Unified Engineering Foundations: Reimagining An AI-First Product Engineering Lifecycle

FEATURING RESEARCH FROM FORRESTER

The Forrester Platform Engineering Capability Model

EXECUTIVE SUMMARY

While generative AI (GenAI) accelerates coding, engineering velocity remains stalled by the 'Toolchain Jungle,' a complex web of fragmented tools, operational friction, and degraded visibility. Mere speed in coding is insufficient if the underlying infrastructure remains siloed. This whitepaper examines these critical challenges and outlines Engineering Foundation Services (EFS), a unified, AI-enabled approach to strengthen engineering foundations and improve key delivery metrics.

Designed to modernize the 'Cradle to Grave' lifecycle, EFS orchestrates cognitive intelligence and autonomous agents to eliminate silos and automate complex workflows. This approach resolves the trade-off between speed and stability, enabling organizations to align high-velocity development with release reliability.

IN THIS DOCUMENT

INTRODUCTION

Most product engineering organizations face a contradiction: they are writing software faster than ever, yet they struggle to ship reliable products at scale. The bottleneck is rarely a lack of developer talent or a shortage of tools. The root cause is the foundation itself, an infrastructure built for a manual era, now buckling under the weight of AI-driven velocity.

Today's product teams inhabit a 'toolchain jungle.' Backlogs reside in one system, code in another, and pipelines operate in isolation, with testing and releases scattered across disparate platforms. In this fragmented landscape, 'swivel-chair integration,' where humans manually bridge the gaps between tools, becomes the default workflow. This operational friction bleeds into critical KPIs: cycle times elongate, defect density rises, and mean time to recovery (MTTR) slips, often without a clear root cause visible to leadership.

THE AI MULTIPLIER EFFECT

The rapid adoption of GenAI has intensified this challenge. While AI accelerates individual tasks like coding and testing, it cannot resolve the structural disconnects that degrade flow across the lifecycle. In fact, applying AI to a broken foundation creates a phenomenon known as 'Agentic Chain Depletion.'. When the underlying toolchain is fragmented, AI agents lose context, hallucinate due to poor data visibility, or fail to execute complex, multi-step workflows.

To survive the shift to an AI-first reality, organizations must dismantle the toolchain jungle. The imperative is to replace fragmented point solutions with a Unified Engineering Foundation that transforms the infrastructure from a liability into a coherent, autonomous delivery engine.

THE STATE OF ENGINEERING: COMPLEXITY VS. CAPACITY

Modern product engineering faces a critical deficit: the complexity of the environment has outpaced human capacity to manage it. A widening gap between ambition and execution is emerging as organizations scale cloud-native architectures and pivot toward AI-first engineering.

While multi-cloud strategies deliver scalability, they introduce exponential coordination and governance challenges. Three structural fractures in the engineering landscape compound this complexity:

1. **The 'Toolchain Jungle' Reality:** Most enterprises are trapped in a maze of disconnected tools. Driven by organizational silos and uncoordinated tech adoption, teams operate across redundant, disconnected systems. This fragmentation creates a visibility blackout, making release cycles difficult to coordinate and introducing operational friction that directly degrades delivery reliability.
2. **The Regulatory Burden:** Rising regulatory expectations demand rigorous traceability and auditability across the pipeline. In a fragmented landscape, enforcing these standards manually is not feasible, necessitating a structural shift to compliance automation and policy-as-code.
3. **The Experience Imperative:** Experience-Driven Engineering has moved from an interface-level concern to a system-level necessity. Organizations now recognize that velocity depends entirely on the 'Developer Experience' (DevEx). If engineers are forced to fight their tools rather than build features, productivity collapses and innovation stalls.

THE OPERATIONAL COST OF FRAGMENTATION

While the market shifts toward AI and cloud-native models, the operational reality is riddled by complex challenges. Failing to manage the 'jungle' complexity manifests in four specific core business issues:

1. **Data Silos and Integration Gaps:** The proliferation of tools like Jira, Trello, VSCode, GitHub, Jenkins, and countless others has created a disconnected ecosystem where data does not flow. Key issues include:
    - **Integration Gaps:** Data does not flow between planning and execution
    - **Visibility Blackouts:** Leadership lacks a unified view across teams and functions
    - **Manual Error Injection:** Redundant data entry across outdated systems increases error rates and slows decision-making
2. **The 'Agentic Chain Depletion' Syndrome:** A critical challenge emerges when organizations layer AI agents onto a broken foundation. Without a unified context, the automation logic quickly erodes.
    - **Context Loss:** Agents fail to execute complex workflows because they cannot 'see' across the full toolchain
    - **Inefficiency Multiplier:** Instead of accelerating delivery, disjointed AI agents compound inefficiencies, requiring more human intervention rather than less

3. **KPI Degradation:** Fragmented processes directly attack the metrics that matter. The friction results in:
    - **Slipping Velocity:** Longer cycle times and delayed releases.
    - **Quality Erosion:** Higher defect density and bug leakage.
    - **Customer Impact:** Reduced satisfaction due to slower feature rollout.
4. **The Talent and Efficiency Tax:** Siloed knowledge and manual governance drain human capital, impacting both retention and productivity:
    - **Talent:** Siloed knowledge creates 'onboarding cliffs,' slowing the ramp-up time for new hires.
    - **Audit Fatigue:** The lack of automated traceability forces teams into manual audit cycles, increasing risk and reducing productive engineering hours, in regulated industries like banking and healthcare

THE ARCHITECTURE OF UNIFIED FOUNDATIONS

Addressing these market fractures requires a structural reorganization of engineering functions. A modern, AI-first foundation operates on three strategic pillars, consolidating 11 core engineering functions into a cohesive, intelligent system.

PILLAR 1: UNIFIED EXPERIENCE (THE SINGLE PANE OF GLASS)

Functions: One Engineering Devspace (OEDS) | Onboarding and Subscription | Knowledge Management | Source Control

The first priority is reducing friction at the edge. Instead of forcing teams to toggle between disparate tools, a unified foundation abstracts these functions into a One Engineering Devspace (OEDS). This ensures that a new developer can provision their environment, access documentation, and push their first commit from a single location, drastically reducing onboarding time and eliminating the 'context switching' tax.

PILLAR 2: AUTONOMOUS ORCHESTRATION (CLOSING THE LOOP)

Functions: DevSecOps | Release Management | Verification & Validation (VnV) | Sprint Execution

Traditional pipelines rely on manual gates. An AI-first foundation replaces these with Agentic Workflows. This pillar governs DevSecOps and Release Management, using autonomous agents to trigger tests, validate builds, and execute sprints. Agents make data-driven decisions, such as automatic rollbacks and test case generation, reducing operational friction that slows release cycles.

PILLAR 3: COGNITIVE INTELLIGENCE (PREDICTIVE VISIBILITY)

Functions: Backlog Management | Issue Triaging | Observability

Data trapped in silos creates blind spots. A unified foundation pulls telemetry from backlogs, issue triage, and observability tools to develop a predictive intelligence layer. Instead of reacting to outages, the system correlates complexity with historical defect rates to predict bottlenecks before code is written.

THE STRATEGIC SOLUTION: TECH MAHINDRA EFS

Organizations are actively investing to achieve this architecture, seeking platforms that deliver modular innovation and seamless scalability. However, attempting to piece together disparate tools to achieve this unity often results in greater complexity.

Tech Mahindra's Engineering Foundation Services (EFS) resolves this paradox. It serves as a purpose-built, customizable platform that operationalizes the architecture described above. By unifying engineering functions through a 'single pane of glass' and infusing them with agentic frameworks, EFS transforms the fragmented toolchain into a streamlined, autonomous delivery engine.

ENGINEERED FOR RESILIENCE: THE CORE FRAMEWORKS

Unified Engineering Foundations are not a monolith; they represent the convergence of four advanced methodologies. The EFS platform creates resilience and speed by integrating these core frameworks:

1. **Cognitive and Agentified Engineering** *(The Intelligence Layer)*	**Cognitive:** AI-powered insights and predictive analytics to anticipate bottlenecks. **Agentified:** Autonomous agents for issue triaging, test generation, and compliance. **Autonomous:** Self-healing systems for automated remediation.
2. **Cloud-Native and Multi-Cloud Ops** *(The Infrastructure Layer)*	**Service Mesh:** Flexible deployment and orchestration across hybrid cloud environments. **AI-Driven Autoscaling:** Dynamic resource optimization and load balancing. **Compliance Automation:** Policy-as-Code for automated, continuous reporting.
3. **Experience-Driven Engineering** *(The Usability Layer)*	**Unified Portals:** Centralized access to docs, repos, APIs, and environments (OEDS). **Journey Analytics:** Embedded analytics to identify and resolve process friction. **Experience Observability:** Real-time monitoring and feedback integration.
4. **Progressive Sustenance** *(The Reliability Layer)*	**Continuous Maintenance:** Ongoing corrective, adaptive, and preventive actions. **Predictive Monitoring:** AI-enabled monitoring and self-healing scripts. **Talent Harmonization:** Automated upskilling and knowledge transfer across teams.

FROM JUNGLE TO ORDER: THE IMPLEMENTATION ROADMAP

To transition from a fragmented toolchain to a unified foundation, Tech Mahindra follows a structured, phased approach designed to maintain business continuity while systematically dismantling operational friction.

EFS IMPLEMENTATION FRAMEWORK

1. **Phase 1: Baseline Assessment**
    1. *Action:* Analyze current toolchain friction.
    2. *Target:* Identify impacted KPIs (MTTR, Cycle Time).
2. **Phase 2: Navigation Path**
    1. *Action:* Prioritize functions based on impact.
    2. *Target:* Create a strategic roadmap through the tool jungle
3. **Phase 3: Design and Construct**
    1. *Action:* Build the standards-aligned platform to unify engineering functions
    2. *Target:* Infuse AI and Human-in-the-Loop capabilities.
4. **Phase 4: Controlled Rollout**
    1. *Action:* Execute orchestrated cutover.
    2. *Target:* Ensure zero-disruption migration and business continuity.
5. **Phase 5: Progressive Sustenance**
    1. *Action:* Activate self-healing scripts.
    2. *Target:* Achieve continuous optimization and system reliability

REAL-WORLD IMPACT: EFS IN ACTION

The following deployments demonstrate how our EFS transforms fragmented liabilities into strategic assets, delivering measurable value across diverse industries and geographies.

Industry / Client	The Friction (Challenge)	The EFS Fix (Solution)	The Business Impact
Financial Lending(UK/ Ireland)	Fragmented toolchain and heavy compliance burden, slowing release cycles	Unified DevSecOps platform with AI-driven compliance reporting	Faster release cycles and improved audit readiness
Data Forensics (ANZ)	Manual test generation and delayed bug detection increasing the cycle time	Shift-left VnV through the SDET model and AI-led test automation	Accelerated certification workflows and earlier defect discovery
Energy Services (USA)	Disparate developer workflows and complexity stalling talent onboarding	Unified Dev Workspace and cloud-native modernization.	Simplified toolchain and rapid developer onboarding
Reinsurance Firm (Global)	Slow release velocity and siloed knowledge hindering collaboration	Integrated DevSecOps, knowledge management, and release orchestration with AI-powered insights.	Reduced operational friction and improved knowledge sharing
Swiffter.io (Platform)	Need for enterprise-grade, AI-accelerated application development	Support the entire engineering lifecycle with a team of AI agents and direct artifact manipulation	Consistency, transparency, and high-speed delivery

ILLUSTRATING REAL AND HYPOTHETICAL EXAMPLES

1. Banking Developer Experience Platform
Financial Services | Real World

Challenge

Manual compliance checks, tool silos, and onboarding gaps created mounting technical debt and slowed delivery.

EFS Approach

- Single developer portal across the lifecycle
- Policy-as-code for automated compliance
- AI-led task execution and reporting
- Secure, AI-assisted onboarding sandboxes

Impact

- Faster time-to-market
- Lower compliance costs
- Higher developer CSAT

2. Industrial IoT Modernization
Manufacturing | Hypothetical

Challenge

Fragmented legacy monitoring led to reactive maintenance and costly downtime

EFS Approach

- Unified asset monitoring platform
- Predictive maintenance AI
- Phased migration to ensure continuity
- Self-healing sustenance capabilities

Impact

- 4× maintenance cost savings
- Defect resolution: days –> minutes
- 360° asset visibility

3. Connected Mobility Services
Automotive OEM | Real World

Challenge

Disconnected vehicle health and service scheduling systems created operational inefficiencies.

EFS Approach

- Unified CCS and ROC platform
- AI-driven predictive analytics
- Automated service scheduling

Impact

- 25–30% higher dispatch reliability
- Improved safety outcomes
- Unified maintenance visibility

CONCLUSION

The disconnect between accelerating code generation and stalling delivery pipelines are defining the challenges in modern engineering. As organizations race toward AI-first models, the ‘Toolchain Jungle has become a strategic risk to systemic reliability. Bridging this gap demands a structural shift from fragmented point solutions to a unified, autonomous foundation

Tech Mahindra’s EFS transforms the ‘Toolchain Jungle’ into a unified competitive advantage.By operationalizing a ‘single pane of glass’ and infusing agentic intelligence across the lifecycle, EFS turns foundational infrastructure into a driver of velocity rather than a drag on innovation. This transformation is the prerequisite for scaling at speed in the cognitive era, ensuring that engineering teams stop fighting their tools and start delivering value.

ABOUT THE AUTHOR

Sneha Ranjan
Global Growth Leader – Digital Engineering, Tech Mahindra

Sneha Ranjan leads the global growth function for Digital Product and Platform Engineering at Tech Mahindra. She brings deep expertise across Technology Go-To-Market, CXO advisory, product marketing, and business strategy, and has spearheaded multiple large-scale transformation initiatives across organizations. With a strong blend of strategic and execution leadership, Sneha has played a pivotal role in shaping growth narratives and scaling digital engineering portfolios globally.

She holds an MBA from IIM Calcutta and a bachelor’s degree in mechanical engineering from BIT Mesra.

Sanjay Datta
Chief Architect – Digital Product and Platform Engineering, Tech Mahindra

Sanjay Datta is an industry veteran with over two decades of experience, currently leading the Design and Architecture Office for Digital Product and Platform Engineering at Tech Mahindra. His experience spans multiple industries including Manufacturing, Logistics, Automotive, ISV/IISV, BFSI, Retail, and CPG. Sanjay is known for seamlessly integrating business strategy, technology, and architecture, and actively contributes to several enterprise architecture boards.

He holds an MCA from Banaras Hindu University and a B.Sc. in Statistics from the University of Calcutta.

BEST PRACTICE REPORT

The Forrester Platform Engineering Capability Model

Ensure Your Platform Engineering Program Has All Bases Covered

February 12, 2025

By Charles Betz, Manuel Geitz with JT Thykattil, Julie Mohr, Naveen Chhabra, Andrew Cornwall, Fiona Mark, Tony Plec, Christopher Condo, Janet Worthington, Melissa Chhay, Kara Hartig

FORRESTER®

Summary

The rapid evolution of cloud, agile, and DevOps and the increasing focus on employee experience (EX) are disrupting traditional IT operating models globally. Traditional team silos in areas like compute, storage, networking, and middleware struggle to meet market demands for innovation and employees' preferences for collaborative and responsive work environments. Technology leaders must adopt product-centric thinking in IT platform management to enhance service delivery. This report explores the challenges IT faces and the viable solution that platform engineering offers.

Traditional IT Organizations Struggle On Many Fronts

The world around technology teams is changing rapidly, and traditional IT operations are struggling to keep up. Disruptive trends such as the project-to-product shift, cloud, and agile/DevOps transformations, coupled with rising expectations from customers and employees, are pressuring technology leaders to transform their operations. Key issues include:

- **Impatience with I&O slowness.** Pressure is increasing on infrastructure and operations (I&O) teams to be faster, more responsive, and empathetic with internal customers and other stakeholders.
- **Automation and self-sufficiency.** Autonomous digital or IT product teams are becoming self-sufficient, reducing the need for traditional infrastructure engineers.
- **Questioning of established approaches.** Fast-moving digital teams show resistance to strict process frameworks like ITIL and IT service management.
- **Neglected digital employee experience (DEX).** Improving IT capabilities to enhance EX is crucial for engagement and retention.

How Platform Engineering Is A Solution

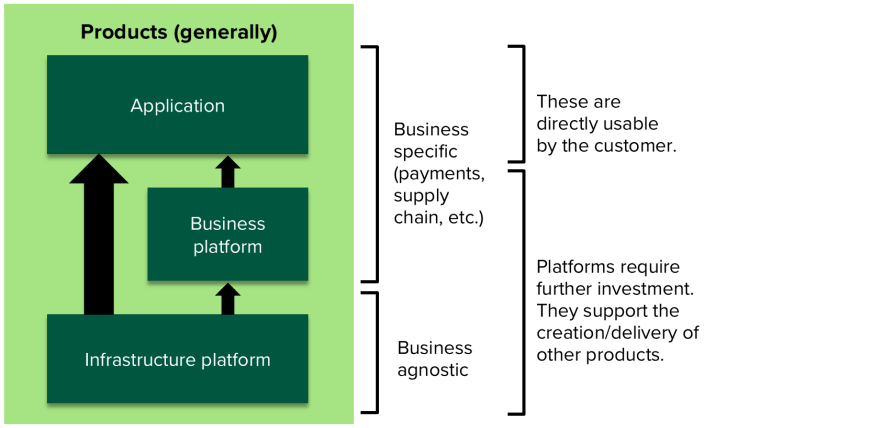
Platform engineering, grounded in product management principles, offers a solution to modernize IT operations. By injecting product thinking into platform teams, technology organizations can position themselves for the future. Key aspects of platform engineering include:

- **Product-centric perspectives.** Product teams emphasize customer and market understanding, business acumen, technical skills, and soft skills.
- **Whole-lifecycle capabilities.** IT platform teams represent a full-lifecycle design/build/run capability, focusing on user personas, quality assurance, and release management.
- **Customer empathy.** This includes adopting a human-centric approach to product management, considering the effects of decisions on broader value streams.
- **Infrastructure focus.** Platform is a broad concept, and there are business and technical platforms. Business platforms embed domain-specific APIs, data, and workflows, while technical platforms are industry agnostic. Infrastructure platforms

may support business platforms (see Figure 1). Platform engineering, the focus of this report and framework, is primarily concerned with infrastructure platforms.

Figure 1

Products And Platforms



Platform Engineering Requires Multiple Capabilities

Forrester defines a platform as:

> A product that supports the creation and/or delivery of other products.

Platform engineering (a bit of a misnomer, but the horse has left the barn) is based on the principles of product management and the product model applied to digital and IT systems. Forrester has compiled a capability model that includes frequently covered technical aspects and less frequently covered management capabilities. Forrester's capability model includes four level-one capabilities:

- **Engagement capabilities are the surface area of the platform.** They directly support the customer (usually development teams) and include developer portals, APIs, patterns, and other documentation and support.
- **Operational capabilities make things happen.** They are the execution that provisions and controls the base resources. They include policy, security, artifacts, control, automation, pipelines, observability, and resilience.

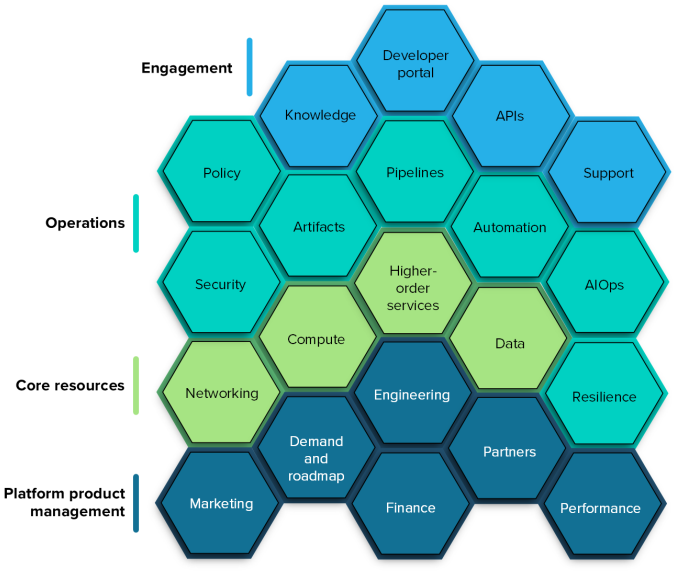
- **Base resources are what the platform delivers to the application teams.** They include compute, data, networking, messaging, and higher-order services (e.g., batch scheduling).
- **Platform product management capabilities manage the platform business.** These are too often overlooked, especially when platform engineering emerges from traditional I&O organizations with a historical focus on order-taking, ticketing, and engineering excellence. They include performance, marketing, roadmaps, partners, finance, and engineering.

A Deeper Look At Your Platform Capabilities

A capability map should never be mistaken for an organization chart (see Figure 2). It is an inventory of things you should think deeply about and ensure you have covered via your organizational resources, which might include not only dedicated organizations but also cross-functional processes, enablement teams, or other mechanisms. Consider the following types of platform engineering capabilities as you work through your platform strategy.

Figure 2

The Platform Engineering Capability Model



Your Engagement Capabilities Are How Your Customers Experience The Platform

They are your front door, so to speak. Your customers will discover your platform, onboard to it, provision it, interact with its APIs, leverage patterns for security and performance, and call for help via these capabilities (and no, there is no such thing as an entirely automated self-service platform). Consider the following detailed definitions and concerns for each:

- **Developer portal.** End users/developers need to be able to discover the platform and its services. Managing your platform like a product means that you understand users' onboarding journey and invite them to be part of the process of defining and even contributing to developer platform capabilities. They will expect easy, frictionless authorization and access, with few, if any, human-in-the-loop workflow-based approvals. Once provisioned and actively developing, they will need information about the ongoing status of the services they are consuming. Usually, larger organizations have a service catalog or portal capability for IT services. If this does not exist, you must fund and create it. Developer-focused portals (e.g., Spotify Backstage, Harness Internal Developer Portal, Atlassian Compass) are gaining popularity. Toyota of North America includes consumable blueprints, a discoverable software catalog, education and training resources, and operational reporting for FinOps and other metrics in its developer portal.

- **APIs.** Access to platform services and resources is typically a two-stage process: initial provisioning (setting up accounts) and day-to-day demand (provisioning virtual machines, clusters, etc.). While setting up the account may require some human approvals, day-to-day demand requires API access. A platform that cannot provision, configure, and manage base resources via API is not a true platform. Typically, platforms support APIs to instantiate and configure needed resources such as processing nodes, data stores, queues, pipelines, and observability probes. There are significant API design questions. Many organizations generally have API engineering capabilities but may not have explored the nuances of supporting self-service provisioning. Architect and author Gregor Hohpe identifies various options: Supplier APIs can be accessed via patterns, wrapped, intercepted, and/or reported.

- **Knowledge.** Users of the platform require ready access to knowledge for their enablement (e.g., with will patterns for appropriate use). How will these be created and maintained? Typically, a wiki will be used for core system quick starts and how-tos. Document patterns as code and manage them via source control. Define the processes, roles, and responsibilities for those in charge of these resources. Saying that they are everyone's responsibility is tempting, but that does not work at scale

or in the long run.

- **Support.** Platforms are typically highly leveraged. End users building tenant applications may not understand the system. The system may not behave as expected. For these and other reasons, you will likely need some level of on-call support. Human contact is required, even in the age of ChatGPT. Most organizations have ticketed support management (e.g., BMC Software, ServiceNow). This may be used to support the base platforms, and tenant applications may leverage it. Fewer have a robust major incident/critical event management capability, which is essential, leveraging products such as PagerDuty or Everbridge.

Your Operational Capabilities Are Where The Rubber Meets The Road

The focus for many platform engineering architectures and frameworks is the operational capabilities, especially those that are more technical. While there are many kinds of infrastructure platform components, the fundamental DevOps chain capabilities appear in most platform engineering discussions, including here:

- **Policy.** Deployments and operational architectures must be controlled for governance and policy. Increasingly, this is done as code (e.g., through Open Policy Agent and similar approaches). Required design patterns, configurations, and hardening standards should all be checked. Are software-bill-of-materials (SBOM) checks increasingly mandatory? What are the consequences if they fail? If there is a change management process, how is risk calculated? Are chaos tests recommended or required by policy?

- **Security.** The platform's direct (administrative/developer) users must be identified and authorized for their construction activities. Furthermore, the products/applications they are building will require identity and access services (which might be quite different from the services controlling admin access to the platform). Which are you supporting? For example, determine if common directory services are available to administrators, if there is privileged access management, and if multifactor authentication, single sign-on, and/or directory services are available for end users of the tenants. The pipeline (also see below) must include security testing such as software composition analysis, SBOM generation, and static application security testing.

- **Pipelines.** Applications (aka workloads) are installed on resources once provisioned. Therefore, infrastructure platforms may include a full set of development pipeline resources, typically including source control; build, test,

and integration capabilities; access to various third-party quality tools; artifact management (mentioned separately here); and deployment automation including feature flagging.

- **Artifacts (generally).** Most developer-oriented platforms will provide access to source control and package management (perhaps via proxying cloud services like GitHub or GitLab). Will the platform store deploy artifacts or packages via JFrog Artifactory, Sonatype Nexus, or Cloudsmith? What services will it make available? For example, will it perform software composition analysis when new artifacts are added to the artifact repository, enforce security policies, and create SBOMs?

- **Automation.** Base resources require provisioning, configuration, and control, generally termed infrastructure automation. What architecture is used for runtime provisioning? Is it Terraform or hyperscaler specific? Does the platform provide a proxy layer to a cloud provider? Once initially provisioned, configuration may be a separate concern (e.g., with Red Hat, Chef, or Perforce Software [Puppet]), which can also control for drift. There is wide variation, which depends on technical feasibility.

- **AIOps.** How is the platform itself monitored and observed, and how are operational insights generated? What is the relationship between AIOps and action (e.g., support)? What services (e.g., monitoring, logging, tracing) are available to tenant applications? How is user experience understood? An application performance management or AIOps tool might be available as part of the platform for real-time insights that span platforms and encompass the whole IT estate, with results communicated with diverse personas in various ways, like being published on the developer portal.

- **Resilience.** As James Cohen notes, "Rare things become common at scale." How is the platform itself managed for resilience, availability, and learning? For example, site reliability engineers might have a specific function in defining the platform approach, leading major incident response and retrospectives, and reviewing operations. A retrospective might lead to identifying a risk for which a chaos engineering approach might be a control.

In the book, Platform Engineering: A Guide for Technical, Product, and People Leaders, authors Camille Fournier and Ian Knowland note: "If your platform is having more than five business-impacting events a week, then it's not providing a stable foundation for the business … Many great SRE practices work much better when led by passionate practicing engineers. These engineers excel at high-impact incident management, consulting on service-level objectives and chaos

engineering, and running game days, production readiness reviews, rigorous incident postmortems, and even weekly operational meetings."

Core Resources Are Your Value Proposition

There are numerous choices, from on-premises bare metal to Kubernetes and function as a service in the cloud. The platform must curate what it makes available — well-crafted selections that limit needless variation and sprawl are essential to the platform strategy. At the level of infrastructure, the common denominator is generality across business verticals (e.g., the same basic platform patterns might be used in both financial services and retail).

- **Compute.** Most platform engineering efforts center on either public or private cloud. Given the expectation that platforms are automated, seeing how it would otherwise work is difficult.
- **Data.** What core data storage and transport services are available? There is wide variation that the platform team must curate: SQL, NoSQL, graph, vector, data fabrics, analytics, AI, generative AI (genAI), etc. Backup and resilience strategies are essential.
- **Networking.** At scale, explicit networking provisioning and configuration are essential platform capabilities.
- **Higher-order services.** What other services will you curate for the platform? Workflow, automation, authentication, authorization, backup, scheduling, platform as a service (PaaS), software as a service (SaaS), etc. There is wide variation that must be curated.

Platform Product Management Ties It All Together In Your Customer's Best Interest

Platforms are products (we will keep saying this) and need to be managed as such. Many traditional I&O organizations trying to pivot to the platform model are struggling with the nontechnical aspects: customer empathy, financial management, user demand, and even internal marketing. Pivoting to the platform model is not simply a matter of renaming:

- **Marketing.** Marketing is more than just a developer portal. You need a go-to-market strategy (yes, this applies to internal platforms as well). Key stakeholders need to be informed and updated about the platform's progress, offerings, and so forth. You may need to seek out your organizational change management professionals.

Product engagement includes developer relations, market research, focus groups, sentiment surveys, creation of a guild/community of interest, publishing of your roadmap for comments, and even posters in breakrooms to establish awareness (in partnership with your corporate communications team).

Smruti Patel, VP of engineering at Apollo GraphQL notes (in Fournier and Nowland): "A year in, we had less than 5% of production traffic on our platform. … We realized that we had left the adoption of the platform to a 'build it and they'll come' mindset."

- **Demand and roadmap.** You must not become a feature factory just building things in response to customer demand — the customers outnumber you badly. So you need a product vision and strategy. How will you decide on new features, components, and services? How will the platform as a product engage with end users for feedback and suggestions for new features? How is the platform's own roadmap managed and coordinated with key stakeholders (e.g., your developer customers, enterprise architecture, security, IT finance). How do you gracefully sunset legacy functionality and publicize its eventual end (deprecation)?
- **Engineering.** Your platform will live and die by the skills and happiness of its staff engineers. How will the platform's core engineering be staffed? What are the roles and expectations? It should be full-time dedicated staff because part-time or overload commitment are antipatterns. What roles do you need? How many? What are expected career progressions?
- **Finance.** The most mature platforms operate as businesses within the business. How is the platform funded? Is any form of cost recovery/chargeback anticipated? Chargeback is difficult to implement but is garnering more interest lately. Alternatively, platforms might have a usage-based funding approach, with the platform challenged to attract more users to unlock more funding but no specific charging mechanism used. Asset management, classic IT finance, and FinOps may all be required services to enable this. The FinOps capability may itself be run as a platform team offering services across the organizations.
- **Partners.** Which major external partners (e.g., hyperscaler cloud providers, SaaS, global systems integrators) and which internal partners (e.g., an internal I&O organization with its own data centers) will support the platform? Providers include Amazon; Google; Microsoft; PaaS and SaaS providers such as ServiceNow, Pegasystems, and Salesforce; classic internal service providers like I&O organizations; as well as governance groups like security and risk, enterprise architecture, and IT finance.

- **Performance.** Finally, accountability is key. As a product, how will the business performance of the platform be managed and communicated to senior leadership? What objectives and key results will define the platform's success? These include programmatic priorities (objectives) as measured by key results such as usage metrics, customer satisfaction, developer experience, Employee Net Promoter Score℠ (eNPS), uptime, and creation/deployment of new features in a timely way. Note that performance is interpreted broadly in a business sense, not just technically like response time or availability.

  Fournier and Nowland suggest questions such as: How much time or money are you saving platform users, what percentage of the potential user base is voluntarily adopting this platform, and what's the customer satisfaction score for the platform? They will help you understand the benefits of adopting the platform, customer demand, and customer opinion about the platform.

FORRESTER®

We help business and technology leaders use customer obsession to accelerate growth.

FORRESTER.COM

Obsessed With Customer Obsession

At Forrester, customer obsession is at the core of everything we do. We're on your side and by your side to help you become more customer obsessed.

Research

Accelerate your impact on the market with a proven path to growth.

- Customer and market dynamics
- Curated tools and frameworks
- Objective advice
- Hands-on guidance

Learn more.

Consulting

Implement modern strategies that align and empower teams.

- In-depth strategic projects
- Webinars, speeches, and workshops
- Custom content

Learn more.

Events

Develop fresh perspectives, draw inspiration from leaders, and network with peers.

- Thought leadership, frameworks, and models
- One-on-ones with peers and analysts
- In-person and virtual experiences

Learn more.

Contact Us

Contact Forrester at www.forrester.com/contactus. For information on hard-copy or electronic reprints, please contact your Account Team or reprints@forrester.com. We offer quantity discounts and special pricing for academic and nonprofit institutions.

Forrester Research, Inc., 60 Acorn Park Drive, Cambridge, MA 02140 USA
Tel: +1 617-613-6000 | Fax: +1 617-613-5000 | forrester.com

ABOUT TECH MAHINDRA

Tech Mahindra (NSE: TECHM) offers technology consulting and digital solutions to global enterprises across industries, enabling transformative scale at unparalleled speed. With 150,000+ professionals across 90+ countries helping 1100+ clients, TechM provides a full spectrum of services, including consulting, information technology, enterprise applications, business process services, engineering services, network services, customer experience & design, AI & analytics, and cloud & infrastructure services. It is the first Indian company in the world to have been awarded the Sustainable Markets Initiative's Terra Carta Seal in recognition of actively leading the charge to create a climate and nature-positive future. Tech Mahindra is part of the Mahindra Group, founded in 1945, one of the largest and most admired multinational federations of companies.

For more information on how TechM can partner with you to meet your scale at speed imperatives, please visit Tech Mahindra | Scale at Speed