units, and organizations. To turn a agile transformation into an agile market-leading strategy, organizations need to foster a culture of "psychological safety" and encourage a "fail fast" mindset.

It is critically important to empower team members when tackling many of these problems. But where do you start? Organizations need to have clearly defined cultural practices and business goals – with clear metrics to track the health of both if they want to overcome the challenge here. To further enable a company-wide agile culture, organizations need to address the following areas:

- Enable longevity, so teams can better foster an agile culture
- Ensure teams are free from interpersonal risks
- Leverage collaborative tools and platforms to spread common practices quickly

Harvard Professor **Amy Edmondson** defines psychological safety as "a shared belief held by members of a team that the team is safe for interpersonal risk-taking." A psychologically safe environment for team members is created when the surrounding organization provides the freedom for teams to experiment and "fail fast." There are various ways this can be achieved:

- Through long-living, multiskilled, product-aligned enterprise-wide DevOps teams implementing an agile approach
- Providing adequate innovation time each sprint for teams to explore new ideas and experiment with proofs of concept
- Establishing a virtual Andon cord that enables any member of the team to effectively stop the production line if they detect an underlying quality problem or way to improve efficiency

Long-existing teams are a well-known and proven practice in many agile delivery frameworks – it is one of the common and dominant practices in the agile way of working.

By equipping autonomous teams with self-service platforms, cloud landing zones, and executable reference architectures, ensures that people are *doing the right thing, as it is the easy thing to do*. This delivers the controls that are needed by regulated organizations support autonomous teams, while still encouraging innovation.

Self-service platforms and landing zones with "built-in" security, governance, and quality, are typically cloud-hosted and are built and managed by a separate platform team – often described as the *cloud foundation*. The job of platform teams is to promote the use of standardized delivery platforms across different product teams so as to avoid them reinventing the wheel. **Mario Suykerbuyk, Eneco CIO, and Marjolein Holsboer, Capgemini Account Executive describe the cloud foundation as "a foundation for high-performance teams to deliver faster, higher-quality to their customers."**

MICROSOFT CASE

Microsoft was an early adopter of the platform team model, forming the One Engineering System (1ES) team in 2014. The 1ES team's mandate was to empower every engineer in the company by standardizing the best tools available to them. The team has shared the lessons learned in this DevOps case study and their 5 steps to culture change whitepaper (Five Steps to Culture Change).

Another key part of the product and platform team model to drive a fail fast, learn, share, and accountable culture is the "InnerSource" (i.e. enterprise-shared source) pattern.

The book "Adopting **InnerSource**" defines this as: " A collaborative and empowering way of involving employees in making and implementing decisions throughout the corporation. It embodies a philosophy of human relations, an approach to rewards and motivations, and a loose, adaptable set of tools and practices."

Open source is an efficient mean of sharing solutions to avoid reinventing the wheel. InnerSourcing explains the concept of open source, but it focuses only on a company. Teams share it among themselves. Using this method reduces the security risks posed by malicious code coming from unknown origins and ensures that non-public code is only available to selected teams. Organizations with a large developer base are adopting **InnerSource** by following GitHub's core tenets of InnerSource. This practice brings shared ownership, shared responsibility, and productive and collaborative culture to any organization.

A culture that emphasizes psychological safety and failure fast philisophy has a profound influence on the performance and innovation of a business. For regulated organizations, long-lived autonomous agile teams create an optimal culture for agile.

2.3 SKILLS

Since increasing the use of agile development capabilities during the mid-2000s, there has been a tremendous rise in team and individual employee expectations. Today's complex landscapes require top talent to build and maintain applications**.** In the shift towards agile development models, the skills needed in software development spans wider than just translating specifications into machine code and focuses on building and maintaining skills that enable a person, a developer, to be an outstanding engineer.

Organizations are not only competing to attract new talent but must also create an environment where employees can be themselves and further develop their skills. Three pillars are essential in overcoming this challenge:

- **An engineering culture of development and growth:** As software engineering is the key discipline for digital value creation, mastering this discipline is the foundation for success. It is the key to having employee growth and continuous development as a part of a company's engineering culture.
- **Technology focus and strategy:** A key capability of this new kind of software engineering organization is a fast and adaptive approach to technology evolution and skills development. This approach needs to balance freedom and technology alignment decisions to be successful.
- **Team skills:** Shifting perspective to teams as the unit for value creation will enable teams to make the most of everyone's expertise.

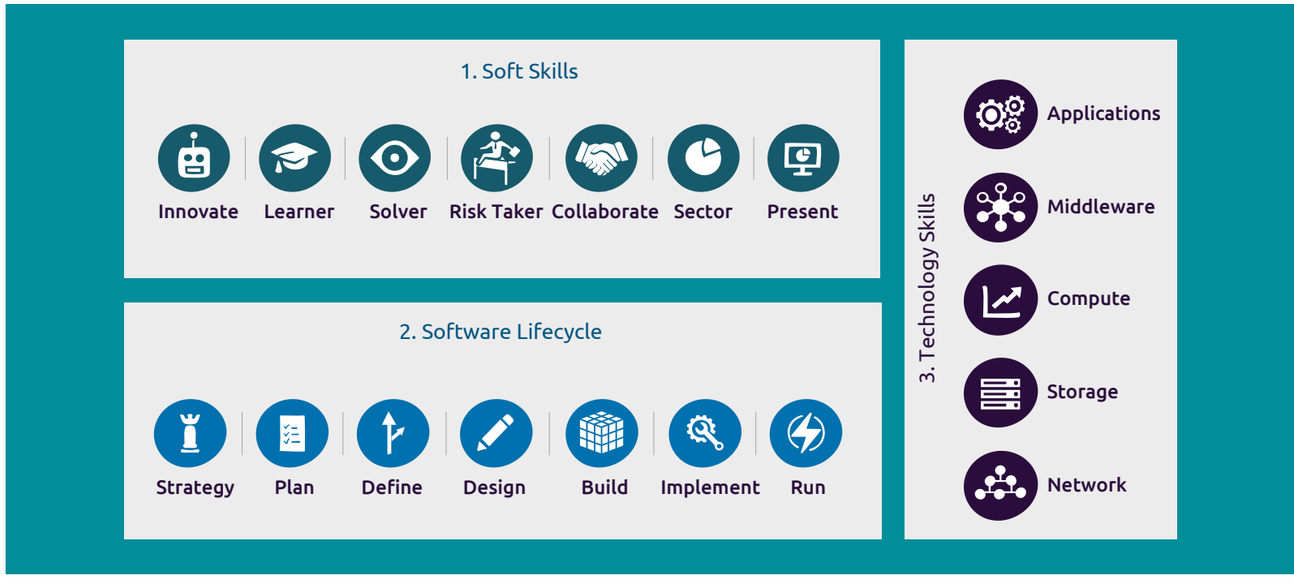
02
return to table of contents

An engineering culture of development and growth

For traditional software companies, software engineering excellence is a part of their DNA. This also applies to digitally native organizations, such as **Spotify** and **Netflix.** When **Spotify** shared its experiences on crafting its engineering culture in 2012, many saw it as a new agile framework and various companies tried to adopt this engineering **model**. However, several "so-called" Spotify implementations failed soon after they began. Growing an engineering culture takes time, and the most important factor here is the continuous learning of the entire organization.

OTTO, a popular German paper catalog retailer – and one of Amazon's few competitors in Germany – successfully shifted into the digital area. A core part of this transformation was the engineering culture they established, and the company is now proud to share their open-source assets in conference talks and **articles.**

Agility from a people perspective might cover 3 main areas: skills, software lifecycle, and the technology stack. In short, it refers to people having mature innovator, learner, solver, risk-taker, collaboration, business domain, and communication skills as well as being able to cover the full engineering skills from build to run.

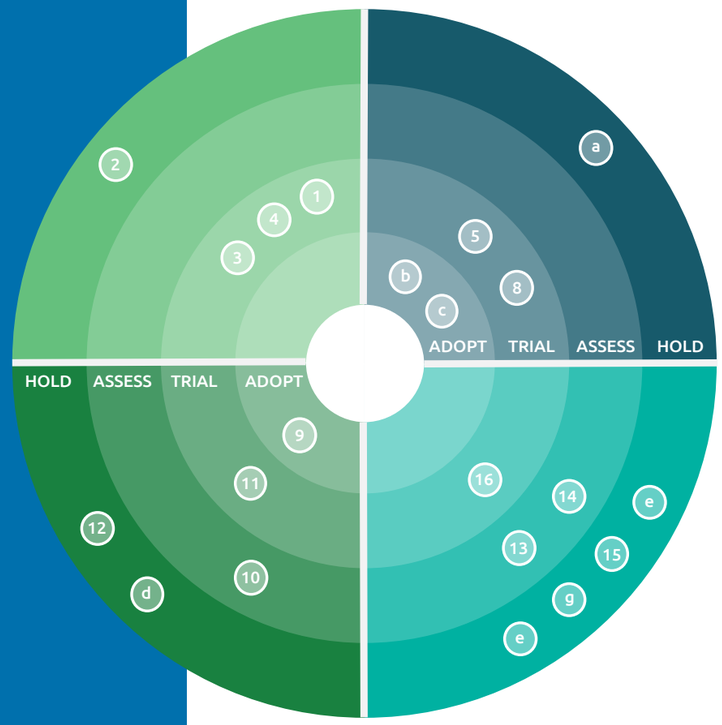


Technology focus and strategy

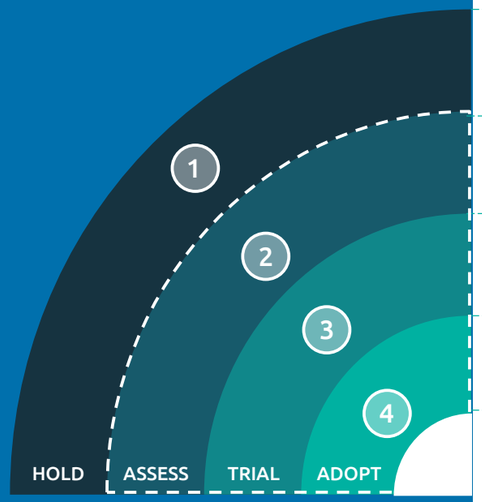
The number of available technologies is growing every day. The periodic table of **DevOps** alone highlights the incredible number of tools available in today's market just for DevOps. These new tools and opportunities are also a challenge to organizations. Which tools are future-proof? How should organizations continuously adapt technological roadmaps? What skills should they develop or acquire?

One example of how companies are addressing this, is through the adoption of technology radars, to provide an overview of current technologies. They provide a clear indication of upcoming trends and new solutions, where the organization can jointly evaluate and adapt their technology roadmap if feasible. However, this needs to be a joint effort from the IT organization.

After all, adopting new technologies and embedding them into existing systems is a key objective for any organization's architecture experts, while platform teams need to provide easy access to any identified technologies. Companies such as IKEA and **Zalando** are just two examples of how organizations can potentially define their technology roadmap to get the results they want.



A technology radar that maps current and emerging technologies where each number equates to a technology trend



	Description
Talk about it	Technologies that have been identified and are not yet well understood or are deemed to be too immature to pursue further.
Look at it	Technologies that are deemed to have potential, requiring further investigation to determine its value to us.
Trial it	Technologies that are perceived to deliver value and are deemed appropriate to evaluate in order to prove or disprove the commercial opportunity.
Deliver it	Technologies that are proven to deliver revenue growth or cost optimization to the business within 3 years.

return to table of contents

Team skills

Team optimization comes from two important sets of team skills: balancing cross-functional expertise and self-organization – which must be monitored by the teams themselves. With growing product centricity, teams increasingly own the entire life cycle. This requires the ability to collaborate across teams and, of course, a deep specialization in the expertise needed to complete whatever is being worked on. Initially, this led to the T-shaped capability engineers, i.e. deep specialism in one discipline e.g. web development, and a general appreciation of other disciplines. However, with the shift to product centricity, team members are expected to understand at least two disciplines in detail that cover for example a build and run aspect. This has led to the development of "Pi shaped" engineers. In this model, an engineer has two deep skills, either web development and business domain expertise, or back-end development and operations monitoring. Pi-shaped capability development is today being practiced in a minority of enterprises but will become more widespread as the shift to product centricity in teams gains popularity.

There are several ways to optimize team structures across and along a software development lifecycle (SDLC): Dojos, and the combination of Chapters and guilds are two examples:

- **Dojos** provide an immersive environment for teams to practice lean, agile, and DevOps methodologies through timebound challenges in a dedicated space Japan, dojos are training or education halls. Technical coaches work with teams to help them learn and refine their craft. **Capital One** adopted the Dojo framework to share insights and expand its DevOps initiative
- Chapters and Guilds are an evolution of the communities of practice concept and provide a way of building cross-team collaboration andgood practice sharing. Chapters bring together members across teams within a particular product domain that can share expertise related to that product. Guilds are enterprise-wide and often focus on specific skill areas such as Testing, digital engineering, data analytics, etc.

In an agile transformation, the technical skills of the individual are important for the development of new products or services – but team skill sets are also extremely important if organizations want to realize the promise of a truly agile company. Learning and development should be a cornerstone of any organization's culture to ensure we offer individuals and teams the necessary platforms and channels for more learning. Finally, making technological roadmaps visible to the organization will both guide skills development in-house and help connect employees in an empowering way.

03 PROCESS

The evolution of technologies creates new possibilities and challenges in governance, trust, and regulations that require organizations to expand their processes beyond IT and evolve their software development capabilities.

return to table of contents

3.1 A BUSINESS- AND TECHNOLOGY-CONNECTED LIFE CYCLE

When designing a race car, two principles are crucially important to the design team: firstly, it should weigh as little as possible to maximize the power to weight ratio, and secondly, its surfaces must be extremely aerodynamic to reduce drag and friction to an absolute minimum. It may be surprising to learn that the very same principles apply to the design of highly agile organizations and, more importantly, the connection across different parts of the enterprise.

Let's first look at the weight principle. Historically, organizations built a "Big is Best" mindset into software. Waterfall life cycle teams typically had large, distinct silos for business analysis, development, testing, release, and maintenance. Requirements were also big to justify the major investments needed – and would easily run to hundreds of pages of documentation. Even the duration of the project was big, with average build times exceeding 12 months before the business would see the results of that investment. With time, however, this big-centric mindset has started to lose its attraction. Agile has provided a rationale that explains why "small" is far more appropriate to today's rapidly changing business needs. Lloyds Banking Group began its Agile transformation in 2014 with a focus on 10 key customer journeys. It found that – faster software development and delivery cycles dramatically improved the time to move from idea to prototype. In 2018, more than 50 customer **journeys** underwent transformation.

Today, teams consist of no more than 7-9 engineers (often called pizza **teams**). Requirements are gathered regularly and typically can be fulfilled in less than a week. Even investments are reducing in size – annualized planning typically resulting in large capex (capital expenditure) budgets, are now allocated to smaller continuous delivery projects that shift funding to an opex (operational expenditure) model instead of capex.

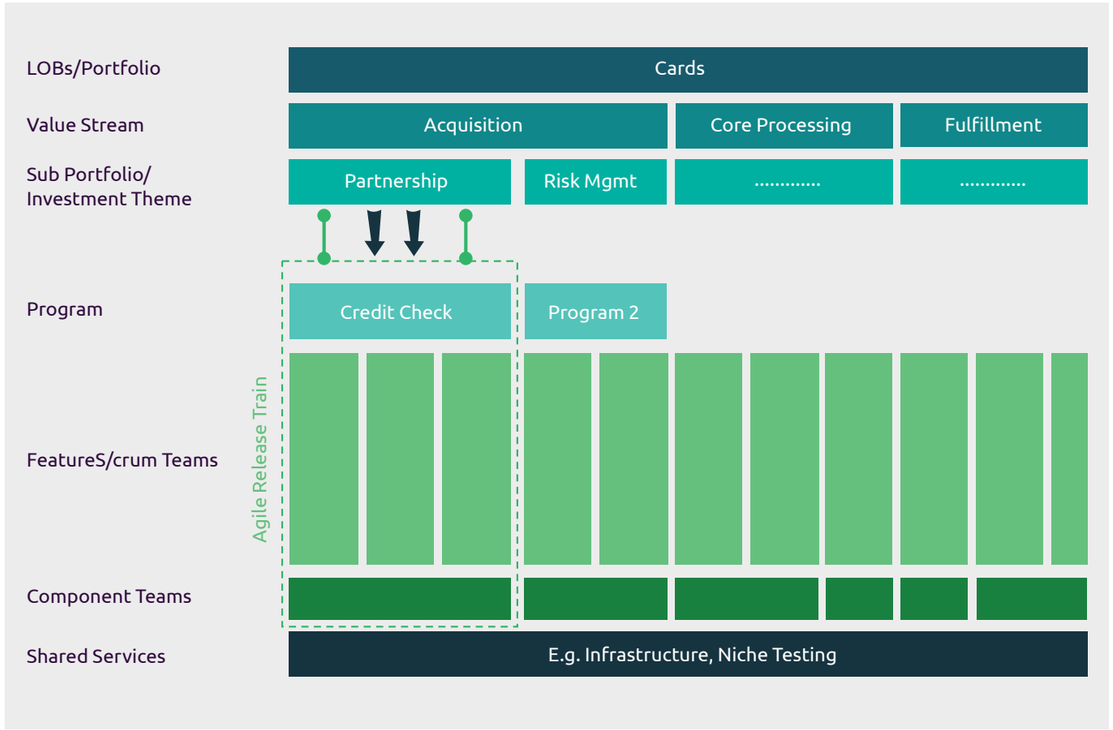
This can be a little daunting, but the good news is that we can do this in various ways. We can coach the requirement owners (or product owners) to consider the smallest requirements for testing. For example, is it necessary to specify all the business rules for the entire transaction from end-to-end or can we divide the transaction flow into separate smaller tests – smaller tests such as login validation, bank balance validation, address validation, etc.? If necessary, we can also focus on reviewing an entirely new business product design. For this process, we can specialize coaching and training in minimal viable product (MVP) design for business communities. However, it's important to understand that there is rarely just one solution here – a combination of solutions works just as well. We might deploy a combination to address a wider business community.

We are just starting to address the new domain of large financial services organizations with the adaptation of financial processes to support scaled agile ways of working. We align the IT funding cycle with old project-based annualized budgets. However, given the significant changes that can impact markets with little warning, such an approach quickly becomes untenable. As a result, businesses need to have the ability to reprioritize IT funding without losing the work they have already done whenever market changes occur. To support this dynamic approach, we see increased adoption of lean portfolio funding methods, which agree on budgets according to delivery capacity, generated business value, and a much shorter return of investment (ROI) horizon.

As agile adoption increases, the challenge of steering the different moving parts towards an overall goal and destination becomes paramount. Scaled agile approaches have grown over the past year, with the Scaled Agile Framework (SAFe) pre-eminent amongst them. Through a set of disciplined techniques and organizing principles, we orchestrate the work items of many agile teams which enables dependencies between teams to be quickly identified and managed. Such techniques include:

- Program increment planning to align business goals and team plans at the start of a set of sprints to the outcomes, dependencies, and critical milestones you expect. We align everything between agile teams
- By aligning value stream structuring with agile "release trains" we bring teams together into a cohesive organization to deliver features associated with a specific value stream or product area
- Additional governance roles including release train engineers to ensure that teams are regularly synchronized to maximize velocity and avoid conflicts.

Just remember, having a light car will still not win any races, as it might be the slowest in the field. What is critical is that it can reduce drag and remove unnecessary friction to remain competitive. Looking back at the traditional life cycle methods and the related structure of IT and business teams, it is perhaps not surprising that waterfall projects had long duration times. There could be significant frictions caused by hand-offs and delays between business and analysts, which could lead to conflicts over development, testing, design execution, and production topics.



02
return to table of contents

An organizational shift is occurring to align product-centric teams with specific value streams. The example shown (diagram 7) describes the situation for a credit card company where business and IT have been vertically integrated through specific business products and customer journeys. Complimenting these teams are component teams that are responsible for common technology architectural components such as cross line of business (LoB) platforms e.g., finance. Over time, architectural changes can reduce the need for heavy component teams by abstracting out business functionality to a more dynamic microservices layer. This will align it with the product needs of specific LoBs.

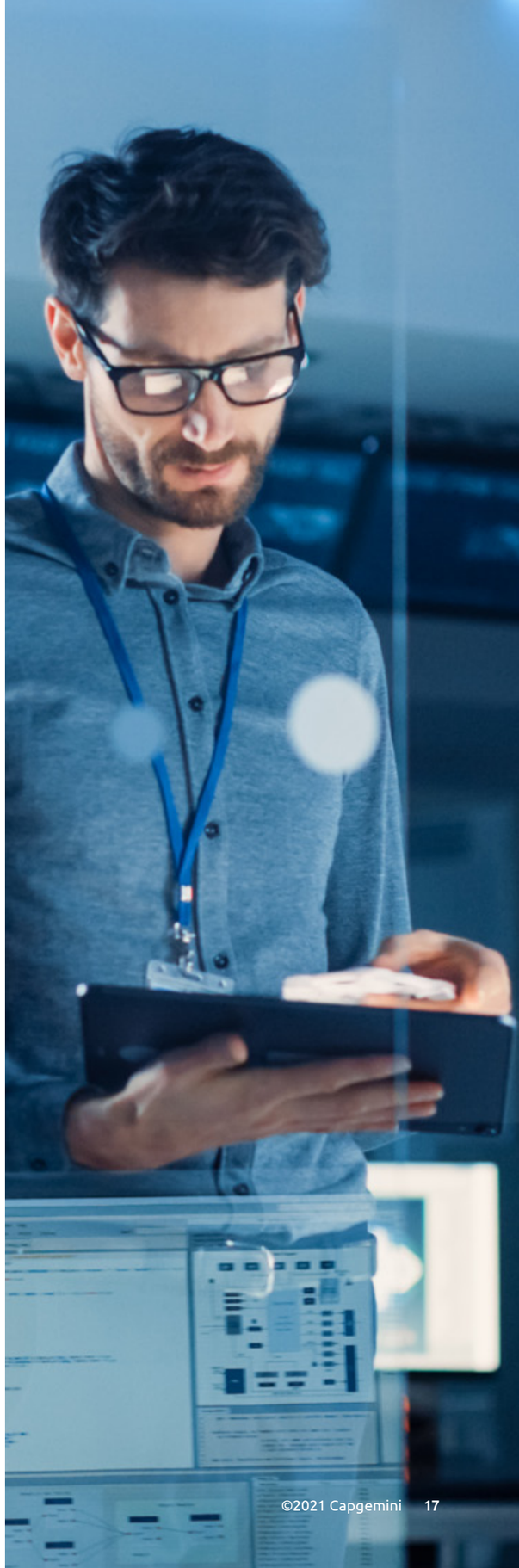
We can now enable the delivery of faster time to market and client product-centricity, by structuring transformation programs to address agile approaches at scale – rethinking the IT funding paradigm, and shifting to the outcome, accountability-based supplier models.

3.2 END TO END AUTOMATION

Leading agile organizations are leveraging Development and Operations (DevOps)-related automation wherever possible as those who automate processes are extensively outperforming their competition.
They reduce time spent on repetitive work, freeing teams to focus on generating business value and providing innovations. Automation is a continuous focus for DevOps teams, and failing to automate remains a long-term barrier to achieving the benefits of DevOps.

In an online survey of 415 IT executives, we found that

- **75%** say automation has led to increased revenue for the business.
- **76%** say the same about overall profitability.
- **86%** say automation has helped to improve the customer experience.
- **84%** say their company's agility has improved.
- Another **75%** have been able to use automation for business model innovation.



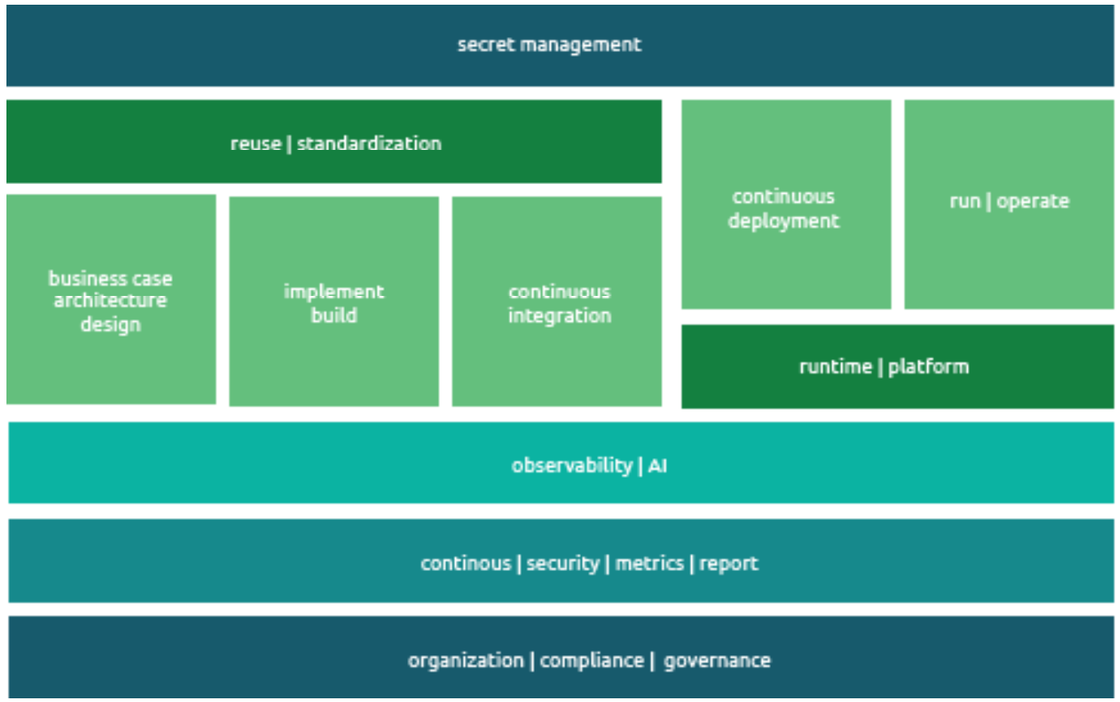
Standardization is a key success criterion for automation to be effective. When teams are leveraging DevOps automation tactics – such as everything-as-code – there should be a standardization of the base reference architecture around common cloud infrastructures and application patterns. Multiple teams in the organization can then reuse common tools (for example automation scripts), which moves these reference architectures from Visio diagrams and Word documents into reusable code templates to accelerate widespread adoption – while also bringing automation to the next level.

However, standardization is not the only thing needed to make automation dreams a reality. We must design existing, new systems, and applications for automation too. A system with many very tight dependencies can be complex to automate, while small independent components are much easier to automate.

The evolution of automation at Google describes the evolution of automation (Using a Database as an example) as following a **path**.

Automation at Google has become so-called "Site Reliability Engineering (SRE)." With SRE, Google optimizes large cloud-based systems in production, with a very strong focus on automation and a principle which states that team members should work 50% of their time on automation – making the system more reliable.

We don't only apply DevOps-related automation practices in the areas of continuous integration and continuous deployment. DevOps toolchains play an important role in their automated CI/CD workflow capabilities. But, every aspect of a team activity, and the context the teams operate in, is a potential target for automation. We group DevOps-related automation practices into the following areas:



return to table of contents

A key challenge for the enterprise is understanding what level of control to localize and what to centralize. For example, digital marketing is likely to use angular.js, an open-source JavaScript framework, or python, an interpreted high-level general-purpose programming language that requires a Java-based pipeline – whereas the back end could be a ".net" or even a legacy application, which requires a different pipeline design. Although there can be a central design authority for the blueprint, the best approach is to include members from all LoBs to provide a catalog of components with standardization on just a few LoBs – such as reported metrics from the pipeline (commits, build failure rates, deployment frequency, etc.).

Pipelines need to address the full end-to-end nature of the software life cycle. To understand what areas we must automate, we need a value stream analysis (also called a Muda assessment). Muda translates roughly as waste and refers to the inefficiencies within processes which you can seek to reduce or eliminate. This assessment walks through the life cycle with client SMEs identifying which parts are currently automated (from requirements engineering through to product deployment) and identifies what is the typical cycle time, wait time, and non-value-add time for each step. This provides a very clear graphical view of where to focus automation first and what benefits we can likely gain.

One area of attention that typically relates to the security part of DevSecOps is security testing. It is still a largely manual process and is often done with a team that is not part of the engineering teams. This can create lags and reduces velocity.

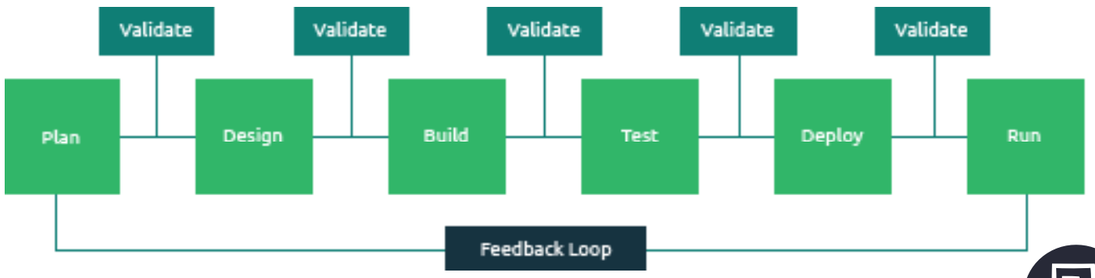
"You can lead a horse to water, but you can't make it drink" as the saying goes. This means that having the best pipeline in the world does not guarantee its use. Therefore, it's critical to invest in an appropriate support structure to help teams onboard themselves into the necessary pipelines.

3.3 QUALITY, SECURITY, AND COMPLIANCE

Quality, security, and compliance are important challenges to most organizations, but to the financial sector, a failure in meeting quality and security standards can have severe consequences. At the same time, the digital landscape requires more frequent deployments, an increasing number of services and deployments, changing regulations, and heightened customer expectations. Among the market leaders in the agile transformation, we see that the effort to ensure quality, security, and compliance continuously affects both the organizational mindset, while leveraging state-of-the-art practices and tools.

Not knowing or understanding organizational goals towards quality, security, and compliance is one of the first and most important challenges organizations need to solve. To ensure quality and security compliance, all agile teams need to understand a certain aspect of the complex web of local-, global-, and industry-specific regulations in their constantly shifting DevOps environment.

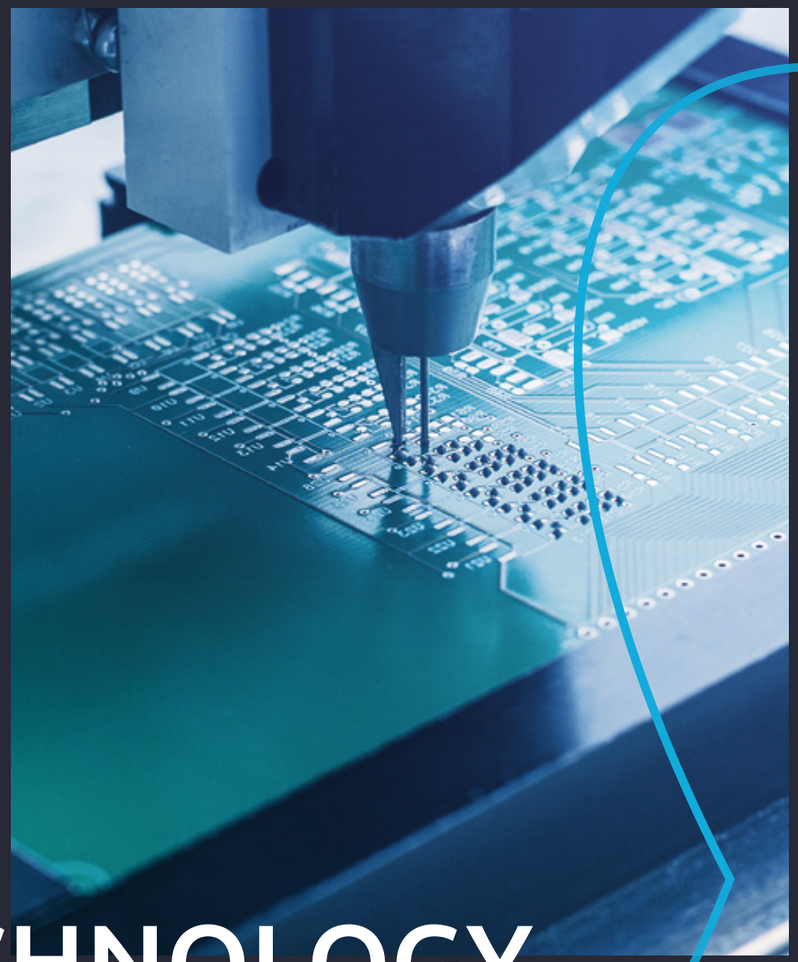
In traditional organizations, the proof that the system is delivering and complying with what you expect through dedicated teams. However, multiple teams in DevOps can provide input, implementation, and proof. As a result, everyone from the individual to the organizational level needs to have an adequate understanding of compliance, security, and quality. Clear policies and a clear strategy, together with a good understanding of the rules must be set by the whole organization. This not only requires a "shift left" approach across the entire DevOps team, i.e., that teams should focus on quality and problem prevention – it must also go as high as the boardroom, where quality must be an item on the Board agenda consistently.



02
return to table of contents

Enterprise-scale consistency is a quality challenge. Organizations could have hundreds of developers working in self-organizing teams, so it is important to have consistent implementation practices across the IT landscape. Almost all enterprises leverage reference architectures with best-practice guidance for how to build and run applications, in addition to how to incorporate governance and security. One of the more mature governance practices is to make this reference architecture executable via template sets, scripts, and tools, and to move the reference architectures beyond Visio diagrams and Word documents into code. We can achieve this by combining it with DevOps tools such as infrastructure-as-code and configuration-as-code.

We refer to observability as the measure of how well we can infer the internal states of a system from knowledge of its external outputs. In control theory, the observability and controllability of a linear system are mathematical duals that are key to the success of enterprises. This is particularly true for highly regulated industries like the financial **sector.** Observability not only enables quality and performance measurement, but also enables them to report on it – creating the feedback loops that enable organizations to learn, improve, and optimize all their processes and technology. Enterprise-scale concerns such as quality, security, and compliance are moving from an "audit model" to a "continuous model," the same as the CI/CD model has transformed the software development landscape. We continuously assess quality, compare them to standards and baselines, and ensure the correction of deviations either via automation, or direct human intervention.



04 TECHNOLOGY

Financial Institutions have recently seen significant traction around open banking, intelligent insurer, core transformation, data estate modernization and more recently digital assets and currencies and technology are at the heart of this transformation. In our annual TechnoVision trend series, we outlined 37 key technology trends that are most relevant to financial **institutions** and related to state-of-the-art software development several key technology enablers are key: microservices, infrastructure, and automation.

return to table of contents

4.1 MICROSERVICES, INFRASTRUCTURE, AUTOMATION & ARCHITECTURE

Microservices for more agility

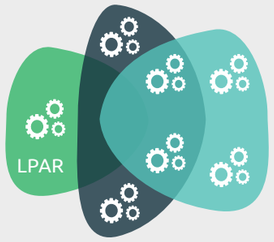
Over the years we have seen several distinct software development patterns; the monolithic approach, most prevalent in the 1980s and 1990s, followed by the service-oriented pattern between 2000-2010. However, many follow a new software pattern today: microservices.

Many view Microservices architecture as a marriage between component-oriented architecture and service-oriented architecture (SOA)**.** "Software as a suite" (Read "software as everything") has many small business components with very specific business-domain responsibilities. And the interface to the outside world through an application program interface (API) of clearly defined services.

One area where microservices depart radically from SOA is in its services' ownership model. With microservices, only a single team or person will develop and change the code for a given service – Netflix is one organization that leverages this approach.The reason microservices has taken off is wholly down to it being a realization of many aspects of Eric Evans' Domain-Driven **Design** – enabling independent, small teams to create and maintain services.

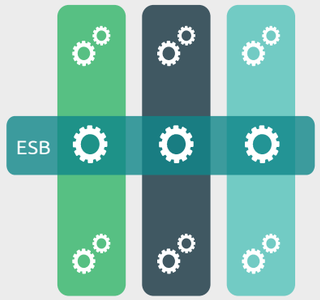
These concepts are key for the financial services industry and will continue to be so in the future – due to continuously shifting end-client demands.

Mainframe
<1990
Monolithic Approach/
Mainframe Architecture



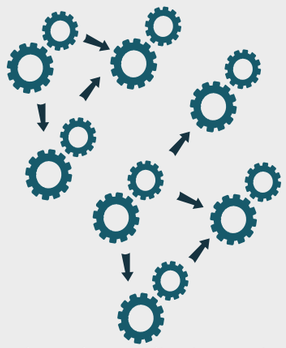
A software design pattern that includes all functional and non-functional features into one "box."

Open Systems
1990-2010
Service-oriented/
Client-Server Architecture



SOA is mainly about "exposing" discrete components of an application as web services.

Cloud
>2010
Microservices/
Cloud Architecture



Independent application services delivering one single business capability in an independent, loosely connected and self-contained fashion.

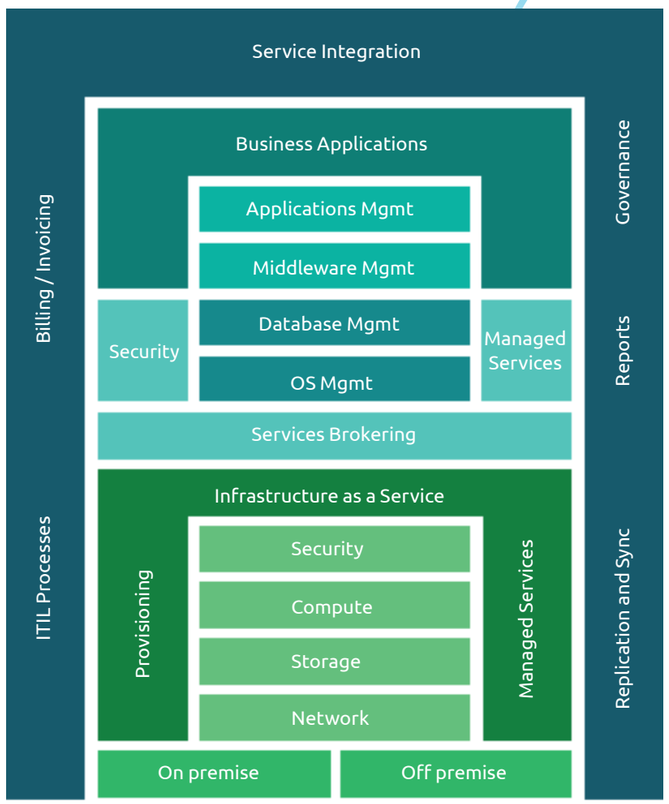
Invisible infrastructure

Following these design principles leads to a microservices-based architectural style; however, to develop and implement microservices-based applications, we must cover other aspects. What is important, from an infrastructure perspective, is that microservices-based applications demand a "utility-based" approach where infrastructure is "invisible" to the requesting **microservice**.

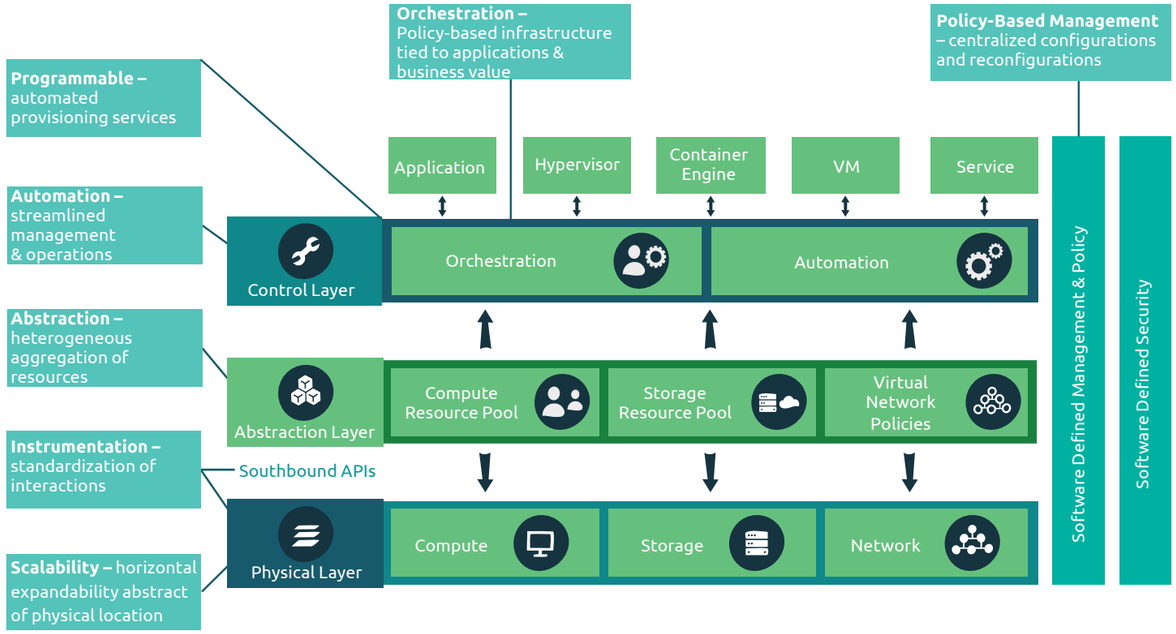
Utility-based infrastructure service delivery refers to the packaging of infrastructure resources, such as computation, storage, and data transport. We can apply this to metered services such as traditional public utilities e.g., electricity, water, natural gas, and the telephone network. In other words – a utility infrastructure service is a metered service supplied through a shared infrastructure, with separated services– therefore supporting the design principles defined above.

In our 2019 issued digital master **report**, we noted that "58% of the insurance sector's digital masters have migrated their legacy IT systems to cloud-based applications, compared to an average of 35% in non-financial services organizations". Modernizing by leveraging cloud-based capabilities can help with addressing and ultimately reducing existing technical debt.

"One benefit of adopting a cloud management platform has, of course, been cost reduction. But it's also helped improve our image within the company. We provide the platform, and our customers can directly provide the infrastructure. Now SILCA is viewed as a mature subsidiary able to provide internal customers with a mature cloud solution". 2018, Philippe Sersot Deputy Chief Executive Officer, CA-SILCA (the IT subsidiary of the Crédit Agricole (**Group**).



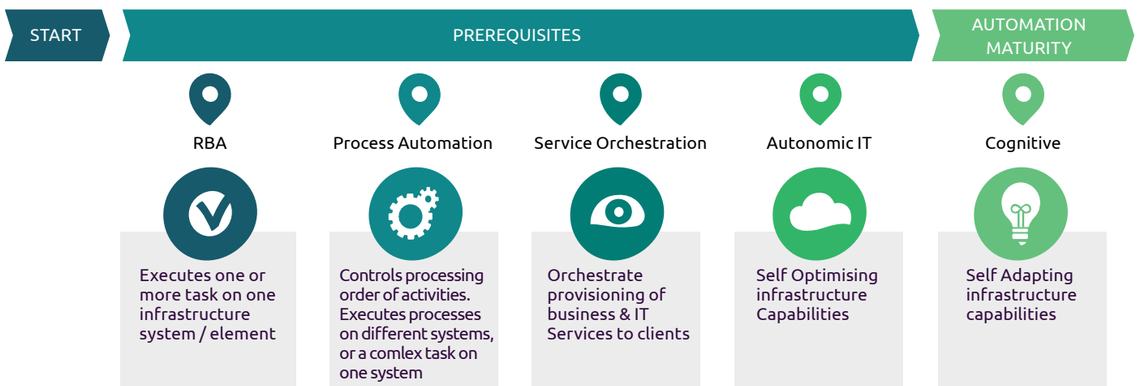
02
return to table of contents



With an infrastructure-as-code approach, it is possible to create, maintain, and operate (including deleting) the different environments for development and implementation (e.g., development, test, user acceptance, and training environments). This can happen unintentionally through automation.

Moving beyond automation

Automating cloud and infrastructure has huge benefits, but what if it could be made even smarter by leveraging AI and AIOps. For example, a Fortune 500 US **bank** found that increasing the use of AIOps could achieve almost 100% up-time. By measuring this in seconds, rather than hours, you can see customer retainment during a period of a high volume of change.



AIOps as implemented through tools such as **MoogSoft** and **Splunk** provides a highly effective approach for quickly identifying underlying production root causes and problems across the IT landscape. These platforms ingest data from many diverse sources including system log files, edge device logs, IT operations management platforms, problem ticket data, connected devices, network traffic data, and event monitoring and alert systems and using advanced machine learning algorithms, sifts, and filters the information to surface underlying patterns that enable production teams to identify the root cause more quickly. Combining this with automated workflows embedded in an enterprise's ITSM platform allows shift-left automation of incident resolution to mature significantly.

AIOps collects and analyzes IT operations data, often in real-time, to improve observability, and continuously fix and improve IT operation performance. We can ingest data to drive AIOps, from many diverse sources including system log files, edge device logs, IT operations management platforms, problem ticket data, connected devices, network traffic data, and event monitoring and alert systems.

Automation provides significant benefits to financial services as it enables cost savings, increases speed, provides better customer experience, and ensures better regulatory compliance. However, with this much hype and buzz around it, it is easy to get carried away with this process. Therefore, organizations must understand that they need to be realistic in their expectations and accept that automation is not something that they can deploy overnight.

Agile Architecture

Agile architecture is 'the art of designing and delivering the "right" solution in such a way that it can respond to change in an uncertain environment' in an uncertain context.

Being able to respond to change means, that we no longer must create a Big Design Up Front (BDUF) where we define and realize an architecture over a long period. Instead, the architecture design in an agile context:

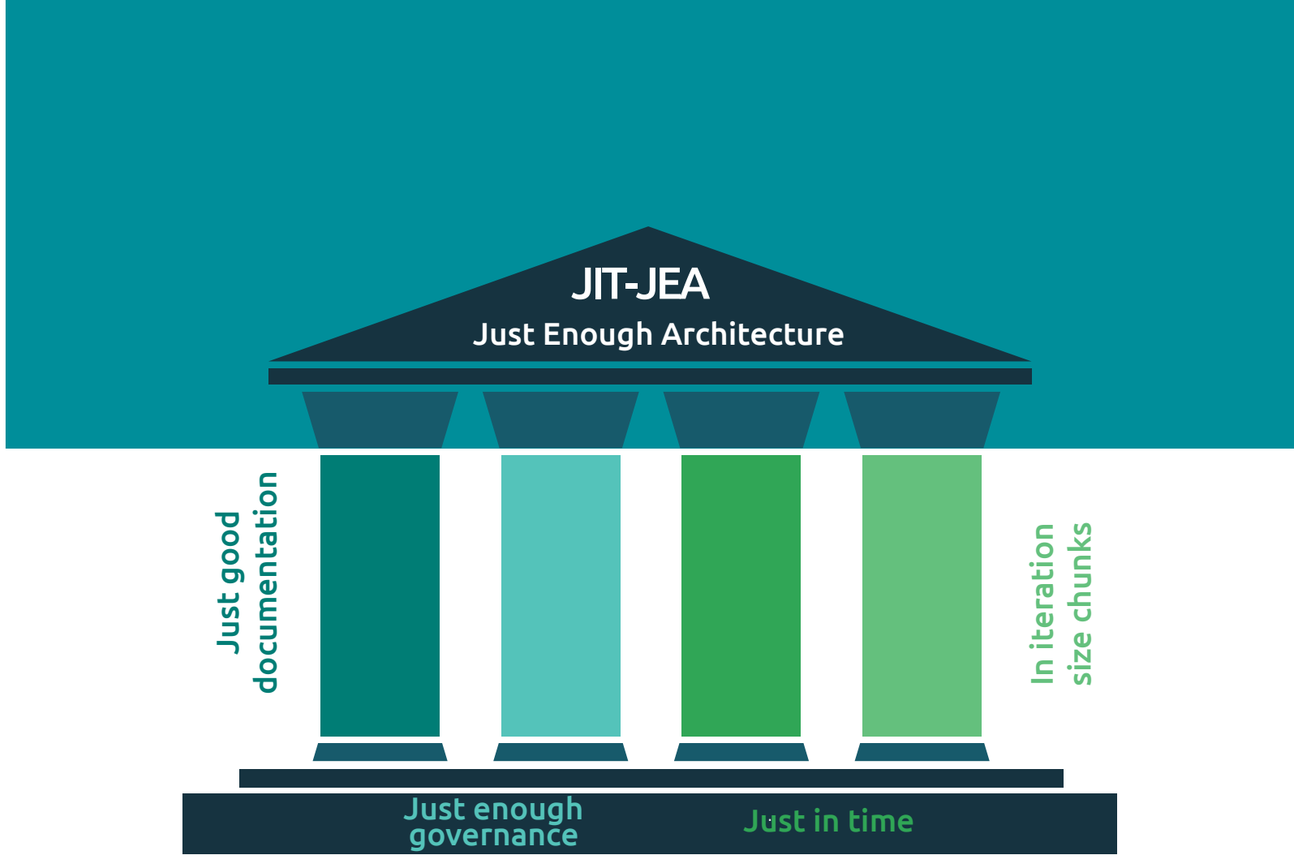
- provides the vision (intentional architecture) where the teams fit in with their (development) work,
- gives the guard rails between which the agile teams make their own design decisions (emergent architecture),
- shows where teams need to connect to ensure interoperability between systems/services (solution architecture),
- guides generic services and tooling,
- identifies design hotspots upfront (e.g., specific to some demanding non-functional requirements) which need to be tested early in the development for example with proofs of concept.

Agile architecture should break up the solution into pieces of work that different teams can take further. Ideally, agile architecture also enables designing for testability, deployability, and releaseability.

To assist our 10,000 architects across the Capgemini Group working on agile architectures, we came up with the notion of JIT-JEA (just in time – just enough architecture). The idea is that the architect will work with the agile team(s) to deliver:

- just good enough architecture
- just good enough documentation
- just good enough governance
- just in time and
- in iteration-size chunks

return to table of contents



To define "what is good enough" and to outline and articulate the level of detail and therefore the amount of related architectural material necessary, we came up with 10 principles that relate to JIT-JEA:

1. **Understand the As-Is;** if there is an As-Is make sure you have a view across business, data, application, and infrastructure of what is currently installed and what will most likely change (not needed for complete green field).

2. **Understand context;** Understanding of the Business (and IT) Context, including external facts (regulation, etc.) that may affect the results.

3. **Define principles;** Formalized and traceable Business Objectives and Principles, driven by the Business Mission and Vision.

4. **Know the requirements;** Understanding and/or delivering the functional and in particular the key non-functional requirements.

5. **Record Decisions;** Documenting the rationale for all Architectural and Design Decisions, ideally reflecting the Principles/Business Needs.

6. **Ensure traceability;** Providing clear traceability back to the Business Objectives within the Architecture.

7. **Develop solution;** Document the Solution(s) including Investigate Solution Alternatives to ensure that decisions are made holistically and not in isolation.

8. **Assumptions and constraints;** Capturing, validating, and managing any assumptions and constraints that affect the Architecture.

9. **Risks and issues;** Proactively document and manage risks and issues, both processes as well as the results.

10. **Plan;** Have a clear and sensible plan/roadmap to achieve the desired business outcome(s).

The reason why we have (or better) need an architecture is to manage risk and cost.

Using architecture frameworks like the Open Group's Architecture Framework (TOGAF 9), Capgemini's Integrated Architecture Framework (IAF) as well as getting help and support from the various Communities, ensures that our architects can deliver JIT-JEA.

4.2 TECHNICAL DEBT

In 1992, Ward Cunningham coined the term "technical debt" in a **report** that drew an analogy between shipping the first release of a software product and incurring a level of debt. This implies that any necessary rework to optimize a code would result in an additional cost or a "debt", which would grow over time. **As with credit-card debt – which typically carries interest – if ignored, technical debt increases over time, accumulating interest on the principal sum and making it increasingly difficult to reduce.** From an architectural perspective, technical debt could ultimately lead to "Ball of Mud" systems that encapsulate the consequence of carrying and building on this debt.

Technical debt, however, can also be a positive thing to incur, as sometimes developing and implementing something with a quick-and-dirty mindset might be the right thing to do. However, typically technical debt refers to clear limitations and risks which can include:

- Longer time-to-market and lack of business agility
- Development inefficiencies – lack of reuse, code replication, and poor documentation
- Increasing complexity
- Higher risk of security breaches
- Increased total cost of ownership (TCO)
- Brand impact – due to lower levels of service availability and resilience

Figure 2: https://vincentdnl.com/drawings/

Addressing technical debt can be more complex than it sounds; as we noted in our annual TechnoVision Trend series, "few organizations master the art of systematic application rationalization. Many IT experts know how to build new systems, few know how to decommission **them**." One of the key challenges is to quantify how much technical debt an organization carries, since establishing the impact and attributing a value to the amount of debt is not straightforward. In a post "Don't let technical debt stop innovation" we suggested an approach to measuring technical debt.

Once the level of debt has been identified we can address it by rationalizing and modernizing the application landscape. And this requires the mindset of a specialized 'tidying up' **guru**. First, it's a matter of commitment, aligning the need for "decluttering" throughout the organization and envisioning the benefits of what an adjusted and more flexible applications' portfolio can bring.

To start any application rationalization and modernization project having a detailed and comprehensive understanding of the current landscape. Particularity with environments that have been running for many years, that are not subject to much change anymore, and which include older technologies (sometimes also referred to as legacy), developing a good enough understanding of the existing landscape is critical.

"Platformification"

Architecture as well as new platform technologies can hold the key to addressing technical debt. There are two approaches to building a platform. As outlined in our World Banking Report, **2020**, a Greenfield and a Brownfield approach is possible. In a build context, a Greenfield approach is where a new platform is built and spun off from existing, traditional models. A brownfield approach, meanwhile, takes a different route as the approach enables organizations to transform their current pipeline model through significant legacy infrastructure enhancements and upgrades to IT capabilities.

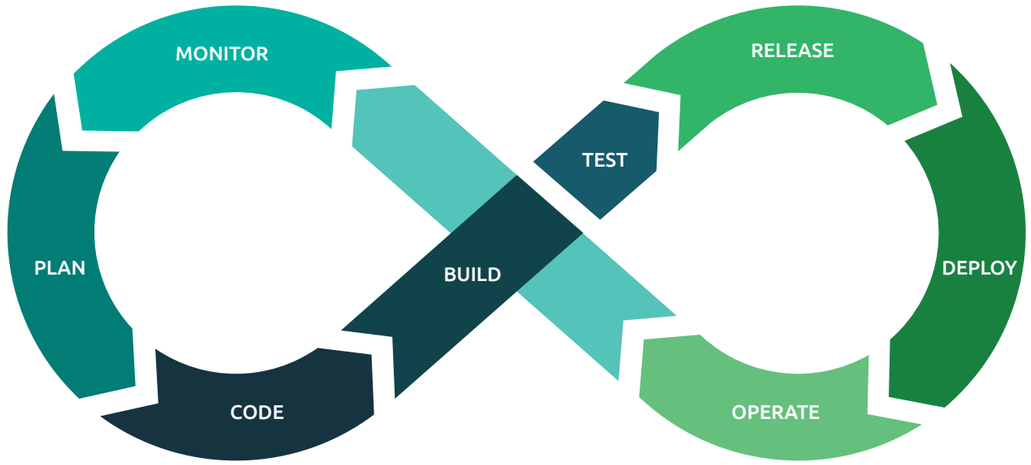
In a Buy context, an organization can acquire the platforms it needs to modernize and, ultimately, address technical debt. Taking this approach accelerates the pace of change, but also comes with the potential for significant levels of resistance due to the level of change needed to ensure **success**.

02
return to table of contents

There is no "single cause" that creates and subsequently grows technical debt. Therefore, any approach addressing technical debt must be multi-faceted. On occasion, a "perfect storm" of events manifests itself to create a compelling reason to act.

4.3 WHERE TO NEXT WITH DEVOPS?

The founding philosophy behind DevOps embodies the principles of closer collaboration, transparent communications, small-batch iterations, feedback loops, continuous learning, and ongoing improvement. As an organization embraces agile and the new ways of working it brings, it must also embrace the cultural shifts associated with these new concepts.

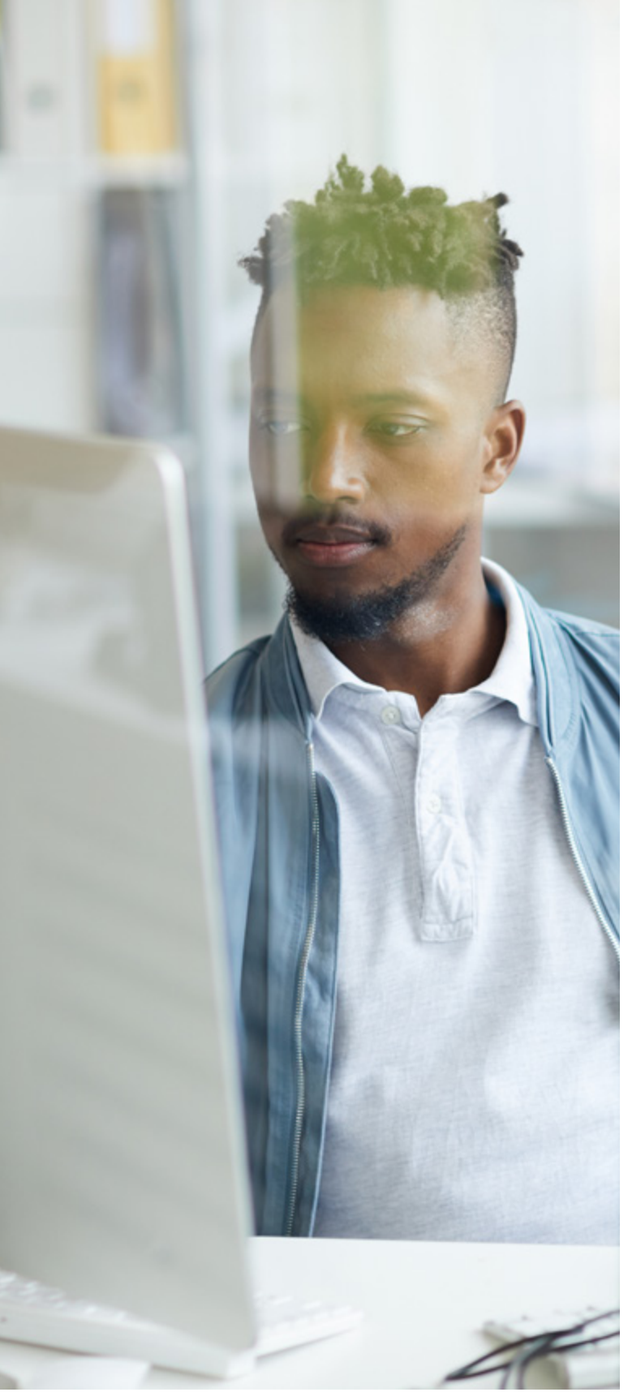


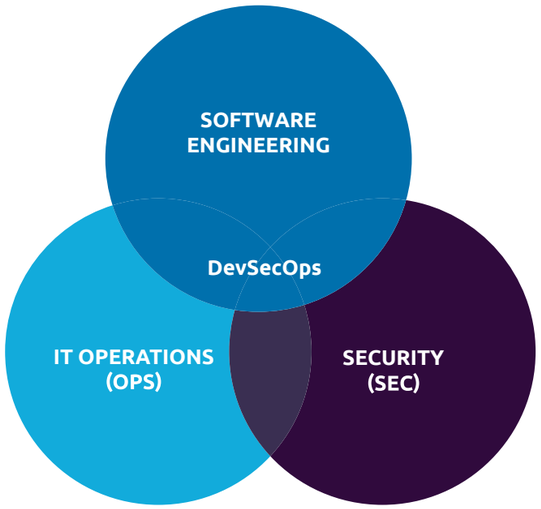
A survey, as part of the State of DevOps Report **2020** by Puppet and CircleCI, found that organizations could achieve greater success by adopting DevOps. They could "…enact deeper structural changes..." and support DevOps enablement.

Specifically, two areas of structural change are commonly focused on by these organizations. These are a platform- and product-focused approach to software delivery, in addition to applying DevOps principles to change management processes to ensure success.

Platform models and product-focused organizations provide capabilities that enable product teams and DevOps teams to better perform their roles and execute responsibilities. Furthermore, giving product end-to-end delivery responsibility for a feature or business outcome is a good delivery model when dealing with a few products. However, this model does not scale in large organizations that have hundreds of products. After all, large numbers of product teams, addressing similar problems, while evaluating technologies, integrating them, and maintaining valuable infrastructure can find it to be both inefficient and expensive to maintain.

Another aspect of how organizations have reached the higher levels of DevOps maturity relates to extending DevOps to other capabilities. One example is security. To reflect this, **DevSecOps** has become the latest abbreviation. But at a practical level, it means you can embed security into the full software development life cycle (SDLC), and that you can leverage automation here to validate the necessary security requirements.





We cannot achieve these capabilities easily, however. In our Global DevSecOps Insight Report **2020**, we noted that highly mature organizations indicate that lack of time, lack of standardization, and onboarding remain their top challenges in realizing higher levels of maturity in DevOps and DevSecOps.

Considering you cannot add security to regulated industries like the financial sector after product development is complete, DevSecOps is becoming increasingly prevalent.

Two themes are gaining traction across the industry which suggest an evolution of DevOps (though many would argue that these are inherent in the original idea of DevOps):

1) Product Centricity – organizations are adopting product-centric teams which integrate fully business and IT teams across the entire business product lifecycle including product design, build, operations, etc. This is where teams are ultimately rewarded not (only) on the quality of the software, but the success of the business product in the marketplace.

2) BizOps – a recent development that brings together the practices of agile, DevOps with the power of data analytics and full automation to enable teams and their product owners to better decide on product feature evolution and increase delivery velocity, through new automation solutions including no/low-code, production self-diagnosis/heal, and security testing.

02
return to table of contents

05 JOURNEY

The transition towards an agile model and the process of implementing and scaling across the organization brings many challenges with it. Over the years Capgemini has observed the following hurdles when it comes to moving from a traditional and waterfall-based software development approach to an agile one:

Focusing on technology only; seeing agility = DevOps tool chain

- Some organizations believe that to increase agility all they must do is deploy a CI/CD toolset or other DevOps-related aspects.

Ignoring organizational and governance change

- Agility must impact the organizational structure – and not just how IT and the business work together, it "mustalso" consider how 3rd parties fit into the structure.

Lack of Leadership

- An agile implementation (which aims to transform an enterprise into becoming more agile) will require bottom-up and top-down engagement. Therefore, executive sponsorship is a must.

The starting point is undefined

- An agile transformation is a major transformation program that has a start, an end, and a plan for getting from A to B. Not having a sufficient understanding of where the starting point for this transformation can impact a successful deployment .

Ignoring Brownfield/traditional approaches

- Focusing on a cloud-native or Greenfield approach will support applications that only run on new technology – what about the apps that also have some traditional technology?

Setting up a digital silo

- It can be tempting to simply set up a separate digital unit that runs fully on agile; however, this develops a new silo that can impact and hamper wider adoption.

In this paper, we have covered areas in the people, process, and technology domains that are crucial in improving the agile software development cycle. However, switching to an agile way of developing software must go together with an enterprise-wide move towards agile. To achieve this, we suggest considering the following key steps to ensure success:

Prepare: Establish the agile development mandate

- Organizations should establish an agile Center of Excellence (CoE) whose first task is to create the case for change and communicate the compelling reason for moving to agile across the enterprise. Ensure that not just the SDLC but also the architecture capability has "agility" built in as standard.

Experiment: Start with customer-focused initiatives and scale gradually

- Agile market leaders start small, but they start with the areas closest to the customer – either a flagship customer journey or customer service. In areas prone to agile ways of working, with a strong customer focus, you emphasize innovation and rapid action over perfectionism.

Orient: Change culture by changing behaviors and developing Pi-shaped-skills

- In our previous global research on digital transformation, Capgemini found that 82% of organizations cited "cultural issues as significant roadblocks to achieving their **aim**." Enterprise-wide adoption of a new and agile culture requires a significant behavior change at all levels. Therefore, to create an agile culture, organizations need to engage, empower, and inspire employees in new ways of working.

Govern: Link agile portfolio planning and operations with business strategy

- Capgemini's recent research on digital mastery found that around 68% of "digital masters" (those organizations with a high digital maturity) constantly look for innovations in their value chains and 65% test promising ideas quickly . This focus on innovation, speed, and the customer experience require organizations to reset the way they govern their strategic portfolios. These, in turn, we should be able to align this with a clear vision, objectives, roadmap, and overall organizational strategy.

Accelerate: Modernize IT with DevOps, cloud, and microservices

- Digital giants from Amazon to Google deploy software thousands of times a day. According to **TechBeacon**, they are "10 companies killing it at DevOps." More than just digital-native companies require this sort of technical prowess – it is also critical to anyone operating in the experience economy. **Walmart**, for example, leverages DevOps to make 170,000 improvements to its software every month. This is not just about speed, it's also about improved quality, lower failure rates, and more efficiency. To prepare for the new economic era – where new data and ecosystems will replace industry sectors, becoming the new currency – organizations need to modernize their monolithic systems if they want to enter new collaborations.

It is essential to take a simultaneous top-down and bottom-up approach to agile process implementation. Only then can you establish the right attitude for collaboration between senior management (which defines the culture of the whole organization) and lower hierarchies (which need to adapt to new ways of working). Along the way, board members must act as change agents and fully support these new ways of working, e.g., leading by example.

To make this journey a resounding success, we recommend starting small and celebrating initial successes before expanding this agile approach across any company.

return to table of contents

AWS example: How Financial Services Institutions are Accelerating and Implementing Machine Learning within their cloud strategies

Baran Karlidag – Manager, Partner Solutions Architecture, AWS

Machine learning (ML) adoption is accelerating in all types and sizes of financial institutions. Multiple drivers contribute to this acceleration: evolving customer experience, personalized recommendations, fintech disruption, new cyber threats, open finance initiatives, portfolio optimization, and increasing data volumes, all drive organizations to experiment with ML.

> **Artificial Intelligence is the broader concept of machines being able to carry out tasks in a way that we would consider "smart". Machine Learning is the application of AI based around the idea that we should really just be able to give machines access to data and let them learn for themselves.**

Gartner predicts that, by the end of 2024, 75% of the organizations will shift from piloting to operationalizing artificial **intelligence**. In Financial Services, a rise in data volumes, an increasing need for cost-effective analytics to enable AI/ML and influence critical business applications and the adherence to regulatory and compliance requirements further add to the challenges.

The strategies described below aim to help Financial Institutions address these challenges for ML teams driving innovation and IT teams providing the platforms and supporting production workloads:

1. Use Repositories for your models and features: Using central repositories to store ML models and features will increase agility and strengthen the governance of an organization. You must capture and store information related to ML workflows to meet audit and compliance requirements. With many ML models in production, a model registry will help organizations effectively manage models and related information and employ continuous integration, continuous delivery (CI/CD) practices.

2. Automate: Automating end-to-end workflows through CI/CD pipelines will enable faster iterations in ML development and reduce errors. Model builders often do not have the time and skills to create CI/CD pipelines, so you can introduce manual activities to the process which is error-prone and slows things down. Automating end-to-end ML workflows at scale helps Financial Services organizations rapidly build new models and evolve existing ones.

3. Use Consistent Tooling Across the Entire Lifecycle: Using tools from different vendors for different workflow stages hinders development. Maintaining the tools, integrating them, training teams, and ensuring security standards for tools create unnecessary overhead. Using a managed service that covers the end-to-end ML lifecycle, from data preparation to managing the production environments, will increase organizations' ML adoption significantly.

4. Benefit from the Agility of the Cloud: ML is an iterative process; a high degree of experimentation is needed to find the best solution for the business problem at hand. This involves experimenting with different technologies using different types of hardware, such as GPUs, which often leads to computational requirements that cannot be met by existing capacity. This adds additional complexity and cost to try and solve. Using the cloud for ML, financial services organizations can access ML technologies optimized for cloud, large-scale infrastructure, cost-efficient on-demand consumption models, and managed services that can scale easily with business needs.

02

return to table of contents

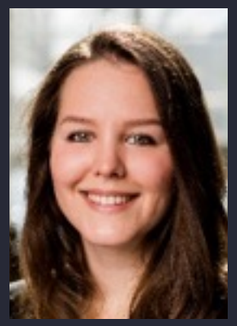
AUTHORS

Gunnar Menzel

VP& Chief Technology & Innovation Officer North & Central Europe

gunnar.menzel@capgemini.com

Gunnar is the Chief Technology & Innovation Officer for Capgemini Europe North and Central. He is a Master Certified Architect, is one of the lead Authors in Capgemini's Technovision Trend as well as is a founding member of Capgemini's Group DevOps Advisory Board. Over the past thirty years, Gunnar has successfully worked with many Organizations to either transforming to an agile model and/or applying agile mindsets and tools, to drive speed, quality, and value for money.



Sara Montonen

Manager, Enterprise Transformation

sara.montonen@capgemini.com

Sara is a Manager in Capgemini Invent's Enterprise Transformation practice. She has primarily worked with clients in the financial services, automotive, and telco industries. Sara has supported our clients in projects ranging from strategy to implementation. She has experience in, for instance, operating model transformations, regulatory implementation, process optimization, market launches, and change management and communication.

Mark K Ashton

Director and Enterprise Architect – Financial Services

mark.ashton@capgemini.com

Mark is an Enterprise Architect in the Capgemini Financial Services business unit. He is a Capgemini certified architect and a strong advocate of developing the architect profession and community.
As an Enterprise Architect, Mark has worked with many large global Financial Services clients, advising, guiding, and delivering to them on some of their most challenging and complex digital, integration and cloud transformation initiatives.

Baran Karlidag

Partner Solutions Architect Manager

karlidag@amazon.co.uk

Baran is a Partner Solutions Architect Lead at Amazon Web Services, managing a worldwide teams industry. With a focus on Financial Services, Baran works closely with AWS Premier partners to help Financial Institutions on their digital transformation journey. Before joining AWS, Baran held various senior technology and leadership roles at HSBC, Royal Bank of Scotland, ABN AMRO, and Reuters. Baran has an MSc in Electrical and Electronics Engineering.

Mark Standeaven

EVP & Agile ADM Delivery Executive Insurance

mark.standeaven@capgemini.com

Mark is an Executive VP with the Capgemini Insurance business and has over 30 years of experience in the IT industry with roles ranging from software engineer through to global transformation leader. He has worked across various industry sectors, including retail, automotive, and finance. As part of his transformation role, he has helped clients across Europe and the US increase their overall IT and business agility through delivery innovation focused on the use of agile, lean, and DevOps as well as transforming enterprise operating models to sustainably optimize cost while increasing feature value to end customers.drive speed, quality, and value for money.

Clemens Reijnen

VP & Global CTO Cloud Sogeti and DevOps leader

clemens.reijnen@sogeti.com

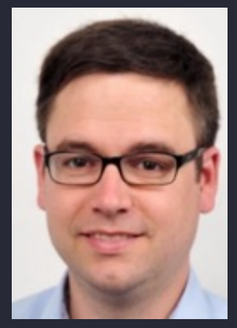
Clemens is Sogeti's Global CTO Cloud Services and DevOps leader. He has been awarded the Microsoft Most Professional Award for 10 years in a row and is a Sogeti Labs Technical Fellow. As a global DevOps leader, he works closely with Sogeti's large enterprise customers to ensure their cloud adoption and Enterprise DevOps transformation programs create value for the business.

David Aline

VP & Application Management Services COE Leader Europe

david.aline@capgemini.com

David joined Capgemini in 2015 to lead major deals like the PSA HERMES pursuit which resulted in the largest win at that time for Capgemini France. In 2016, He took the lead of the French ADM Business Development team and since 2019, leads the European AMS COE, managing presales activity generating nearly 2B€ booking. Having had numerous customer interactions, his current focus is on the transformation of major European accounts from traditional to Agile@Scale working methods.



Marc Bauer

Delivery Architect Director, Center of Excellence Agile & DevOps

marc.bauer@capgemini.com

Marc is helping clients across Europe in their transformation towards Agile, DevOps and Product-centric organizational structures. Being an software architect and a certified Professional Scrum Trainer from scrum.org since more than 10 years, he trained teams across the globe in agile software engineering, self-organization and software craftmenship. He is passionate about building engineering cultures which put the users and value in the focus.

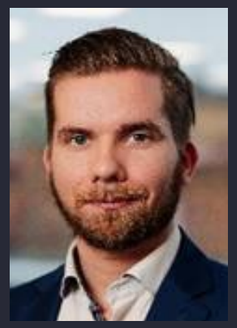
return to table of contents

Mark Oost

Global CTO Analytics & AI at Sogeti

mark.oost@sogeti.com

Mark has over 10 years of experience within AI and Analytics. Before joining the Group, Mark was responsible for AI within Sogeti Netherlands where he was responsible for the development of the team and business as well as the AI Strategy. Before that he worked as a (Lead) Data Scientist for Sogeti, Teradata, and Experian. Where he worked for clients from multiple markets internationally on technologies around AI, Deep Learning, and Machine Learning.



Henrik Dahlman

Senior Consultant, Enterprise Transformation

henrik.dahlman@capgemini.com

Henrik is a Senior Consultant in Capgemini Invent's Enterprise Transformation practice. He has primarily worked with clients in the financial services, banking, and public sector. Henrik has supported our clients in projects ranging from strategy to implementation and he has international experience from, process optimization & harmonization, implementing an agile way of working and in change and project management.

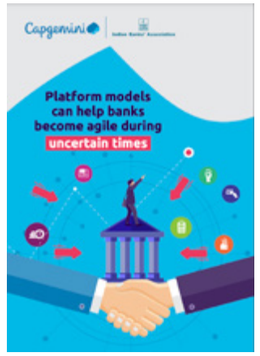
Special Thanks go to:

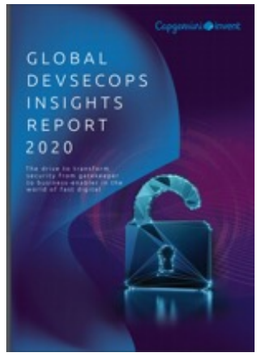
Pintu Patel, Jerzy Szwarc, Fabian Weber, Dr Karl Prott, Andreas Lutz and of course to Philip Ras. Also a Special Thanks to Stijn Follet who provided invaluable feedback

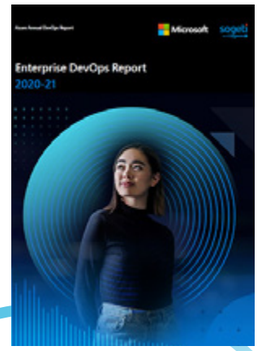
02

return to table of contents

FURTHER READINGS







REFERENCES

01 Bottomley, J. (Nov. 2019) Banking in the Future. https://www.hsbc.com/insight/topics/banking-in-the-future

02 West, D. (March 2018) CIO HSBC. https://www.bcg.com/en-gb/publications/2018/transforming-business-through-technology-interview-darryl-west-group-cio-hsbc

03 Buvat, Evans, Manchanda, Rachlin, Sullivan, Slatter. (Mar. 2019) Where are banks and insurers on their digital mastery journey? https://www.capgemini.com/gb-en/wp-content/uploads/sites/3/2019/03/30-min-–-Report.pdf

04 Tolido, Menzel, Hessler. (Jan. 2021) TechnoVision – Change Making. https://www.capgemini.com/wp-content/uploads/2021/01/TechnoVision-Change-Making-2021.pdf

05 Steinacker, Coppola, Mahapatra, Scheunert, Levaux, Welde, Folgerø, Tomovich, Lee, Ruh, Chaniot. (Mar. 2019) Digital Transformation Review – Twelfth Edition https://www.capgemini.com/gb-en/wp-content/uploads/sites/3/2019/02/Download-%E2%80%93-Digital-Transformation-Review-12.pdf

06 Menzel, Macaulay. (Mar. 2016) DevOps - The Future of Application Lifecycle Automation. https://www.capgemini.com/de-de/wp-content/uploads/sites/5/2016/03/devops-the-future-of-application-lifecycle-automation.pdf

07 V-Model (software development) https://en.wikipedia.org/wiki/V-Model_(software_development)

08 Alleau, Kumar, Ng. (Sep. 2020) Global DevSecOps Insights Report. https://www.capgemini.com/fi-en/wp-content/uploads/sites/27/2020/09/Global-DevSecOps-Insight-Report-2020.pdf

09 Kroll, Boeing, Schmidt, Vogg, Thöle, Lengfeld, Rauch. (May 2017) Agile Organizations. https://www.capgemini.com/consultingde/wpcontent/uploads/sites/32/2017/08/cc_agile_organization_pov_20170508.pdf

10 Jérôme Bourgeais and Jérôme Dejardin (Apr. 2019) https://www.capgemini.com/wp-content/uploads/2019/04/Transformation-agile.pdf

11 Greverie, Menzel, Kessel-Fell, Harrington, Chichowlas, Crummenerl, Follet, Buvat. (Nov. 2019) Agile at Scale. https://www.capgemini.com/wp-content/uploads/2019/11/Report-%E2%80%93-Agile-@-Scale-1.pdf

12 Donnelly, C. (June 2018) Barclays banks on DevOps to support 'all-in' move to AWS public cloud. https://www.computerweekly.com/news/252443653/Barclays-banks-on-DevOps-to-support-all-in-move-to-AWS-public-cloud

13 Edmondson, A. (June 1999) Psychological Safety and Learning Behavior in Teams. https://www.jstor.org/stable/2666999?seq=1

14 Capgemini (May 2019) Eneco and Capgemini Sharing Mutual Goals https://youtu.be/QuYpB6eAP-A

15 Cooper, Klaas-Jan (September 2018) Adopting InnerSource. https://www.oreilly.com/library/view/adopting-innersource/9781492041863/

16 Watson, Jagoda, Pavangadkar, King. (Jan. 22, 2018) Introduction to Innersource. https://resources.github.com/whitepapers/introduction-to-innersource/

17 Balestra, P. (June 2021) Spotify Engineering. https://engineering.atspotify.com/

18 The Netflix Tech Blog. https://netflixtechblog.com/

19 Kniberg, Ivarsson (Oct. 2012) Spotify Scaling. https://blog.crisp.se/wp-content/uploads/2012/11/SpotifyScaling.pdf

20 Bunning R. (Sept. 2020) ING improves on the so-called Spotify Model using LeSS. https://www.scrumwithstyle.com/ing-improves-on-the-so-called-spotify-model-using-less/

21 Otto GmbH. https://en.wikipedia.org/wiki/Otto_GmbH

22 Periodic Table of DevOps Tools. https://digital.ai/periodic-table-of-devops-tools

23 INGKA Technology Radar. https://techradar.ingka.com/

24 Zalando Tech Radar. https://opensource.zalando.com/tech-radar/

25 Agile Alliance. The Dojo – Implementing an Immersive Learning Environment for Teams. https://www.agilealliance.org/resources/sessions/the-dojo-implementing-an-immersive-learning-environment-for-teams/

26 Alger, L. (June 2018) DevOps Dojo – The Path to Capital One's Success. https://www.devopsonline.co.uk/devops-dojo-the-path-to-capital-ones-success/

27 Horta-Osório, A. (Feb. 2018) LLyods – Strategic Update. https://www.lloydsbankinggroup.com/assets/pdfs/who-we-are/our-strategy/strategy-downloads/2018/2018-lbg-strategic-update-booklet.pdf

28 Hern, A. (Apr. 2018) The Two-Pizza Rule and the Secret of Amazon's Success. https://www.theguardian.com/technology/2018/apr/24/the-two-pizza-rule-and-the-secret-of-amazons-success

29 Li, Menzel, Defaut. The Automation Advantage. https://www.capgemini.com/gb-en/service/cloud-services/the-automation-advantage/

30 Murphy, Looney, Kacirek. Automation at Google. https://sre.google/sre-book/automation-at-google/

31 (March 2021) DevSecOps in Regulated Industries. https://www.capgemini.com/wp-content/uploads/2021/04/DevSecOps-in-Regulated-Industries_March-2021.pdf

return to table of contents

32 Ahmed, Pradhan, Remanan, Chakraborty. (2021) TechnoVision – Change Making. https://www.capgemini.com/wp-content/uploads/2021/04/TechnoVisionFS2021-Final.pdf

33 Fowler, M. (April 2020) DomainDrivenDesign. https://martinfowler.com/bliki/DomainDrivenDesign.html

34 Buvat, Evans, Manchanda, Rachlin, Sullivan, Slatter. (Mar. 2019) Where are banks and insurers on their digital mastery journey? https://www.capgemini.com/wp-content/uploads/2019/03/30-min-%E2%80%93-Report.pdf

35 Li, Menzel, Defaut. The Automation Advantage – Cloud Services. https://www.capgemini.com/gb-en/service/cloud-services/the-automation-advantage/

36 Telling, M. (Aug. 2018) KeyBank: An AIOps Success Story. https://www.moogsoft.com/blog/aiops/keybank-aiops-success-story

37 Open Agile Architecture. https://pubs.opengroup.org/architecture/o-aa-standard/

38 Menzel, G. (July 2020) Sustainability in ICT; It is Time to Take it Seriously. https://www.linkedin.com/pulse/sustainability-ict-time-take-seriously-gunnar-menzel/

39 Technical Debt. https://en.wikipedia.org/wiki/Technical_debt

40 Menzel, G. (May 2021) Don't let technical debt stop innovation. https://www.linkedin.com/pulse/dont-let-technical-debt-stop-innovation-gunnar-menzel/

41 World Retail Banking 2020. (June 2020) https://www.capgemini.com/gb-en/news/world-retail-banking-report-2020

42 Gyanchandani, Saraf, Manchiraju, Pillai (Mar. 2021) Platform Models Can Help Become Agile During Uncertain Times. https://www.capgemini.com/wp-content/uploads/2021/03/Platform-models-can-help-banks-become-agile-during-uncertain-time.pdf

43 (2020) State of DevOps Report. https://puppet.com/resources/report/2020-state-of-devops-report/

44 Bonnet, D. (July 2018) Capgemini Research Institute, "Understanding digital mastery today – Why companies are struggling with their digital transformations. https://www.capgemini.com/resources/understanding-digital-mastery-today/

45 Null, C. (2020) TechBeacon – 10 companies killing it at DevOps. https://techbeacon.com/devops/10-companies-killing-it-devops

46 Schwartz, S. (March 2018) Walmart relies on DevOps and the IoT to better its business model. https://www.retaildive.com/news/walmart-relies-on-devops-and-the-iot-to-better-its-business-model/518604/

47 (June 2020) Gartner Identifies Top 10 Data and Analytics Technologies. https://www.gartner.com/en/newsroom/press-releases/2020-06-22-gartner-identifies-top-10-data-and-analytics-technolo

48 Natu, Ping, Vasilakakis, Huang, Jeyasingh, (July 2020) Machine Learning Good Practices in Financial Services. https://d1.awsstatic.com/whitepapers/machine-learning-in-financial-services-on-aws.pdf

return to table of contents

Capgemini

About Capgemini

Capgemini is a global leader in partnering with companies to transform and manage their business by harnessing the power of technology. The Group is guided everyday by its purpose of unleashing human energy through technology for an inclusive and sustainable future. It is a responsible and diverse organization of 270,000 team members in nearly 50 countries. With its strong 50 year heritage and deep industry expertise, Capgemini is trusted by its clients to address the entire breadth of their business needs, from strategy and design to operations, fueled by the fast evolving and innovative world of cloud, data, AI, connectivity, software, digital engineering and platforms. The Group reported in 2020 global revenues of €16 billion.

Learn more about us at

www.capgemini.com

GET THE FUTURE YOU WANT

MACS_2021_3592