DXC LEADING EDGE

Platform culture:

A FIELD GUIDE TO MASTERING THE ART OF INFORMATION FLOW

How to revitalize and recast platforms as operating models

Operating in the **maelstrom**

caused by the pandemic, war and other external shocks is an everyday business reality — and pivoting to deliver in an inconstant world has become the new necessary.

Yet few organizations have adapted well; many find they lack the means to create the flows of data and software capable of distributing value to the right destination at the right velocity.

Those that have adapted are often relying on platforms as a primary path to achieving that highly productive state where work, data, and fast and distributed transformation coalesce.

Platforms can take many shapes: Some are internally focused on productivity gains, while others target markets facilitating ecosystems. Others assemble apps and infrastructure into

workflows through a technology platform. And some are high-value industry platforms with custom capabilities. While there are plenty of varieties, digital is the central ingredient — and culture is what delivers the change needed to support new operating models.

We at DXC Technology are not the first to endorse the idea of change delivered via a platform powered by the grassroots, rather than change programs ordained and implemented from the top.[1] DXC has a special vantage point, however, as we've observed the way that every new platform updates the people, processes and technology of the business as it evolves and grows.

We've had a hand in helping drive organizational changes, from silos of project teams toiling with difficult, slow and complicated systems to autonomous cross-functional teams easily consuming platforms as products to bring new features to life much faster (see **Figure 1**).

Platforms are capable of supercharging delivery efforts across organizations, when the platforms are reimagined and operated as change agents. DXC Leading Edge's previous paper, **Rethinking digital platforms as change agents in a software-defined world**, outlines how dynamic digital environments consisting of people, processes and technologies transform modern platforms into conduits for high-velocity innovation and flow of work, data and change. In DXC Leading Edge's paper **Mastering platform-driven business** — part of our **Accelerated Now series** — we discuss the importance of designing an organization to flow, and how software engineering provides this flow.

[1] McKinsey & Company, *Build a change platform, not a change program*, October 2014.

TRANSFORMING TOWARD A DIGITAL NATIVE OPERATING MODEL

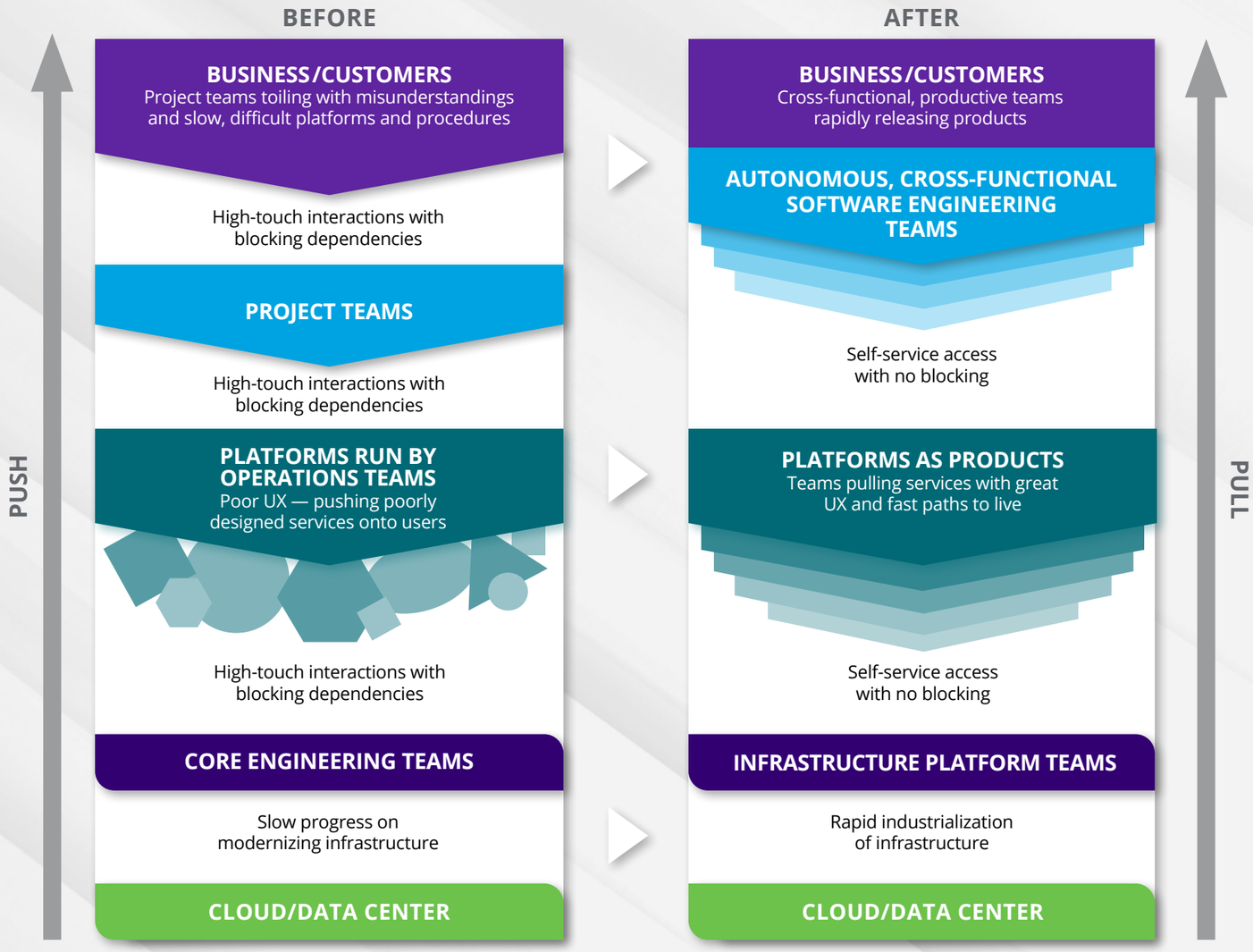


Figure 1 ▲

This current paper, a field guide, focuses on the cultural changes our customer experienced when a DXC Technology SWAT team developed and continuously curated a new product management mindset with the customer, where platforms could drive change on multiple levels and across multiple teams. When taken together, the working practices in this field guide help to power an insurgency philosophy.

By treating the platform as a product to drive close alignment between platform output and business value … the DXC team was able to respond to the shortcomings that were choking delivery for the firm.

The outcome of DXC's actions liberated and directed value flow at a major London financial institution. We believe that this approach is an exemplar for how platforms can be revitalized and recast as operating models.

By treating the platform as a product to drive close alignment between platform output and business value, and remaining faithful to an Agile view of the world, the DXC team was able to respond to the shortcomings that were choking delivery for the firm. The DXC team asked — and answered — the simple question: "If the product feature is not delivering value now, why am I delivering it now?" The question shifts the role of platform from the domain of infrastructure to the revitalized space of platform as product.

By tearing up the project management rule book, the team achieved clear results — including bringing a "good enough" product to market up to three times faster than did previous methods.

This field guide dives deep into the aspects that characterize a successful platform-as-product approach. These include:

- Placing preeminent, multiskilled and autonomous teams at the heart of delivery
- Industrializing software delivery
- Embracing a new team dynamics approach
- Remaining aligned with project scope
- Overcoming new-world frictions
- Adopting new interaction patterns

To learn more about the power of platforms, see our related papers:

- **Mastering platform-driven business**
- **Rethinking digital platforms as change agents in a software-defined world**

DXC's pioneering approach

Many financial institutions have ingrained ways of working that have developed over decades, and in some cases centuries, of delivering value. This process becomes more complex when considering the global scale of business. In order to turn the eyes, heads and minds of a customer institution to the future, the DXC team has nurtured an operating model and practice that matches the platform's delivery cadence with requisite value served to disparate owners across the global business. It has assiduously cultivated a product that comes with its own budget, customer base and governance.

Locked into its set ways of doing things — as so many businesses of this size, longevity and legacy are — the London financial company struggled to deliver. It was stuck in the mode of releasing a product only when it had been exhaustively tested; sometimes, by the time it was released to business stakeholders, it was no longer relevant or useful.

Into this slow-moving environment of rigid practice and silos of product development, a pod of five DXC engineers working with an application team with a modest remit decided to deliver stuff that was "good enough" to launch — often called a minimum viable product, or MVP — and that could be improved incrementally. The net result was to quickly bring change to the market.

Five years down the road, the team's insurgency philosophy has reaped outsized rewards and pervaded other development domains. The platform is a proven entity that not only fuels developers but also standardizes and industrializes software development at scale. Crucially, too, it has evolved and grown to become the target operating model (TOM).

Methodically and intelligently automating software products as components embeds value into infrastructure and libraries. The process also impacts the product's success in going viral; this cannot be underestimated. The customer's product teams across the financial services organization were attracted to the ecosystem, and the platform has effectively become the preferred operating model.

The journey was organic, says the platform team leader, who emphasizes the need for organizations to continuously adapt according to their own business imperatives. The methods outlined here encapsulate a way of thinking and doing that are economical and elegant, and make the platform a steward of value delivery.

Putting people and processes at the heart of development programs confers an innate and rewarding dynamism.

The platform way calls for courage and determination in the face of old-world intransigence: Old models must be broken, and rule books must be torn up. But as platform masters know, putting people and processes at the heart of development programs confers an innate and rewarding dynamism. Nothing is more exciting than using platforms to deliver product and value.

The way of accelerated delivery: 7 principles

The everyday reality of advanced, digitally intense firms, where platforms yield business value in streams of increments, demonstrates these seven principles:

1. Led by a preeminent team

In a world of continuous and iterative change, the multiskilled and autonomous team takes center stage, surpassing the role of technology (see **Figure 2**).

The idea of having a preeminent team, which is core to the mission at the London financial institution, grew out of a prior experience faced by a couple of the DXC engineers who continue to collaborate on this project. They had worked with a government agency to make changes to its benefits payment processes, as required by new legislation. But getting any changes through the existing IT delivery organization — which had become a peat bog of skills-based silos, including application delivery, development, networking and Linux — was convoluted.

Facilitating change meant breaking the mold. Silos were torn down, and every team was tasked with delivering a different piece of functionality or product. So, each included a Unix specialist, a network person, a coder and so on.

The delivery platform was treated as a product within its own team. In essence, the new model comprised multiple autonomous, multiskilled teams; each is embedded with every skill or capability

Figure 2 ▶

TYPES OF TEAMS

	ENABLEMENT TEAM	PLATFORM TEAM	STREAM-ALIGNED TEAM
	Small team of experts in a specific domain who mentor full time and enable stream-aligned product team members to grow their capabilities and expertise so they become self-sufficient.	**Team that builds and supports a platform, which is a foundation of self-service APIs, tools, services, knowledge and support — arranged as a compelling internal product.** **Platform team should be distinct and separate from enablement team. Platform can be a logical platform built upon other inner-platform(s).**	**Team that builds and delivers customer or user value as quickly and independently as possible, without requiring hand-offs to other teams so they can perform parts of the work.**
PURPOSE AND GOALS	• Help teams increase autonomy and flow of value • Reduce teams' dependence on experts by teaching rather than doing the work • Reduce teams' learning curves when adopting new principles, patterns, practices, technologies and culture	• Enable stream-aligned product teams to deliver work with substantial autonomy • To serve stream-aligned product teams • Reduce cognitive load of teams	• Delivery of a single, impactful stream of work
TEAM SKILLS	• Expert in specific domain(s) • Able to explain things well • Mentor; enjoy helping others • Driven by willingness to see another team succeed	• Full-stack software engineering skills • Product management skills	• Full-stack application development • Test/test automation • UX/UI user research • Product management
EXPECTED BEHAVIORS	• Look for blocks to fast flow • Sense problems and opportunities across teams • Help product teams succeed (measured by success measures of product teams) • Not be a blocking dependency and not "doing the work" for the teams • Not be an ivory tower that dictates technical choices • Proactively seek to understand stream-aligned product teams	• Create self-service capabilities (no: email, tickets, waiting for response) • Strong focus on user experience; make it compelling; opt-in, not forced to use • Support platform when things go wrong • Use product management techniques and skills (sit with users); actively understand usage • Create thinnest viable platform, and continue to make thinner and easier to use	• Strong focus on delivering customer/user value • Continuous feedback and adjustment • Product management techniques and skills, actively understand usage • Well-understood scope and interfaces, minimal hand-offs to other teams • Agree and report against business-focused metrics
AREAS OF INFLUENCE	• Security/compliance • DevOps • Way of working/Agile • Test automation • Performance • User experience • Mobile	• DevOps pipeline template • Test automation • Security/compliance • Identity access management • Environment provisioning • Performance • Cloud platform • Logging/monitoring	• Product • Service • Single user journey • Single user persona enablement

required to deliver its remit. This approach, which predated container technology and DevOps packaged thinking, represented a radical departure from the norm.

2. Industrialized through software delivery

To restructure the approach to manage the platform-as-a-product, software delivery is imperative: turning software into reusable commodities — and enabling its automation. This entailed taking a piece of code or functionality, which was normally hand-cranked and that would be needed more than once, and automating it so that it could be scaled safely.

The lessons learned from adopting automation technology for the benefits project at the government agency were integrated into a new, evolving model. Gradually, the platform became the engine of industrialization: constantly and intelligently executing processes in order to componentize, modularize and automate so that software developed for specific products becomes part of the shared infrastructure and libraries.

The platform became the engine of industrialization: constantly and intelligently executing processes in order to componentize, modularize and automate.

3. Adopt insurgency philosophy for new team dynamics

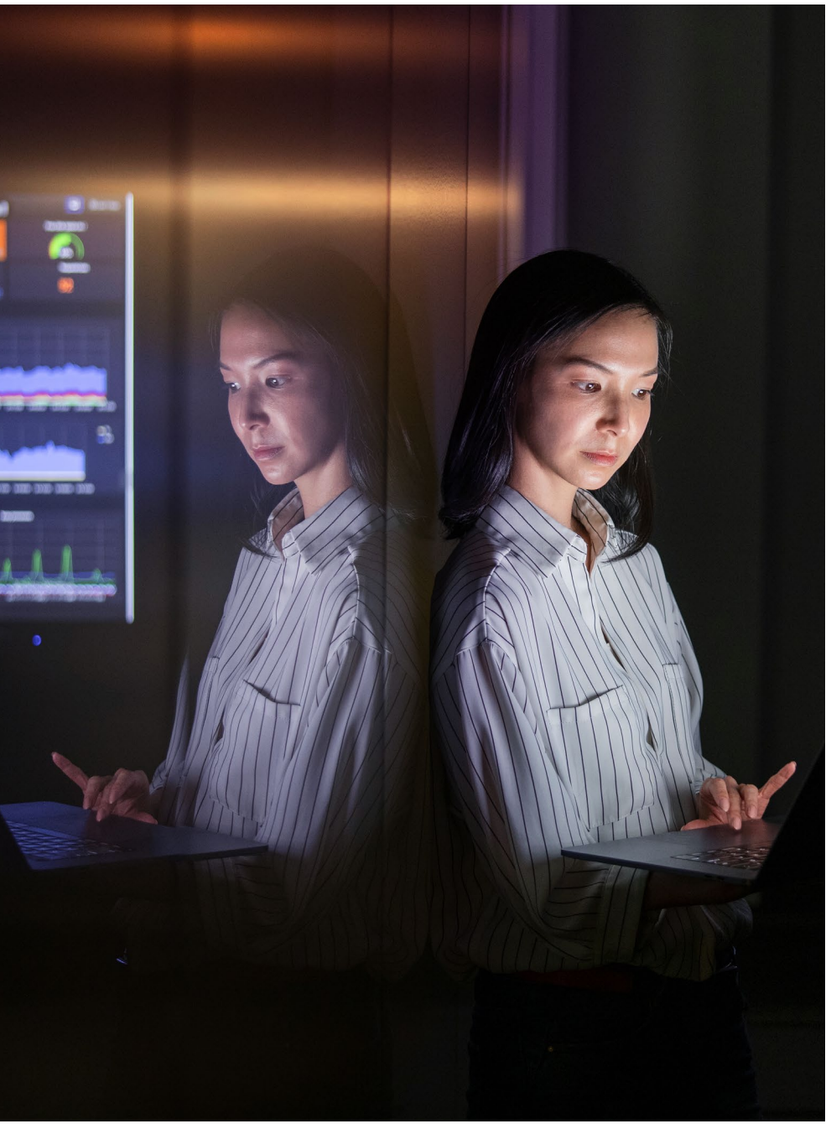
At the financial institution, against a backdrop similar to that of the government agency, DXC developed a new type of team without a template — although it is validating to reference our work about rethinking how teams operate within platforms against the book *Team Topologies* by Matthew Skelton and Manuel Pais.[2] This is a highly recommended read for any business that is rethinking how teams work within platforms.

Teams cohered around the ethos of delivering capabilities to accelerate the business. This shared motivation helped overcome early difficulties and navigate conflicts that are an ongoing part of platform-led delivery. Ideally, the management load should be minimal within each team, which takes responsibility for its own technology decisions. Difficulties and learnings can be shared or reported within a scrum of scrums every morning when team representatives meet.

4. Size and longevity: Less is more

An early rule was around team sizes. The DXC leaders followed Amazon founder Jeff Bezos' oft-quoted two-pizza rule: If a team can't be fed by two pizzas — i.e., roughly 7 to 8 people — then it's too big, and should be split into smaller units, each with its own remit. Equally important, teams are transient and come together for the purposes of delivery. Once the product is finalized, they can be disbanded and members can select a new team with a fresh remit. Paradoxically, enablement teams attain a longer life: Because they perform, they are given another purpose rather than switched off.

[2] Matthew Skelton and Manuel Pais, *Team Topologies,* 2019.



DXC discovered that team members enjoyed the business focus and constant learning that accompanies transience and an insurgency ethos. Even the Kubernetes diehards understood that in a few years' time, their favorite technology would be commoditized, embedded in the cloud and accessed by an API. Constant evolution is a given for teams working within platform environments that are change agents; moving people between teams on a regular basis keeps parity between teams, shares information and maintains momentum.

5. Scope is paramount

The biggest challenge when working in this product-oriented way is to get the scope of teams right. Teams can all too easily be inflated by a "we can do that" over-eagerness; conversely, they may push work away, even when it is logically theirs. As an essential prerequisite, it falls to the project manager to prescribe the vision for each team, define its workload, and champion and manage the effort.

In this approach, enablement is expressed as a bottom-up process. Engineers from the platform team in each stream-aligned product team facilities are "doing stuff" and sharing it back out.

6. New-world frictions

Even when teams are precisely scoped, friction can occur in gray areas where remits collide or impinge on common territory.

Here is an example: the issue of who should take care of authentication and authorization for a specific product set. A product team must ensure its application under development can securely make use of the designated authentication and security software, and safely call other services. But what happens when multiple products need to access the same security features, and the customization work is done in different ways? At this point, it makes sense to break the authentication piece out and allocate it to a dedicated team.

These kinds of adjustments are made on a regular basis. Another example might concern whose job it is to spin up a database — is it the job of the platform (infrastructure) team or the application delivery team that needs it? The question can be answered by posing another question: How many customers are there for the database? If the answer is none, then it falls to the development team. Such negotiations are ongoing for common components like workflow.

7. New interaction patterns

In a platform environment, it's normal for gray areas to be regularly batted around at scrums and standup meetings. Embedding appropriate and proportionate interaction patterns and touchpoints into the platform environment is key. Calling out gray areas early on avoids duplication and keeps the platform flowing. Communication is essential also for decisions about industrialization: If database access is only for one customer, for example, there's no point in automation.

Each team needs to understand who they are interacting with on a daily basis and how communications should be set up with the teams consuming their enabling service. Daily scrum of scrums for leads of each stream-aligned team and management team, including a release-trained manager, are supported by an architecture governance.

Do the plumbing, and release value

Setting up the platform environment around multiskilled and autonomous teams takes care of the infrastructure, or plumbing, and allows developers to get on with their job of delivering business value rather than having to set up databases, virtual machines, containers and so on. In traditional IT shops, a huge amount of developers' time is sucked up by plumbing. The platform team's job is to take care of the plumbing. By developing once so that components can be consumed by many, the platform team creates a self-service capability through pipelines.

When rethinking platforms as agents of change, it's helpful to think of them as moving things from left to right on a **Wardley Map** — a world-recognized framework for **value chain mapping** created by DXC Leading Edge senior researcher Simon Wardley — turning everything into libraries of infrastructure and practice. Crucially, the way people work changes when they are organized into stream-aligned teams. Creating product teams and embedding a platform engineer in each orchestrates people, efforts and outputs in a coherent way. Organized in this way, under any maturity scheme, a team will progress up the curve.

Creating product teams and embedding a platform engineer in each orchestrates people, efforts and outputs in a coherent way.

[3] Jonathan Allen, Thomas Blood et al., *Reaching Cloud Velocity: A Leader's Guide to Success in the AWS Cloud,* Section 2.3, "Mapping your way through" (Simon Wardley), 2020.

TURNING PLATFORM ENVIRONMENTS INTO CHANGE AGENTS

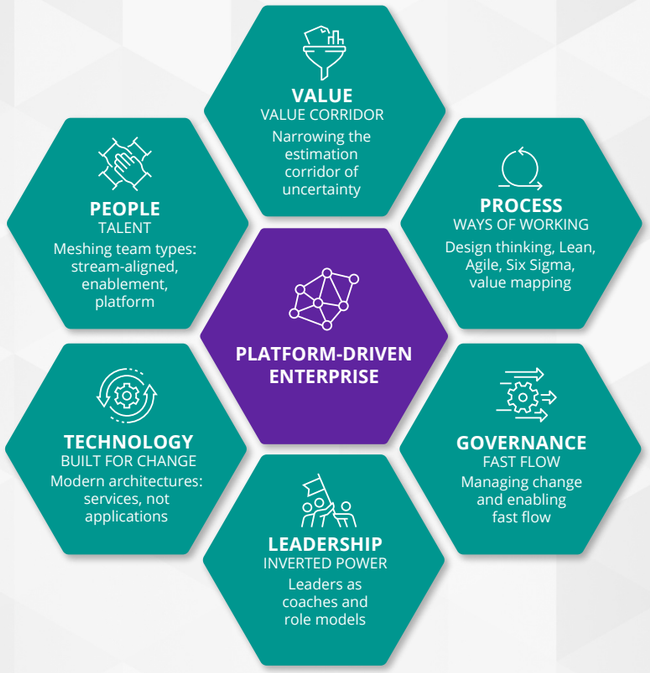


Figure 3 ▲

How platforms change organizations

In converting platforms into change agents, it's useful to consider the six lenses through which we scrutinize activity and change (see **Figure 3**) — value, process, governance, leadership, technology and people.

Value

The fulcrum upon which the successful platform pivots is value delivery. Value generation can be embedded into the platform environment of engineers, tools and processes by recasting it as a product. In the old world, the platform is treated as a discrete infrastructure project with a set plan and finite investment, at the end of which it is turned off.

Treating a platform as a product is transformative because the product is accompanied by budget, an investment life cycle and iterative thinking, which in turn support longevity and a higher degree of success. Crucially, this approach yields early value realization, which builds confidence, encourages people to think long term and attracts further investment. When teams deliver against a budget, increments of the product can be quantified as returns, making estimates of future value more accurate, too.

When you flip it and treat the platform as a product, it encourages people to think long term and to continually adjust. The new paradigm does, however, create new problems: Adopting a product mindset also surfaces flaws in decision making early on that need attention, which trumps the alternative of sticking with early decisions and delivering flawed products 5 years down the line.

But the irrefutable incentive to adopt platform-as-product thinking is the capacity to deliver business value early. A "good enough" ethos justifies the speedier delivery of a raft of applications and products that can be delivered in flight and improved along the way. This cadence leads to the notion of proportionate delivery, where value is released incrementally in step with business needs.

When you flip it and treat the platform as a product, it encourages people to think long term and to continually adjust.

At the London financial institution, it typically took 2 years from inception to getting a product out. Following a reorganization around platform thinking, a product was delivered every 6 months. The benefits of being able to release digital orchestration capabilities for customers globally have been huge. In fact, the issue looming next year is expected to be not technology delivery but the ability of business customers to absorb the rate of change.

Process

When industrialization is the primary objective, processes require careful curation to support change agents. The biggest challenge in deploying Agile processes is gearing up from a development mindset to running a business project at scale. In other words, how

to take a bunch of software that's been developed individually to a state where it is repeated, supported, developed and scaled?

Industrialization calls for a radical shift from the incumbent data center mindset: Instead of developers nurturing, caring for and fixing software, they follow the strategy that, if a server fails, just dispense with it and stand up another. And equivalent pieces of software are treated the same. Standardization is achieved through automation, templating and blueprinting, which means that software can be securely and predictably scaled.

In a platform world, there's no hand-cranking, and engineers focus on automation.

Automation is the most important of these. In a platform world, there's no hand-cranking, and engineers focus on automation. It's a throwaway world where serverless and container technology is never patched but simply replaced and redelivered through the pipeline. Carmaker **BMW executes its computer simulations** by managing one of the largest container platforms in the world. An eye-popping 2,000 prewarmed, freshly built containers are spun up in 15 minutes, every day.

Similarly, **chaos engineering, as championed by Netflix**, aims at achieving continuous assurance. No one should notice when server X is turned off, as it should fail over without a hiccup. The new-world order means dispatching the artisan developer and system admin in favor of automation.

Governance

Like leadership, governance presents challenging terrain to negotiate in order to keep the platform flowing and not be throttled by the iron grip of procedure. The financial firm adopted "insurgency governance." While complying with industry and organizational rules that reduce risk in a regulated sector, it developed a more fluid governance regime within the platform domain. Existing governance is non-negotiable, but the team owns the platform and decides how and when to put in controls. This influences other partners and technical design authorities (TDAs).

Platform leaders took their cue from the Agile world and from Jeff Bezos in particular: It's a waste of time and energy micromanaging a team that is competent or making decisions that can be easily reversed. In Bezos-speak, these are Type 1 or Type 2 decisions; respectively, irreversible choices or easily rectified choices. However, even within the Type 2 decisions, allowing teams and individuals to make their own decisions about tools and versions can lead to chaos and increase management overhead.

The answer at the financial firm was to embed governance — or controls — and manage Type 2 decisions within the platform itself. Offering a basket of tech tools and goodies makes life easy for developers who can obtain a database or a set of microservices with the click of a button. People may choose to implement microservices differently. This led to the concept and design of a chassis — everything is prebuilt, leaving on top the task of developing the API.

Pushing decisions down while managing them in parallel not only helps in standardization but powers industrialization. It also manages risk in a way that is acceptable for the customer. DXC's exemplar platform lets developers sign off on decisions around how to code, implement and organize a business process. The platform TDA is thus autonomous; consulting the program TDA is only for big ticket queries, such as security oversight. In turn this is approved by the market TDA, owned by the customer.

Leadership

Sound platform leadership depends on getting the tension right between centralized control and the devolution of decisions to engineers and developers. Recognizing that different stages or aspects of the program will call for that tension to be revisited and retuned is just as essential. Pioneering firms have benefited from having a visionary at the helm, and the fruits of a servant-led capability transformation center include a mature platform that serves value creation.

The starting point at the financial organization was to shun the micro-governance often favored by lead architects. Instead, the platform leader cultivated an attitude and organizational cultural change within the delivery team based on implicit trust. "Do what needs to be done to deliver, and I'll get out of your way" was the ethos from which productive program behaviors and rules were gradually assembled. The leader and his people felt their way toward a new way of lean working rather than following prescriptive rules for decision making and team interactions.

Platform environments are a novel space for leaders because they need constant curating. Navigating a route between the two poles of centralization and devolution is a continual adjustment.

When every decision is made centrally, the number of queries and escalations increases and eventually throttles workloads. But when every team member makes their own decisions, chaos can quickly ensue. Within platforms, daily debate is required to get the balance right in order to maintain flow.

Technology

The rise of the multiskilled and autonomous team as our unit of delivery impacts not only how we organize to deliver, but also our approaches to the technology underpinning that delivery. By taking a team-based approach we are breaking down the mechanistic silo approach where each silo is only responsible for a step in the chain to one where a team is responsible for a self-contained capability — be that a business function, user interface or technical enabler.

This means decoupling the architecture into discrete addressable chunks to match the decoupled teams. This has led to the rise of platform-powered microservices-based, API-first, cloud-native and headless (MACH) architectures:

- **Microservices.** Independently developed, packaged and deployed business services that are addressed through an agreed-upon API
- **APIs.** The method by which services interact/integrate with each other; multiple microservices communicate through APIs to form a microservices-based architecture
- **Cloud-native.** The use of cloud commodity services that allow for hosting, storage, automated service update and elastic scalability based on a pay-for-what-you-consume model
- **Headless.** Decoupling the front-end user experience from back-end services

In essence, each bounded component delivered by a team is pluggable, scalable and replaceable, and can be continuously evolved through iterative, Agile delivery to meet the ever-evolving business requirements. The modularity and autonomy inherent in MACH-based architecture enables composable architectures, where components can be reused — assembled outside of their initial application architecture to build new business applications more quickly and easily.

This then increases return on investment and reduces the time needed to bring new services to market.

> The modularity and autonomy inherent in MACH-based architecture … increases [ROI] and reduces the time needed to bring new services to market.

People

Platforms help distribute the cognitive load, taking the plumbing, or white noise, away from coders so they can concentrate on the job in hand and deliver value. Without the platform, developers would be troubleshooting environmental issues and doing plumbing that delivers no business value.

In traditional delivery models, people in infrastructure, network and other technical silos spend hours sending paper and emails back

and forth in the process of delivering products. In the platform-led delivery model, product teams have all the skills needed to deliver a business product, thus reducing hand-offs. But a significant vulnerability remains, and it threatens to deter the productivity and impact of the platform: Do people and teams understand what they can or can't do?

Most teams suffer from not having clearly defined scope, and this detracts from their performance. A big piece of the DXC platform effort was to document teams' scope and make it explicit. A lot of time is spent on onboarding, and a platform person is embedded in every product team. Having autonomous teams, reporting into program level via standups, is workable. Having a maximum of seven teams reporting to a tribe leader, and seven tribes reporting to the center leader, was deemed optimal to retain visibility from top to bottom.

Most teams suffer from not having clearly defined scope, and this detracts from their performance.

It's worth noting that the communication mechanisms of the new world can bring their own problems, too. Agile ceremonies that are followed dogmatically can be time-wasting and ineffective; smaller teams leveraging email communications are more effective than daily standups. Beware any rigidity, as it kills delivery. Agile ceremonies that are followed dogmatically — such as daily standups — can be time-wasting and ineffective for smaller teams, which can leverage email communications more effectively.

Launching and scaling platform-led change

In our report **Rethinking platforms as change agents in a software-defined world**, we discussed the main efforts required to launch and scale platform-led change (see Figure 3 in that report). Now in this section we expand on and give more context to those efforts.

Launching

The goals

Normally an external driver precedes any IT build. Picking something with a clear goal and business value is optimal, such as an engine-controlled system that needs a digital twin and supporting systems. The crucial question remains whether the platform will help realize the goal. If a company only needs to release a product every 6 months and has a captive audience, there's no justification for the upheaval or cost of investing in Agile.

Business mandates to drive down costs or deliver applications faster may push IT to make a bid for the driver's seat and respond by proposing a platform. Whether the call to action comes from the IT function or the business, it takes a united vision and effort to realize it, facilitated by ways of reporting success and benchmarks for knowing what good looks like.

The organization

A platform begins with a decision, is guided by vision and depends upon having the courage to throw away the rule book when necessary. It needs organization in a new style and spirit. Getting a bunch of good people together and leaving tribal badges at the door is a sound start. Rigorous rules around keeping teams small, communication minimal but robust, and maintaining access to funding were **devised by Kelly Johnson** to solve chronic procurement problems at aero-engineering giant Lockheed Martin. These work very well for today's platform environments.

The vision must articulate platform-as-a-product: The loosely cobbled space of people, processes and technology is defined by the product it delivers. Otherwise, it's in danger of becoming an amorphous blob — the fate of many old-world IT projects. As we've discussed, it's critical to define ownership and team remit, including what is beyond its scope and the rationale for disbanding the team, if necessary. These elements are central to the product ethos.

Communication flows keep the platform flowing and viable as a change agent. We've seen that this is needed for the continual adjustment of tension between centralized and devolved decision making. A scrum of scrums among product leaders provides a forum for discussing gray areas that are a permanent feature of the fast-moving and dynamic platform and providing senior management with regular reports.

A critical aspect of virally propagating success during a platform launch is to share product achievements and learning. Success breeds further positivity and progress, attracting more investment in turn and creating a virtuous circle of success. Communities of interest (CoG) at the London financial services company proved an effective way of propagating success within the platform. CoGs were designed with a basic structure and members given targets and tasks, but this was premised on simply sharing experience across platforms and letting junior members shine. Whichever platform group, process or communication is set up, it must be aligned to clearly defined goals and measurable outcomes.

The technology

For incumbents constrained by project methods and presumed ways of doing things, an insurgency philosophy helps spin up the necessary clean slate environment. A chassis — comprising technology building blocks to provision a minimum functionality — works alongside the vision and rules of engagement to provide that clean slate. A chassis enables a platform to become a just-in-time engine, delivering products and increments of value to the business at the requisite time.

The key to making the chassis indispensable is to make it a candy store for coders, cramming it with highly consumable widgets that stymie reinvention, promote standardization and boost productivity. The chassis is the engine that powers the pipelines along which products are developed, launched and finessed. It's a non-negotiable tool for the platform leader, whose mission is to make the entire environment sticky: Thinking like a fast-moving consumer goods (FMCG) chief by relentlessly developing demand and updating its tech components helps maintain the chassis' role in assuring platform momentum.

With product delivery souped up in this way, impediments and bottlenecks that may slow down the platform flow can occur around testing and release stages. These delays are more likely to beset incumbents than startups, because the two are burdened by different traditions and philosophy. Traditional IT shops test everything to within an inch of its life, whereas startups tend to do a cursory test and publish, and if a problem crops up, follow up with another release. It's necessary to find a happy medium between a unicorn company and old-school IT shop.

A change advisory board (CAB) presents a potential blocker: It's possible to have weeks of changes queued up, with mass batch releases becoming riskier to absorb and manage. The leaders in the financial services organization secured permission to release changes frequently, and instituted a monthly CAB. When acceptance is slower than development, though, there's a delay of markets and customers absorbing changes, too. Navigating between TDAs and CABs is essential for platform flow and maintaining momentum as a change agent.

Scaling

Scaling a platform to become an organization-wide change agent that drives innovation and production means moving far beyond a sandboxed enablement of discrete product development (see **Figure 4**). It calls for the ability to network and run in parallel multiple platform environments that are powering product development and other innovation tasks. For this to happen, a TOM must be embraced by all parties: a willingness to cohere sound shared governance, communication patterns and an Agile philosophy.

Bayernwerk scales up platform to drive innovation

DXC is helping South German power plant and heating supplies operator Bayernwerk modernize rigid business processes by scaling its modern business integration platform in the cloud, named Socrates, across all lines of business. The platform enables better use of business data and the creation of new digital business models to support Bayernwerk's goal of co-creating innovative business products that will shape the future of energy.

STEPS TO SCALING A MODERN PLATFORM ENVIRONMENT

PHASE ONE

ROADMAP FOR BUILD

Establish platform and enablement teams, platform-as-product governance, the technology chassis and a fast route-to-live pipeline. From the start, build for self-service with APIs. A platform product roadmap guides the build.

PHASE TWO

PLATFORM AS PRODUCT

Multidisciplinary teams assemble digital products, aiming toward autonomy. Meanwhile, the platform is managed as a product, establishing pull by offering new, high-quality UX portals, and constantly upgrading features and route-to-live speed and reliability, based on daily feedback. The component and service libraries become increasingly API-accessed.

PHASE THREE

DECOUPLED ENVIRONMENT

The environment moves toward decoupling architecture and teams. The new, high-performing practices drive discussions among other development teams, security, release management, testing, compliance and change management about the processes in the other paths to production/release.

PHASE FOUR

STANDARDS AND BEST PRACTICES

Each major platform now has a dedicated team. Interaction patterns are implemented to reduce cognitive load and toil for all teams, and Type 1/ Type 2 decision making is becoming the norm. Communities of practice harvest the excellent, which supports developing standards and best practices. Stakeholder education continues.

The technical teams are establishing high degrees of automation, standardization and compliance through route-to-live pipelines, and there are prepaved common microservice application patterns. Inner sourcing is working at scale, and industrialization roadmaps are followed.

Figure 4 ▶

Getting started

Pursue viral drive to organic operating model

Acceptance of a common operating model is as important as provisioning any enabling technology, but embedding a platform-led insurgency philosophy outside a tight-knit team of 10 people is a big ask. When one party's platform culture and operating style meet another party's working method (whether internal or external) there can be a clash of worlds. If another team hasn't yet embraced Agile ways and is entrenched in **PRINCE2** methods, for example, there will likely be stasis.

The platform exemplar discussed here was a greenfield site, and the leadership prescribed a TOM based on scaled Agile. Selling the platform culture and method across the organization depended on viral success rather than edict, and meant the operating model was organic. Early on it was apparent that there was no point scaling the platform unless it was possible to exert some influence on those consuming it.

Secure a stake in the investment cycle

When you scale a project, it's essential that you stake out a piece of the organization's investment cycle, otherwise the money will run out and the project will close down. Platform-as-product positioning is key, with business plans for the platform covering people, processes and underlying technology.

IT shops are generally not accustomed to this rigor and need to get used to the idea of competing for investment: It's a good discipline because if you can't defend or articulate what is being done on a platform, you shouldn't be doing it. A recommended first job is to take ownership of all infrastructure elements, explain to the business the reasons for them and present these back to the board. This exercise provides traceability and accountability, which are both levers for industrialization and scaling.

Platform leaders will need to be highly proactive to combat the perception that platforms are a finite technology project. Without being challenged, the stereotype will persist and work against success. Despite 5 years of acting as a change agent, the platform leader and team at the financial services company are still asked by program leaders and others: "When will the platform be finished?" Product positioning remains a continuous education exercise.

Keep product perspective top of mind

When you're engaged in establishing platform culture and bedding in an organic TOM, both premised on Agile, staying true to a product mindset maintains the constant nurturing and flexible outlook that engenders success.

It's worth looking at how the professional FMCG brigade go about their business, noting their constant attention to how a product is adopted by customers, merchandised, improved and packaged. Without similar market testing and tweaking, the platform — and its bigger sibling, the TOM — are in danger of becoming white elephants.

Local, audited autonomy encourages an ethos of decision making and militates against no decision making, which is the death knell of delivery.

Having proved itself in the smaller arena of the platform, the TOM can be introduced to the wider world. And as it expands further across ecosystems, the TOM changes, and accepts compromises and additional controls. Governance essentially becomes more contextual, and at the local level of controls it becomes guidance.

Prescribe team audit trails for accountability

In this organic, fluid and fast-changing environment, accountability is essential. At the financial services company, each team must make its own decisions and keep a decision log. This way, there's an audit trail, and managers and leaders can understand in full context why any decision is made. Local, audited autonomy encourages an ethos of decision making and militates against no decision making, which is the death knell of delivery.

Importantly, it's about devolving decisions to the right levels, to be made by local and informed people. When platforms act as change agents, no one is standing in front shouting, "Charge!" Nor is culture changed from the top. Instead, autonomous, multiskilled and accountable teams share and produce — and investment in training is even more important than spending on tooling.

As we can see, it's clear that thriving in today's increasingly digitalized world will demand shifting the role of platform from the domain infrastructure to the revitalized space of product. By leveraging platforms as products, businesses will speed delivery and drive business value.

Businesses that hope not only to survive but to retain leadership in their markets as they compete with more advanced digital competitors must embrace this approach. Shake off old and rigid blueprints, and join the forward-thinkers who are curating the platform as change agent and adopting it as operating model.

About the authors

Bill Murray is a senior researcher and advisor for DXC Leading Edge, conducting research programs on the metaverse, digital twins, platform businesses and ecosystems, and software-intensive organizations. Connect with Bill on **LinkedIn** and **Twitter**.

Nolan O'Dwyer is lead architect for DXC Technology's UK Modern Application Delivery capability. He brings extensive experience in applications transformation, cloud technologies, Agile delivery models and DevOps philosophy.

Robb Shally is a managing partner in Americas Consulting at DXC Technology and leads the IT Modernization and Optimization practice. Connect with Robb on **LinkedIn**.

Learn more at
dxc.com/leadingedge
dxc.com/applications