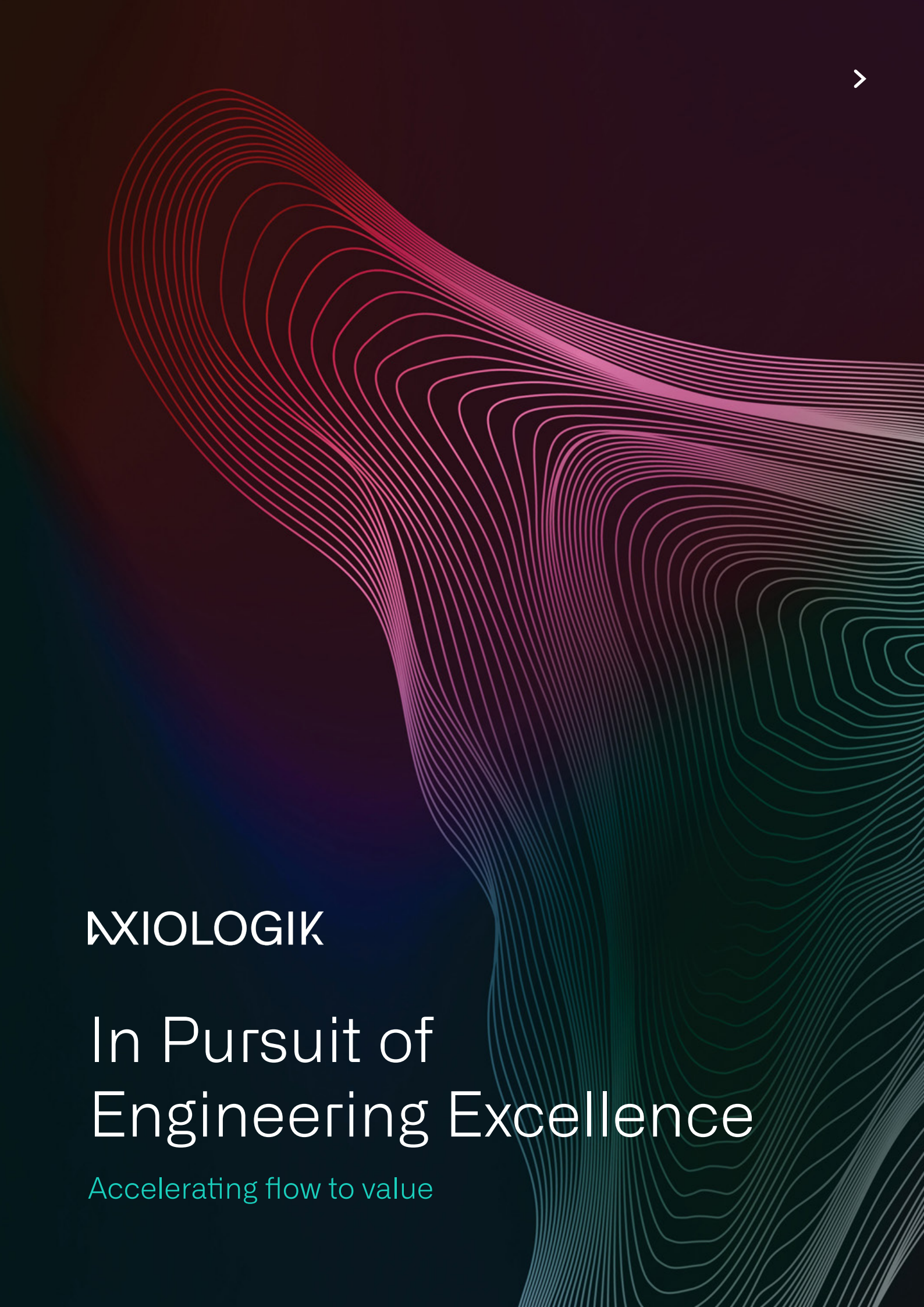

AXIOLOGIK
In Pursuit of Engineering Excellence
Accelerating flow to value

Organisations that invest in the “how” of software delivery (as well as the “what”) outperform those that don’t. According to Accelerate (Frosgre, Humble and Kim 2018), the highest performing organisations are twice as likely to exceed organisational goals – profitability and market share – as those that don’t. They are also twice as likely to exceed qualitative goals, such as customer satisfaction and positive customer experience.

This paper looks at the challenges organisations face when investing in the “how”. It provides guidance on proven, practical steps to drive tangible improvements.

Contents

Optimising the “how”

Historically, organisations have focused on the “what” of software delivery, but focusing also on “how” will drive outstanding results.

Business performance is proven to be directly correlated with an organisation’s ability to deliver technology change. Despite this, the majority of organisations fail to invest in the “how” of software delivery, choosing by default to focus solely on the “what”. They are missing a significant opportunity.

Effective software delivery processes ensure that products meet market demands, improve reliability, and maintain quality, which enhances customer satisfaction and retention. By focusing on the “how,” organisations can streamline workflows, reduce technical debt, and ensure a more predictable delivery schedule, all of which help to avoid costly rework and minimise risks.

Importantly, investing in optimised software delivery practices is a virtuous circle. It empowers teams to quickly respond to changing requirements, leading to greater flexibility and innovation. It also fosters a collaborative culture, breaking down silos between development, operations, and quality assurance teams, which in turn reduces bottlenecks and increases overall productivity.

This paper

Achieving engineering excellence involves optimising teams to maximise value delivery (fast flow); optimising the eco-system in which product teams fit; and cultivating a culture that focuses on ownership, accountability, and relentless, continuous improvement.

This paper looks at these topic areas providing practical guidance for improvement.

The Challenge

Despite years of improvement, organisations are still constrained by a myriad of legacy issues.

At the turn of the millennium, Agile methodologies began to rise to prominence with the promise of optimised delivery value streams driving fast flow to value. Early works including Kent Beck's seminal book, XP Explained, outlined approaches to iterative, incremental development, supported by modern engineering techniques (today referred to as DevOps).

The "Agile" revolution saw the breakdown of traditional silos, and the end of the dominance of the Design-Build-Run approach to software delivery. And as approaches matured, organisations promoted experimentation, shorter feedback loops, and incremental value delivery, more recently expanding further to consider security verification and scanning (DevSecOps).

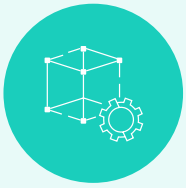
While the intention was the rhythmic cadence of early and continuous delivery of software, organisations typically optimised for a localised improvement within the "delivery" phases of the value stream (i.e. "Design" through to "Build") and did not close the gaps within the wider business. This led many to fall into an antipattern whereby they wrapped this new Agile thinking inside of their Waterfall methodology. Instead of daily or hourly releases, cycle times remained in weeks.

Even forward-thinking organisations still retain siloed delivery units and an "us and them" view of Operations. According to the DORA 2023 report, the majority of delivery organisations are less than 50% efficient, and those of high maturity, only achieve between 60 and 80% efficiency.

In this section, we look at why this is the case, discussing the challenges organisation face when optimising delivery for fast flow:

Organisation and team interactions

Page 6 >



Technical architecture

Page 7 >

Tools and frameworks

Page 7 >

The need for control – governance, risk, compliance, and internal audit

Page 8 >

Organisation and Team Interactions

Fast flow is reliant upon dynamic organisations that empower staff and reduce bottlenecks. Although Agile, DevOps, and DevSecOps aimed to break down organisational silos and encourage greater collaboration, inefficient designs remain that constrain flow.

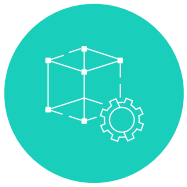
The most common challenge is that of interdependency, a prevailing constraint in complex organisations. Interactions become convoluted and time-consuming when teams are reliant on others; engineers have less autonomy, feel less empowered, and have a lower sense of purpose; cognitive load is often greater; and the risk to service delivery is increased when making changes.

Communication challenges are amplified through poor culture - as organisations and as individuals we have a set of values that guide us and influence the way we interact. These inform the way that we create and share knowledge, and become evident in the way we cooperate in the working environment.

Having a “bad culture” can lead to:

- A lack of trust.
- An environment that is not conducive to experimentation and learning, which results in a reduction in software delivery performance – there is huge value in normalising “failure” and encouraging inquiry to drive continual improvements.
- A reduction in job satisfaction where developers feel they are not making good use of their skills and abilities – which leads to a lack of pride and a reduction in quality.
- Unclear ownership for production services and product outcomes.

A negative culture will naturally lead to resistance to change. The “not built here” mentality can degrade the working environment and its ability to make the necessary changes that support continual improvement.



Technical Architecture

Just as a legacy platform will constrain business agility, badly architected systems will stymie flow.

Conway's Law (1967) identified that "organisations, that design systems, are constrained to produce designs that mimic the communication structures of these organisations".

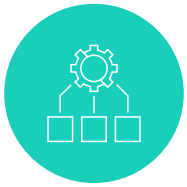
For example, small, distributed teams are more likely to produce a modular, loosely coupled, microservice architecture. Contrast this to large, collocated teams that are likely to develop a tightly coupled monolithic architecture.

Technical architecture can have a significant impact on the ability for teams to deliver value efficiently. Tight coupling between components introduces dependencies between teams – this can be observed by watching PMOs and traditional project managers coordinating a schedule of coding changes across multiple teams, creating complex joint release and deployment plans.

Even loosely-coupled architectures can be negated through poor implementations – for example, efforts to develop a microservices architecture may inadvertently result in a "distributed monolith" if clear boundaries of both form and function are not in place.

Further examples include "Lambda pinball" where the domain logic is lost across multiple serverless functions designed around buckets, queues, and a complex graph of cloud services.

Finally, challenges may exist because of the underlying platforms, including network and security, such as those associated with managing on-premise, cloud, and hybrid infrastructure environments. While infrastructure-as-code techniques often attempt to treat the environment like applications, organisations may struggle with patterns for ownership and embedded security controls (which we shall come to later).



Tools and Frameworks

The correct tooling is essential to optimise flow, but getting it wrong can lead to process bottlenecks and cognitive load.

Agile and DevOps adoption has seen an increase in the landscape of COTS and SaaS platforms that seek to enable an accelerated engineering environment - there is an entire industry dedicated to business agility and automation providing a plethora of choices spanning the entire software delivery lifecycle.

In an engineering environment, the topic of tools and frameworks is not just limited to architectural design decisions and the centralisation/decentralisation of core platforms. The developer experience begins with the local development environment and the tools that the engineering team will spend most of their time working within, such as their IDE and associated frameworks. Navigating this experience can sometimes be fraught with challenges.

Organisations can face issues ensuring these tools and platforms integrate (backend) and provide a cohesive user experience for the internal teams (frontend). This problem manifests not only between the teams, such as a work item being traceable through source code, build, deployment and ITSM toolkits, but with consistency in overall experience.

Some organisations may opt for a single vendor tooling solution, whereas others may pick best-of-breed for each use-case and glue the tools together later. Each approach has its pros and cons, such as having access to fewer power-user features or having to manage a complex environment of tools spanning technological and business processes.

Alternatively, organisations may allow the teams the autonomy to pick and choose their own toolchains – trading the advantages of enterprise standardisation against the localised decision-making power that may help some functions to optimise flow locally.

Governance and the need for control

Getting the balance right between control and agility is critical to fast flow.

Change governance is essential for the successful management of technology. Organisations need to be confident change is secure, will not cause failure and has not introduced trojan code. If issues do arise, it must be possible to quickly understand the cause and impact, and take immediate remedial action.

In highly regulated environments, change is a compliance issue. It is necessary to be able to audit the changes that have been made, understand the systems impacted, and ensure that all required checks have been successfully completed.

However, it is important that management and governance processes do not slow the pace of change unnecessarily. They should not be a hindrance or frustration to teams - in a fast-moving digital world, change cadence should be measured in minutes, hours, and days, not weeks.

Unfortunately, in many cases, governance controls are ineffective whilst also constraining the flow of change.

Firstly, those tasked with meeting control requirements and for managing risks may have conflicting incentives. For example, if there is greater emphasis on meeting financial rather than control orientated objectives, the team may feel disenfranchised from its purpose, and react negatively and objectively.

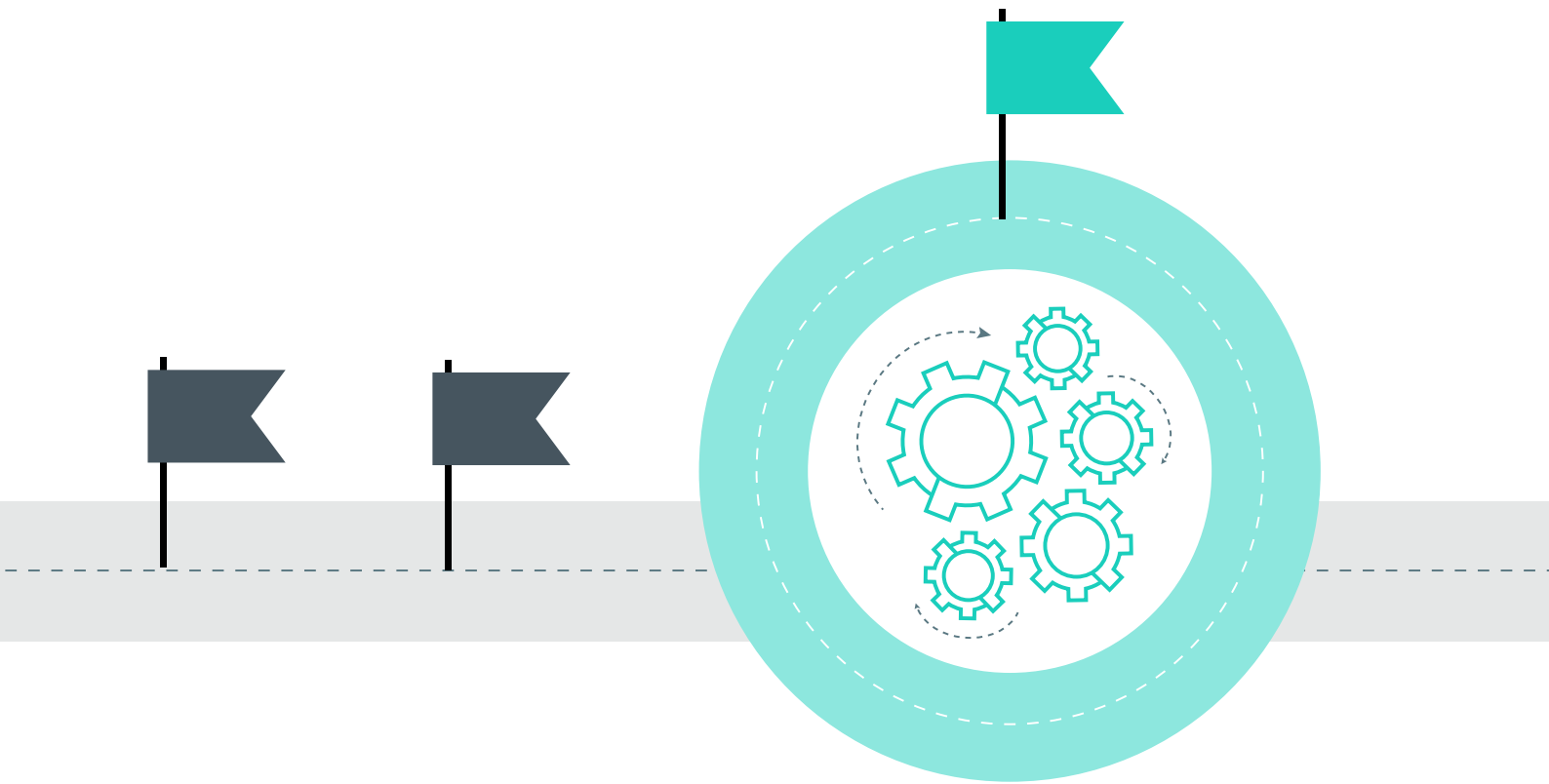
Secondly, those tasked with writing and maintaining the control requirements may not be sufficiently independent from the team that is implementing the controls. If there is a lack of organisational independence, then conflicting incentives may materialise.

For example, it is normal for quality assurance teams to sit within the software development function. While this can often work well, and is a common practice, care needs to be taken regarding the level of independence offered and whether the culture exists whereby the assurance function can guide and also openly challenge the leadership on risk-related issues.

Finally, the audit trail that evidences control requirements can be manual and time-consuming to generate, and equally time-consuming to verify. These factors combined can limit the organisation's ability to deliver sooner and safer because they are typically formed into manual gates as part of the path-to-production and release management process.

As much as we focus on "sooner", it is this space that ensures organisations do so "safely".

The ultimate aim is for organisations to achieve "just enough governance" balancing the need for control with pace.



Establishing a roadmap for engineering excellence

Having identified some of the key challenges, we can explore some of the techniques to optimise the delivery value streams. These are covered in the following sections under: people, tools, and processes.

Importantly, improvements can be applied incrementally - learning from our engineering approach, it's not a big-bang change activity.

Start by setting clear, long-term goals that reflect the organisation's vision and values. The vision should address key areas such as innovation, quality, scalability, and sustainability. Align these objectives with broader business goals to ensure engineering efforts contribute to the overall success of the organisation.

Next evaluate the current state of the engineering team processes, and technologies. Use value stream mapping (VSM) to understand the efficiency of processes and undertake a Developer Experience (DX) survey to assesses the overall satisfaction and engagement of developers within your organisation.

Finally, break down the long-term objectives into specific, measurable milestones. These milestones may include improving development cycles, increasing automation, or boosting team skills. Each milestone should have clear success metrics, such as reduced technical debt, faster release cycles, or enhanced customer satisfaction.

Engineering excellence requires a commitment to continuous learning and improvement. Encourage teams to embrace agile methodologies, conduct regular retrospectives, and foster a culture of collaboration and feedback. Provide opportunities for professional development and invest in tools that enhance productivity and quality.

People

The digital operating model is critical to enabling an environment that supports fast flow, and this is as much an architectural concern as it is a management one. Organisations should push the flow of information down into individual teams to support decision-making and in turn remove obstacles that inhibit these decisions and prevent work from being done.

Loosely coupled architecture

This allows for the creation of small, cross-functional, agile teams that are aligned to a discrete service, subsystem, or platform. These teams are themselves loosely coupled from other teams and from the rest of the organisation, and are therefore empowered to make significant changes without directly involving others.

It is important to view loosely coupled architecture as a driver to loosely coupled teams, for in this context they are describing the same thing, albeit from different perspectives. A loosely coupled design, based around microservices and event-management patterns, enables teams to code, build, and release without having any dependency on other parties, thereby avoiding potential bottlenecks – thus the team design can also be called loosely coupled.

Optimise team interactions

Collaboration between teams should be established intentionally and considered in a similar way to adopting modern technical patterns. For example, a well-described discoverable API, with a self-service onboarding process, helps break any dependencies between two components. Consider the same example now with teams – just as a team exposes their technical services through an API, CLI, or self-service, they should also publish a Team API describing clearly how they operate, communicate, and engage.

The seminal work, Team Topologies, provides an excellent reference in the context.

Effective communication channels are crucial to establishing a fast-flow environment, and it is perfectly valid to eliminate channels that are proving inefficient. Recommended techniques to achieve this include the physical office layout, for example by moving teams in relation to each other. This can enable decision-making to become fast and synchronous rather than asynchronous and slow – consider how effective the communication is between a product lead who is sat in another building to one that is sat amongst the engineering team; one model facilitates the speed of delivery and the other inhibits it.

Organisation

Establishing boundaries between teams, including physical factors such as separate building, separate floor, or separate desk workspace, will naturally restrict those teams from actively collaborating – this may have negative consequences, but in other scenarios these boundaries can be a positive enabler for fast flow, especially if applied intentionally.

For example, an enforced space (separate office) between teams can be replaced by a self-service interface and this alternative can greatly improve efficiency and flow state. Physical proximity matters, even in a highly digital and virtual world, therefore office design can make or break the effectiveness of team interactions.

Context is key, and therefore it is important that when optimising team interactions, technology leaders identify the bottlenecks in the value stream, understand their causes, and use this information when reviewing the possible impact of the software architecture, the physical office layout, the wider organisational design.

Process

Having explored some of the people-related areas that help drive fast flow, we can look at processes as part of the path-to-production.

Value Stream Mapping

Before organisations look to optimise the flow of work, they should first understand what the value stream actually entails, and a Value Stream Map is the ideal starting point.

By mapping out value streams, identifying waste, errors, rework, cycle and process times, the organisation can identify bottlenecks and focus on process improvements; recognising this is a continual exercise where change is applied incrementally.

Managing Work-in-Progress (WIP) limits

Managing WIP is an enabling constraint that helps to flush out inefficiencies in the flow. By limiting the work-in-progress, teams can identify the real bottlenecks more easily. This is described later in the context of establishing trunk-based development and Continuous Integration.

Work in small batches

One of the key points of the Agile Manifesto is to incrementally deliver value in smaller units of work – not only does this allow organisations to elicit customer feedback sooner, but it also has the added benefit of amplifying other forms of feedback such as quality and test feedback. Teams can experiment sooner and establish a culture of continual learning.

This practice has a multiplying and enabling impact on other practices – it allows for managing WIP limits, trunk-based development, and short-lived branches, for example, because these practices are simpler to adopt when the unit of work is small (and conversely, almost impossible to implement if the batches are large).

Process improvements

Continuously reflecting, learning, and implementing improvements, both big and small, is essential for creating a lean and efficient engineering function. This practice goes beyond typical agile ceremonies. As we touched upon earlier, a culture that encourages experimentation and promotes learning from “failures” will drive a ruthless drive for ongoing improvement.

Metrics and dashboards

Throughout the lifecycle, capturing and publicising key data points will allow teams to visualise the work and the associated quality measures and business outcomes.

It is important to remember that “what you measure is what you get”, it is therefore essential to be conscious of unintended consequences. An example of this was the attempt to drive down GP Waiting Times in the UK, where a target was set that GPs should see patients within 48 hours. This directive had good intentions – however, this led to GPs restricting their booking systems to a 2-day forward view, thus meeting the Government target with 100% success, yet leaving thousands of people unable to book an appointment at all.

Be careful not to measure product teams purely on throughput (story points and velocity) as this may lead teams to become feature factories that lose the link to business value and outcomes. Other unintended side effects of this approach may include an increase in defects, security vulnerabilities, a loss of service performance, and ultimately a reduction in customer satisfaction.

The same tools that operations teams use for observing health-related metrics should also be used when engineering to monitor business outcomes i.e. those that are used to capture and visualise key metrics.

Technology

Organisations that adopt a DevSecOps approach and "bake-in" both production first and security first thinking will see significant benefits in productivity, service quality, stability and resilience.

The journey of adopting Continuous Delivery can be broken down into a few small incremental practices that build upon each other to drive up engineering excellence and accelerate theengineering environment.

Everything as code

Ensuring that all code and configuration is maintained in a version control system, such as Git, is a crucial baseline for all subsequent roadmap items. This applies to all forms of text-based content, such as code and business logic, configuration, and infrastructure-as-code definitions.

As we will explore in more detail later, codifying the control policies and managing these in a version control system will enable a further step-change in an organisation's ability to bring functions such as security and compliance closer to engineering, and maximise the delivery of business value sooner and safer.

Branching strategy

Adopting trunk-based development, with short-lived feature branches allows for gating of merges into the trunk. Merges should take place at least daily, and branches should live for less than a day, and certainly no more than 3 days. This practice also focuses the team on breaking work down, facilitates the adoption and enforcement of Work-in-Progress (WIP) limits, which in turn helps prioritising tasks, limits branch sprawl, and acts as an enabler for other practices such as Continuous Integration and Continuous Delivery.

Continuous Integration

Every merge to the trunk is built, code is checked for quality issues, and unit and integration tests are run. This process allows for fast failure on the basics of engineering best practices and allows for codification of the team's coding standards. It is important that teams adopting trunk-based development and Continuous Integration rally whenever there is a build failure – restoring a healthy state is the team's highest priority. There should be no more commits or merges until the pipeline is green again.

Shift left on security

As we discussed with the history of DevSecOps, it is important to bake security scans into the pipeline – these include secret scanning, dependency analysis (SCA), static analysis security tests (SAST), and dynamic analysis security tests (DAST).

Similarly, if you focus on ensuring that "we can react to a zero-day vulnerability" you, by extension, have worked on optimising your technical processes and path-to-production that you can also use for non-security work items - if I can take this to production in less than 24 hours then I can take that to production in greater than 24 hours also.

Test and Assurance - Automation

While some forms of manual testing will remain, teams should automate the execution of functional and non-functional testing within their value stream. It is important to focus on what is inline to the path-to-production and what can be performed in parallel i.e. determine what is considered a 'stop the deployment' event, which can then be codified and established as part of the Governance Automation - we will explore later.

Continuous Delivery

With everything in place, we can move towards automated deployments with an approval gate around either the deployment or the release. Furthermore if teams can distinguish between these two events, such as by adopting Feature Flags[1], this further facilitates a move into Continuous Deployment (an often-confused term, but rarely practiced).

[1] "Feature Flags" allow for new code or features to be deployed into production that will be activated later – essentially by switching the flag from false/off to true/on. This practice enables Technology to manage when code is deployed and Product to manage when the feature is released to users.

Governance

Perversely, in many organisations governance controls do not improve service quality. They are a “tick-in-the-box” exercise designed to shift accountability from those making the change and those accountable in the event of failure. “We followed the process” is a familiar refrain when things go wrong.

When optimising delivery value streams for fast flow, governance and compliance controls should be designed to support the process and add value, not simply incur additional overhead. Teams should be trusted, and mandatory policies should be automated as far as is possible.

The goal is to create guardrails that ensure accountability, regulatory compliance, and risk management while still fostering agility, speed, and innovation. This requires a cultural shift, in which accountability for governance and compliance is delivered through shared responsibilities rather than external constraints. Teams should be encouraged to find innovative ways to meet compliance requirements efficiently and be rewarded for their efforts to streamline governance without cutting corners.

Change Management Process

To facilitate fast flow, we need a streamlined change management process that balances governance and control with decentralised decision-making. This approach empowers those closest to the product and customers to make decisions while ensuring that the necessary safeguards are in place.

This can be achieved by having stakeholders fully engaged throughout the delivery process – where no one with decision-making rights joins the party late. In addition, this can be accompanied by the removal of the CAB that historically saw multiple parties come to the table at the last minute with limited knowledge of the service or the work being released. The need for an IT Service Management team remains vital, mostly as process owners to ensure that the organisation’s controls are met and pass audit, but not to hold a technical reviewer or approval role.

Even in organisations where a distinction has been made between “Normal” and “Standard” changes, there will be opportunities to use automation to capture and provide the assurance required that implements the minimum viable change management process.

Empower Teams with Guardrails

A set of predefined governance guardrails should be used to allow delivery teams the autonomy to make decisions. These will clearly outline non-negotiable compliance and security requirements, but leave the “how” to the teams, empowering them to innovate while remaining compliant. Provide teams with tools or templates (“golden paths”) that streamline compliance tasks (e.g., security checks, privacy audits) so they can self-govern and ensure adherence without needing constant oversight.

Shift-Left on Compliance

Integrate compliance checks early in the development cycle (“shift-left”), embedding them into the daily activities of digital delivery teams. This ensures that compliance doesn’t become a late-stage hurdle. This collaborative approach ensures governance and compliance are aligned with the business’s digital delivery objectives from the outset.

Policy-as-Code

Automate governance controls, particularly in DevOps, to ensure continuous compliance through common tools and scripts (“golden paths’). For example, an inner-source library of Infrastructure-as-Code (IaC) can be validated against codified policies to ensure that infrastructure is compliant by design. Configuring policies in a version control system ensures the same rigour around review as application code and provides an audit trail for changes.

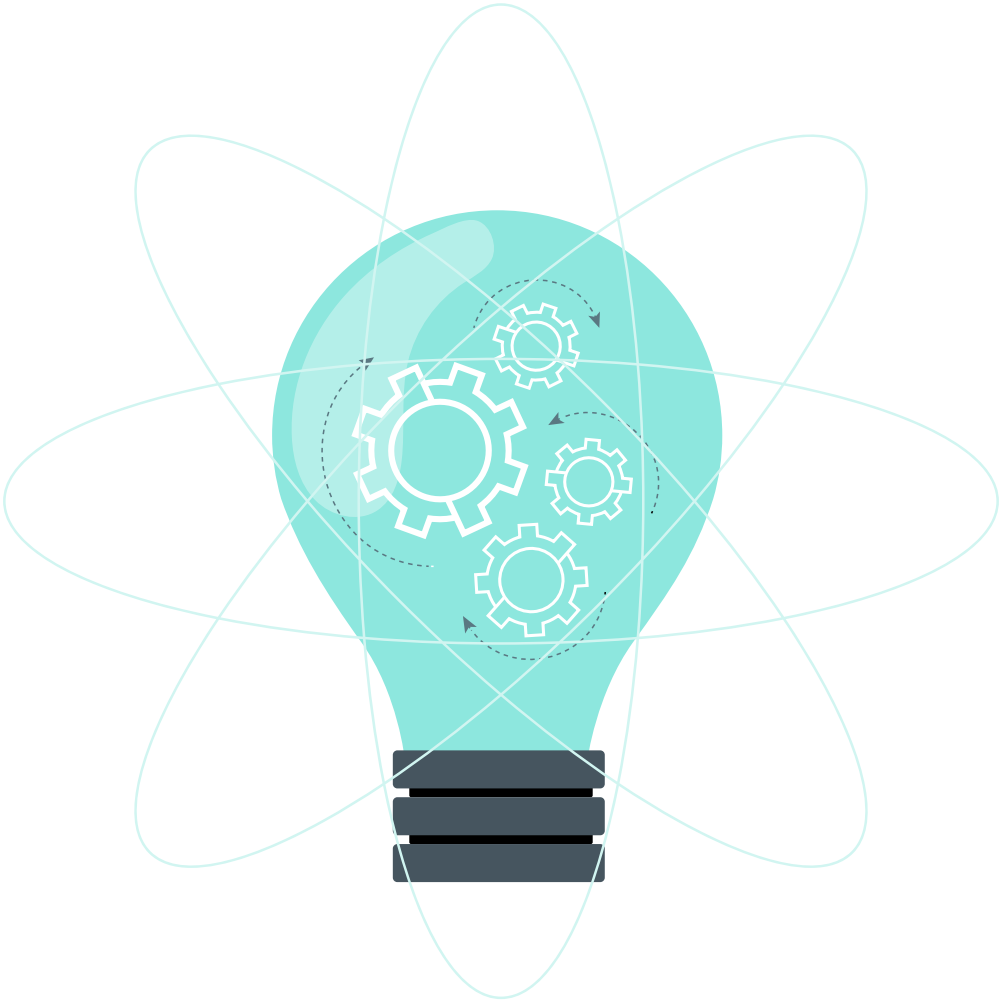
Minimise Bureaucracy

Adopt a lean governance approach by stripping down unnecessary bureaucracy and focusing only on what truly matters. Establish “minimum viable compliance” – the least amount of compliance and control required to meet legal and regulatory obligations. Ensure that processes, systems, and infrastructure are compliant by design, reducing the need for continuous manual interventions. This approach allows for both control and flexibility.

Real-Time Monitoring and Analytics

Use real-time data and dashboards to monitor governance adherence and delivery performance. Data-driven insights can help identify areas of inefficiency, allowing for rapid course correction without impeding the value stream. Develop key performance indicators (KPIs) that measure both compliance adherence and delivery efficiency. Metrics like cycle time, lead time, and governance intervention frequency can highlight the balance between control and optimisation.

As discussed under Metrics and Dashboards, real-time analytics should be an integral part of day-to-day working practices.



A huge opportunity

How effective are your delivery processes?

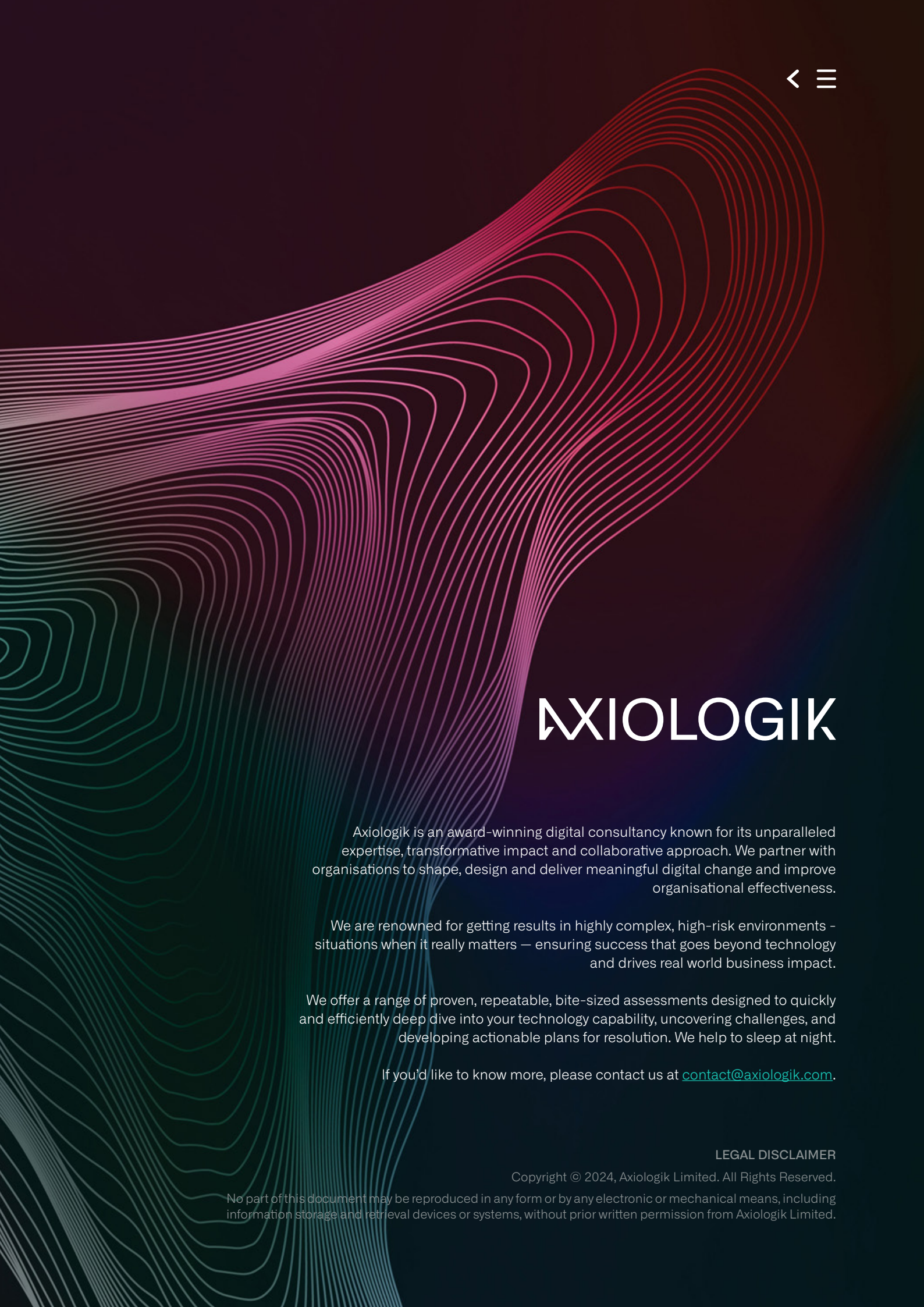
A 2020 report by McKinsey indicates many organisations achieve only 25 to 30% of their potential. This is a staggering metric. Unless you are a high-achieving organisation (and statistically, you are probably not), the opportunity for improvement is massive.

Through the early part of the twenty first century, organisations invested heavily in “agile” transformations, ironically, looking to achieve big-bang improvements. Although progress was and continues to be made - Agile and DevOps practices are now ubiquitous - the full opportunity has only been realised in a few elite organisations.

Organisational structures, processes, architecture, tooling, and governance often hinder end-to-end value flow. Suboptimal operating models, bottlenecks, communication overhead, inefficient tools, and restrictive governance can all impede progress.

The good news is, these challenges are not insurmountable, and do not require organisation-wide transformation programmes to achieve positive outcomes. Change activities can be iterative and incremental: start with a simple diagnostic (e.g. value stream mapping); identify the constraints and bottlenecks; and start to address them (in priority order) - learning from success and sharing these to broaden the change across the enterprise.

Importantly, this is a virtuous circle. As delivery teams are relieved of constraining processes and governance, and suboptimal environments, their personal satisfaction will increase, along with productivity. And as teams feel empowered to drive change and improve their environment, they will adopt a dynamic culture that embraces change innovation.



AXIOLOGIK

Axiologik is an award-winning digital consultancy known for its unparalleled expertise, transformative impact and collaborative approach. We partner with organisations to shape, design and deliver meaningful digital change and improve organisational effectiveness.

We are renowned for getting results in highly complex, high-risk environments - situations when it really matters — ensuring success that goes beyond technology and drives real world business impact.

We offer a range of proven, repeatable, bite-sized assessments designed to quickly and efficiently deep dive into your technology capability, uncovering challenges, and developing actionable plans for resolution. We help to sleep at night.

If you'd like to know more, please contact us at contact@axiologik.com.