Hype Cycle for Agile and DevOps, 2023

Published 27 July 2023 - ID G00792717 - 116 min read

By Analyst(s): Keith Mann, Joachim Herschmann, Akis Sklavounakis

Initiatives: Software Engineering Practices; Build and Deliver New Digital Products/Experiences to Drive Business Results; Software Engineering Technologies

Leading software engineering teams continuously improve their Agile and DevOps practices. Software engineering leaders should use this research to better understand the innovations helping to advance their capabilities.

Additional Perspectives

- アジャイルとDevOpsのハイプ・サイクル：2023年
(19 October 2023)

Analysis

What You Need to Know

Agile and DevOps are core capabilities that help modern software engineering organizations deliver customer value quickly and reliably. These capabilities combine technical ideas, such as automation and delivery pipelines, with human-centric concepts like collaboration, community and empowerment. Also, Agile and DevOps enable organizations to continuously experiment, learn and improve, so innovations continue to emerge, as seen in this Hype Cycle.

Few organizations have a consistently high level of Agile and DevOps maturity. Software engineering leaders must support their most advanced teams while guiding — or even restarting — the journey of others. Organizations will never face the same challenges in the exact same sequence. Innovations that could further improve already functioning areas may be inappropriate for less mature teams. Accordingly, software engineering leaders should look for ideas that are more proven to help new and struggling teams advance their capabilities.

The Hype Cycle

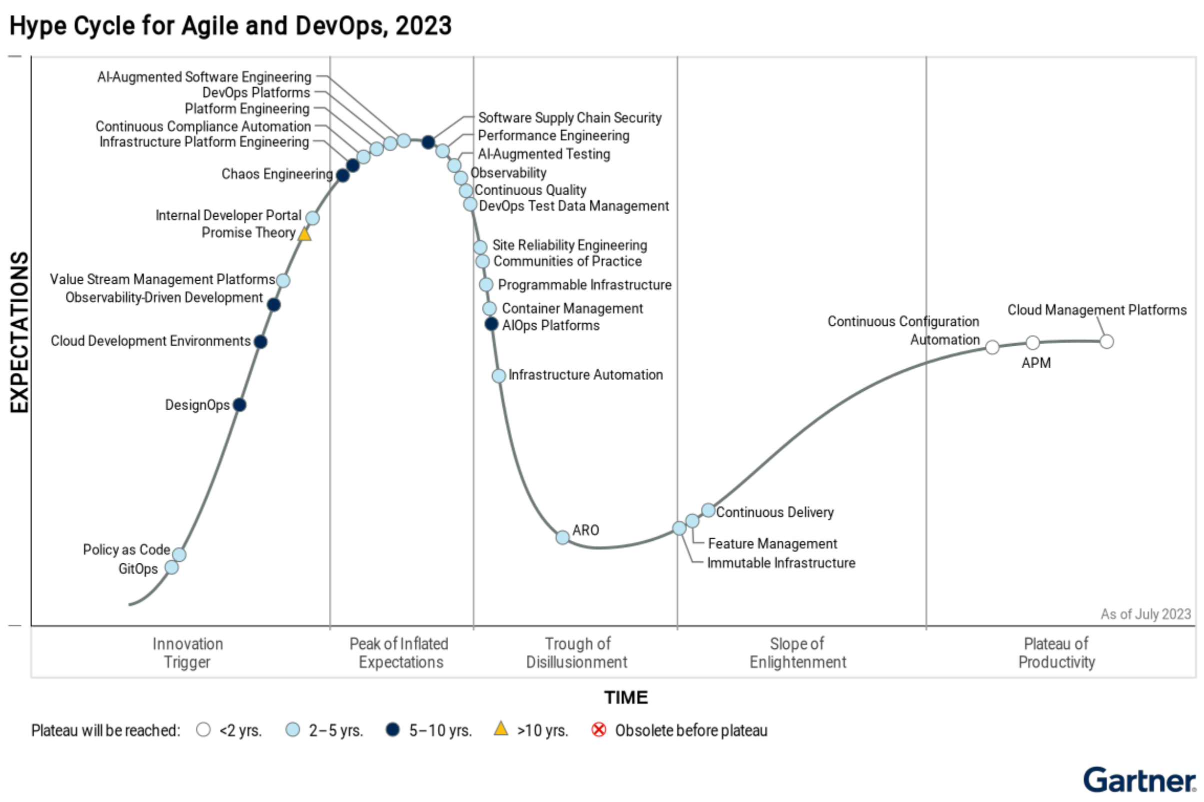
Making the most of scarce Agile and DevOps talent — that is, increasing developer productivity — remains a priority for software engineering leaders and a prominent driver of innovation. The demand for new software continues to exceed developer capacity in most organizations.

Platform engineering has soared to the peak of the Hype Cycle, and emerging innovations such as internal developer portals have appeared. AI is beginning to be integrated into software development practices. Despite the need for AI solutions and the growing enthusiasm regarding these tools, these innovations may not yet be suitable for widespread adoption. Instead, software engineering leaders may have to choose to pilot these emerging innovations while relying on more mature solutions like continuous configuration automation (CCA) to fill the gaps.

Many organizations have pursued excellence in the DevOps Research and Assessment (DORA) metrics — such as change failure rate and mean time to recover — through proven innovations like continuous delivery and feature management. Newer innovations — for example, chaos engineering and site reliability engineering — can help push DORA metrics even higher, but they may provide diminishing returns, especially considering their cost and complexity. Software engineering leaders should set targets for these metrics based on customer demands — instead of improving them based on arbitrary goals. DORA metrics don't address important factors like developer experience or value delivery.

Quality and security measures are reflected in this Hype Cycle through innovations like AI-augmented testing, DevOps test data management and continuous quality. Close collaboration between security stakeholders and software engineering teams is key to developing secure software.

Figure 1: Hype Cycle for Agile and DevOps, 2023



The Priority Matrix

The Priority Matrix maps the time to maturity of technologies and frameworks in an easy-to-read grid format. It answers two high-priority questions:

- How much value will an organization receive from an innovation?
- When will the innovation be mature enough to provide this value?

Application performance monitoring (APM) and CCA offer enormous benefits and are within two years of mainstream adoption. CCA greatly enhances deployments' efficiency and reliability, and APM provides valuable insights into the behavior of software in production. Software engineering leaders must evaluate APM and CCA vendors — particularly the latter, as CCA requires training and even changes to the organizational culture. These two requirements take time, so software engineering leaders must start preparing them as soon as possible.

Table 1: Priority Matrix for Agile and DevOps, 2023

(Enlarged table in Appendix)

Benefit	*Years to Mainstream Adoption*			
	Less Than 2 Years	***2 - 5 Years***	***5 - 10 Years***	***More Than 10 Years***
Transformational		AI-Augmented Software Engineering; Observability; Platform Engineering; Site Reliability Engineering	Software Supply Chain Security	
High	APM; Continuous Configuration Automation	AI-Augmented Testing; ARO; Communities of Practice; Container Management; Continuous Delivery; Continuous Quality; DevOps Platforms; Feature Management; GitOps; Infrastructure Automation; Internal Developer Portal; Performance Engineering; Policy as Code; Programmable Infrastructure; Value Stream Management Platforms	AIOps Platforms; Cloud Development Environments; DesignOps; Infrastructure Platform Engineering; Observability-Driven Development	Promise Theory
Moderate		Continuous Compliance Automation; DevOps Test Data Management; Immutable Infrastructure	Chaos Engineering	
Low	Cloud Management Platforms			

Source: Gartner (July 2023)

Off the Hype Cycle

The following changes have been made to 2023 Hype Cycle for Agile and DevOps:

- Autonomous testing has been replaced by AI-augmented testing.
- Digital platform conductor tools are now covered in Hype Cycles focusing on infrastructure and operations (I&O).
- Dojos have been removed from this Hype Cycle because they became obsolete before reaching maturity.
- Platform ops has been replaced by platform engineering, which is covered in this Hype Cycle.

- Securing development environments has been merged into the software supply chain security (SSCS) innovation.
- Software-defined infrastructure has been removed from this Hype Cycle because it became obsolete before reaching maturity.
- In the Magic Quadrant for DevOps Platforms, value stream delivery platforms are covered as DevOps platforms.

On the Rise

GitOps

Analysis By: Paul Delory, Arun Chandrasekaran

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Early mainstream

Definition:

GitOps is a type of closed-loop control system for cloud-native applications. The term is often used more expansively, usually as a shorthand for automated operations or CI/CD, but this is incorrect. According to the canonical OpenGitOps standard, the state of any system managed by GitOps must be: (1) expressed declaratively, (2) versioned and immutable, (3) pulled automatically, and (4) continuously reconciled. These ideas are not new, but new tools and practices now bring GitOps within reach.

Why This Is Important

GitOps can be transformative. GitOps workflows deploy a verified and traceable configuration (such as a container definition) into a runtime environment, bringing code to production with only a Git pull request. All changes flow through Git, where they are version-controlled, immutable and auditable. Developers interact only with Git, using abstract, declarative logic. GitOps extends a common control plane across Kubernetes (K8s) clusters, which is increasingly important as clusters proliferate.

Business Impact

By operationalizing infrastructure as code, GitOps enhances availability and resilience of services:

- GitOps can be used to improve version control, automation, consistency, collaboration and compliance.
- Artifacts are reusable and can be modularized.
- Configuration of clusters or systems can be updated dynamically. All of this translates to business agility and a faster time to market.

- GitOps artifacts are version-controlled and stored in a central repository, making them easy to verify and audit.

Drivers

- **Kubernetes adoption and maturity:** GitOps must be underpinned by an ecosystem of technologies, including tools for automation, infrastructure as code, continuous integration/continuous deployment (CI/CD), observability and compliance. Kubernetes has emerged as a common substrate for cloud-native applications. This provides a ready-made foundation for GitOps. As Kubernetes adoption grows within the enterprise, so can GitOps, too.

- **Need for increased speed and agility:** Speed and agility of software delivery are critical metrics that CIOs care about. As a result, IT organizations are pursuing better collaboration between infrastructure and operations (I&O) and development teams to drive shorter development cycles, faster delivery and increased deployment frequency. This will enable organizations to respond immediately to market changes, handle workload failures better, and tap into new market opportunities. GitOps is the latest way to drive this type of cross-team collaboration.

- **Need for increased reliability:** Speed without reliability is useless. The key to increased software quality is effective governance, accountability, collaboration and automation. GitOps can enable this through transparent processes and common workflows across development and I&O teams. Automated change management helps to avoid costly human errors that can result in poor software quality and downtime.

- **Talent retention:** Organizations adopting GitOps have an opportunity to upskill existing staff for more automation- and code-oriented I&O roles. This opens up opportunities for staff to learn new skills and technologies, resulting in higher employee satisfaction and retention.

- **Cultural change:** By breaking down organizational silos, development and operations leaders can build cross-functional knowledge and collaboration skills across their teams to enable them to work effectively across boundaries.

- **Cost reduction:** Automation of infrastructure eliminates manual tasks and rework, improving productivity and reducing downtimes, both of which can contribute to cost reduction.

Obstacles

- **Prerequisites:** GitOps is only for cloud-native applications. Many GitOps tools and techniques assume the system is built on Kubernetes (frequently, they also assume that a host of other technologies are built on top of K8s). By definition, GitOps requires software agents to act as listeners for changes and help to implement them. GitOps is possible outside Kubernetes, but in practice K8s will almost certainly be used. Thus, GitOps is necessarily limited in scope.
- **Cultural change:** GitOps requires a cultural change that organizations need to invest in. IT leaders need to embrace process change. This requires discipline and commitment from all participants to doing things in a new way.
- **Skills gaps:** GitOps requires automation and software development skills, which many I&O teams lack.
- **Organizational inertia:** GitOps requires collaboration among different teams. This requires trust among these teams for GitOps to be successful.

User Recommendations

- **Target cloud-native workloads initially:** Your first use case for GitOps should be operating a containerized, cloud-native application that is already using both Kubernetes and a continuous delivery platform such as Flux or ArgoCD.
- **Build an internal operating platform:** This is the foundation of your GitOps efforts. Your platform should manage the underlying infrastructure and deployment pipelines, while enforcing security and policy compliance.
- **Embed security into GitOps workflows:** Security teams need to shift left, so the organization can build holistic CI/CD pipelines that deliver software and configure infrastructure, with security embedded in every layer.
- **Be wary of vendors trying to sell you GitOps:** GitOps isn't a product you can buy, but a workflow and a mindset shift that becomes part of your overall DevOps culture. Tools that expressly enable GitOps can be helpful; but GitOps can be done with nothing more than standard continuous delivery tools that support Git-based automation.

Sample Vendors

GitLab; Harness; Red Hat; Upbound; Weaveworks

Gartner Recommended Reading

Innovation Insight: Top 4 Use Cases for GitOps

Is Using GitOps-Style Automation With Kubernetes Right for Me?

How to Scale DevOps Workflows in Multicluster Kubernetes Environments

Designing and Operating DevOps Workflows to Deploy Containerized Applications With Kubernetes

Automate the Application Delivery Value Stream

Policy as Code

Analysis By: Paul Delory

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

Policy as code (PaC) languages express governance and compliance rules as code, so they can be enforced programmatically by automation tools. PaC languages are often domain-specific and declarative. With PaC, policies are treated as software, making them subject to version control, code review and functional testing. The most mature PaC tools can render any business logic in code. You can use them today to enforce infrastructure compliance, authorization, Kubernetes admission control, and more.

Why This Is Important

In the most mature automation pipelines, infrastructure and operations (I&O) engineers mostly spend time on optimization, governance and compliance. They no longer build infrastructure; that work has been automated and turned over to others. Now, the I&O function builds the guardrails around the infrastructure services that their end users consume. I&O must align with security and compliance teams. PaC brings policy enforcement into their automation pipelines, while preserving a separation of duties that mirrors a typical IT org chart.

Business Impact

Policy as code improves:

- **Security, compliance and automation:** PaC combined with infrastructure automation implements policies automatically, with implicit compliance guarantees.
- **Alignment of security and operations teams:** PaC allows security and compliance teams to interface directly with automation pipelines to ensure conformance.
- **Visibility and auditability:** PaC provides both documentation of policies and evidence they are being enforced.
- **Time and effort spent:** PaC means less toil for operators.

Drivers

- **PaC tooling:** Several dedicated PaC tools are now on the market, many of them are open-source. The Open Policy Agent, a Cloud Native Computing Foundation project, has become the *de facto* standard for PaC. Indeed, even some other PaC tools now use Open Policy Agent policies alongside or instead of their own policy engines.
- **Increasing regulation:** New regulations such as GDPR have increased both the difficulty of compliance and the pressure on compliance teams. PaC allows compliance teams and auditors to document their policies in detail, and to verify that they are being enforced.
- **Security breaches:** Similarly, a spate of newsworthy security breaches at public companies — caused by infrastructure misconfigurations — has put every IT organization's security and compliance practices under increased scrutiny. No I&O team wants its security failures to be the reason for its company getting negative headlines.
- **Growth of DevOps and DevSecOps:** More and more companies are embracing DevOps and DevSecOps — which means more and more companies are encountering the hard governance problems of automation. Many teams that implement infrastructure as code quickly are finding that they need better policy enforcement, and PaC can help.
- **Cloud optimization and cost control:** Beside their benefits for security and compliance, PaC tools can also be used to enforce the build standards for infrastructure, including budgets. In the public cloud, where oversized or unnecessary infrastructure incurs direct out-of-pocket costs, programmatically enforced policies can help to control spending.

Obstacles

- **Scarcity of downloadable content:** PaC tools will not gain real traction until they have an extensive library of community-generated content. Ideally, users would simply download the policies they need from a free, public repository, rather than having to write their own policies. Over time, as the user base expands and commercial offerings see increased adoption, PaC tools will reach a critical mass of downloadable content that supports real-world use cases.
- **Skill set:** Many I&O professionals lack the skills to operate automation and PaC tools effectively. Gartner clients routinely report that their automation and policy management are hindered primarily by people, not tools. PaC will magnify these existing skills challenges.
- **Organizational inertia:** PaC promises improved collaboration between I&O and security or compliance teams. But in some organizations, this change would be unwanted. Internal resistance of this kind will slow the rate, scope and scale of PaC initiatives.

User Recommendations

- **Start small:** Choose a pilot use case where PaC is likely to provide real business benefits, expanding to others once PaC has proven its value.
- **Upskill staff:** PaC languages are not always intuitive. Technical staff will need practice to reach proficiency.
- **Prioritize existing templates:** Focus your PaC efforts on use cases that have ready-made implementation templates — ideally, publicly available downloadable content. For example, almost every PaC tool on the market has a canned implementation of the CIS benchmarks.
- **Break down team silos:** Use PaC to build a common workflow for automation and policy enforcement that spans I&O, security and compliance teams.
- **Integrate PaC into automation pipelines:** Use PaC to build guardrails for automation tools, so that they cannot take actions that are out of compliance.
- **Measure before and after:** Use observability tools and value stream mapping to define your starting state, then compare it to the end state. Collect real data to quantify the value of PaC.

Sample Vendors

HashiCorp; Palo Alto Networks; Progress; Pulumi; Styra

Gartner Recommended Reading

Using ‘Policy as Code’ to Secure Application Deployments and Enforce Compliance

How to Protect Your Clouds With CSPM, CWPP, CNAPP and CASB

Innovation Insight for Continuous Compliance Automation

Innovation Insight for Cloud-Native Application Protection Platforms

Magic Quadrant for DevOps Platforms

DesignOps

Analysis By: Will Grant, Brent Stewart

Benefit Rating: High

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

DesignOps is a set of operational practices that enables design-team management and product-level delivery of design assets. The team management side stresses strategic alignment with business operations for the central design function and career development. The product delivery side combines user experience (UX), product management and technology operations to enable efficient and DevOps-compatible plans, estimates and processes that support quality, collaboration and ongoing innovation.

Why This Is Important

DesignOps introduces formalized approaches to governance, operations and people management. As a set of easy-to-use operational standards, DesignOps continues to gain in popularity. Digital product companies and agencies are discovering the tremendous value of a proven operational approach for UX team management and design delivery on product teams.

Business Impact

DesignOps represents the first widespread implementation of operational methods and techniques created for both designers and developers. DesignOps adds value during the creation and delivery of design assets. DesignOps practices should cover how teams are organized such that they can better support ongoing feature enhancement and idea generation without interrupting the continuous workflow of development teams.

Drivers

- **Innovation:** When coupled with DevOps, DesignOps leads to more innovative solutions. As a practice, DesignOps employs dual-track agile, which sets aside ongoing tracks of work dedicated to new discovery, idea generation and design exploration. This work acts as a constant source of evidence-based, multidisciplinary innovation.
- **Speed:** DesignOps reduces the time to market for major updates and incremental feature enhancements alike. Due to the concepts of continuous discovery and continuous delivery, developers engage in tech design, architectural explorations and proofs of concept (POCs) sooner than before, and with a deeper understanding of the overall vision.
- **Collaboration:** DesignOps increases communication and camaraderie between design and development teams. The design-development gap exists for many reasons, one of them being culture. DesignOps promotes multidisciplinary teams in workshop settings, design sprints or one-on-one “pairing and sharing” that promotes understanding, empathy and relationship building between these two crucially important groups.

Obstacles

- Few UX practitioners are educated in detailed planning and estimation, using a common work breakdown structure (WBS).
- Few product managers are trained in UX planning, estimating and tracking, and many of the design platforms lack robust change control solutions, although this is improving.
- Popular enterprise agile planning (EAP) tools are not designed with UX practitioners, activities and deliverables in mind (although this is changing), leading to resistance from UX teams to adopt tools they perceive as “for developers.”

User Recommendations

Software engineering leaders should:

- Encourage design and UX professionals to adopt a DesignOps practice to better manage the complete design life cycle.
- Ensure that the DesignOps practice covers the following three key aspects: how UX teams are organized, the tools and processes for delivering UX artifacts and how success is measured.
- Develop Agile training designed for UX professionals, available as part of the DesignOps practice.
- Determine the value of a DesignOps approach with a pilot program involving an existing high-performing team.

Following a successful pilot, application leaders and the pilot team members should:

- Engage in a productwide rollout that involves training, updated product plans and the allocation of one or more people to the role of design manager.

It should be noted that a successful rollout of DesignOps at the product level requires buy-in from product management, design and development teams, as well as robust logistical and administrative skills.

Gartner Recommended Reading

How Design, Development and Product Management Can Work Together Successfully

The 4 Secrets of Design-Led Companies

2023 Software Engineering Primer: Build and Deliver New Digital Products/Experiences to Drive Business Results

Keys to DevOps Success

Cloud Development Environments

Analysis By: Manjunath Bhat

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

Cloud development environments (CDEs) provide remote, ready-to-use access to a cloud-hosted development environment with minimal effort for setup and configuration. This decoupling of the development workspace from the physical workstation enables a low-friction, consistent developer experience. CDEs comprise elements of a traditional IDE, such as code editing, debugging, code review and code collaboration. They increasingly include ML-based coding assistants and integrate with DevOps platforms.

Why This Is Important

CDEs provide consistent, secure developer access to preconfigured development workspaces. This frees them from setting up their own local environments, eliminating the need to install and maintain dependencies, software development kits, security patches and plug-ins. CDEs are prepackaged with language tooling for multiple programming languages, enabling teams to write code for different application stacks with standardized and templatized workflows.

Business Impact

CDEs are becoming popular due to their seamless integration with Git-based repositories and continuous integration/continuous delivery (CI/CD) tools, thus enhancing developer productivity and developer experience. They also enable security and platform teams to govern and secure user access to development environments, even from personally-owned devices. This mitigates the security and operational risks. CDEs provide a scalable way to support a distributed workforce and hybrid work environments.

Drivers

Gartner predicts that by 2026, 60% of cloud workloads will be built and deployed using CDEs. Four factors are driving their increased adoption:

- Remote work and remote onboarding of software developers create a need for a frictionless onboarding experience. The ability to share the development environment among distributed team members makes remote debugging and pair programming easier.

- Organizations increasingly need custom configured hardware to support specific application architectures, such as Apple M1/M2 silicon for mobile app development or GPU support for data and analytics workloads.
- The ability to centrally manage, govern and secure development environments has become especially important to minimize the threat of software supply chain attacks. In addition, CDEs make it easier to support and secure bring-your-own-device use cases.
- Automating DevOps workflows introduces more plug-ins, extensions and API integrations, which makes the distribution of updates and configurations to local machines cumbersome and unreliable.
- Building modern, distributed applications such as mesh services, event-driven apps and Kubernetes applications introduce complexities, making it harder to develop and test on local machines.

Obstacles

- CDEs incur costs in addition to what an organization may already be paying for DevOps tooling. The cost can be prohibitive for teams that rely on open-source development tools for application development and delivery on local machines.
- Connectivity presents another obstacle. Poor or inconsistent internet speeds adversely affect developer experience. In many cases, developers simply like to use their own local machines to get their work done. That allows them to use their own plugins, editors and scripts, all of which will be difficult through a browser-based interface.
- Security and compliance policies can prevent the use of cloud for development in some organizations. This can rule out CDEs that rely on public cloud services. Note that CDEs don't necessarily have to be provisioned in public cloud environments.
- There may be resistance from developers since it may impede their ability to research, experiment and innovate using full or elevated permissions on local development machines.

User Recommendations

Software engineering leaders should:

- Pilot the use of CDEs when their teams are developing with cloud-hosted source repositories. Plan for downtime resulting from service outages, since CDEs rely on the remote development environment being up and running.
- Ensure that CDEs are part of an overall developer self-service platform with centralized governance. The benefit of centralized governance and developer agility can be a win-win for platform and product teams.
- Use CDEs as one of the “quick wins” to improve the onboarding experience for new developers and reduce ramp-up time.
- Work with security experts and enforce strong authentication and authorization policies to mitigate security risks. CDEs present an additional, high-value attack vector, as they become the pipeline through which intellectual property flows.

Sample Vendors

Amazon Web Services; Codeanywhere; Coder; GitHub; Gitpod; Google; JetBrains Space; Red Hat; Replit; Strong Network

Gartner Recommended Reading

Quick Answer: How to Create a Frictionless Onboarding Experience for Software Engineers

Infographic: Platforms and Tools to Scale the Delivery of High-Quality Software

Innovation Insight for Internal Developer Portals

Innovation Insight for ML-Powered Coding Assistants

Observability-Driven Development

Analysis By: Manjunath Bhat

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

Observability-driven development (ODD) is a software engineering practice that provides fine-grained visibility and context into system state and behavior by designing observable systems. ODD works by instrumenting code to unravel a system's internal state with externally observable data. As part of a shift-left approach to software development, ODD makes it easier to detect, diagnose and resolve unexpected anomalies early in the development life cycle.

Why This Is Important

Building observable systems can expedite issue resolution as observability data is a useful debugging aid. Designing for observability also amplifies the benefits of other resilience engineering practices, such as site reliability engineering (SRE) and chaos engineering. ODD enables software engineers and product owners to understand how the software is performing and how it is being used. However, the practices to institutionalize ODD are still emerging.

Business Impact

Built-in observability helps develop reliable software. ODD reduces the time to troubleshoot issues in production and preproduction environments. This feature helps software engineering teams reduce cycle time, making developers more productive during testing. ODD also helps ship code confidently since observable data makes it easier to troubleshoot production issues, thus minimizing downtime. In business terms, it translates to compliance with regulatory and contractual SLAs.

Drivers

- Organizations adopting SRE and chaos engineering practices need the data provided by observability design. These practices are fundamentally data-driven and support software reliability.

- User experience (UX) is subjective and difficult to measure. It depends on various factors that span the complete technology stack. Some factors, such as last-mile network connectivity, affect UX but are difficult to optimize unless systems are designed to be observable and extract data related to such signals, like response rates and latency.

- Mobile and edge environments present unforeseeable challenges since applications can run on potentially unknown devices and untrusted environments. These production environments can significantly differ from the local environments used to test the application. Therefore, ODD can help capture and investigate the “unknown unknowns” at runtime.

- Distributed systems exhibit emergent behavior and are difficult to predict. Therefore, the need for observability increases as issues can arise due to unexpected component interactions. Troubleshooting issues in distributed applications requires fundamentally different techniques than monolithic or client and server applications. Building observability narrows down the problem domain and helps engineers inspect the problematic component.

- OpenTelemetry is an open standard that has seen increased open-source implementations and vendor adoption over the past year. This has improved consistency when adopting ODD across products. Organizations also favor using open standards and protocols to ingest telemetry data. This makes ODD accessible to most developers even when they lack the budget for commercial tools.

Obstacles

- Monitoring and, by association, observability, is commonly viewed as an operational responsibility. Software engineers often lack expertise with observability as a practice and with the tools and frameworks used to implement observability.
- Software engineering leaders must overcome the perception that observability can be achieved merely by implementing an observability tool. Observability is a domain that must be designed and built into systems to ensure that it provides business benefits.
- A piecemeal approach to observability may thwart efforts to adopt ODD at scale. As a standard practice, ODD provides greater benefits when all system components are designed with an observability mindset. For example, distributed tracing requires that all components contributing to the trace are "instrumented" and propagate context for diagnosing response-time issues. Platform teams entrusted with driving ODD at scale can help overcome this obstacle.

User Recommendations

- Adopt ODD as a standard software engineering practice to handle unexpected and unforeseeable system behaviors and anomalies.
- Make observability a priority since it provides critical insights to resolve production errors. It provides continual feedback to understand how the software is being used.
- Keep up with the pace of innovation in observability by using open standards and open-source technologies, such as OpenTelemetry.
- Be wary of vendor hype regarding observability merely to provide access to logs, metrics and traces. Use ODD as a fundamental software engineering practice that improves UX and resilience with granular insight into system state and behavior.

Sample Vendors

Chronosphere; Datadog; Dynatrace; Grafana; Honeycomb; Logz.io; New Relic; observIQ; ServiceNow; Splunk

Gartner Recommended Reading

How to Identify and Resolve Application Performance Problems

Improve Software Quality by Building Digital Immunity

Value Stream Management Platforms

Analysis By: Hassan Ennaciri, Akis Sklavounakis

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Early mainstream

Definition:

A value stream management platform (VSMP) is a platform that seeks to optimize end-to-end product delivery and improve business outcomes. VSMPs are typically tool-agnostic. They connect to existing tools and ingest data from all phases of software product delivery — from customers' needs to value delivery. VSMPs help software engineering leaders identify and quantify opportunities to improve software product performance by optimizing cost, operating models, technology and processes.

Why This Is Important

As organizations scale their agile and DevOps practices, higher-level metrics that assess performance and efficiency of their product delivery are essential. VSMPs integrate with multiple data sources to provide DevOps-related telemetry. These insights enable stakeholders to make data-driven decisions in an agile manner and correct course as needed. The visualization capabilities of VSMPs help product teams analyze customer value metrics against the cost required to deliver that value.

Business Impact

VSMPs help organizations bridge the gap between business and IT by enabling stakeholders to align their priorities to focus on delivering customer value. VSMPs can provide CxOs with strategic views of product delivery health and pipelines, allowing them to make data-driven decisions about future product investments. These platforms also provide product teams with end-to-end visibility and insight into the flow of work to help them address constraints and improve delivery.

Drivers

- Improved software delivery with business priorities and objectives.
- Timely decision making driven by insights from data.
- Optimization of delivery flow through reduction of waste and elimination of bottlenecks.
- Visibility and mapping of end-to-end software delivery processes and identification of cross-team dependencies.
- Quality and velocity improvements of product deployments.
- More stringent governance, security and compliance requirements.

Obstacles

- VSMPs are not focused on continuous integration/continuous delivery (CI/CD) capabilities. Execution of the delivery pipeline requires use of a custom toolchain or DevOps platform.
- VSMPs require customization and data from tools used by multiple stakeholders in the organization, sometimes outside of software delivery. Collaboration with these key stakeholders to deliver the desired insights is paramount.
- VSMPs are still evolving and not all vendors have all the core capabilities.

User Recommendations

- Accelerate business outcomes by leveraging real-time, data-driven metrics and value stream insights provided by VSMPs.
- Leverage VSMPs' AI-powered analytics and insights to surface constraints, detect bottlenecks and improve flow.
- Build customized dashboards and views of product delivery for multiple stakeholders and leadership.
- Utilize VSMPs to assess the performance, quality and value of products, including development costs and ROI.
- Use VSMPs to gain a consolidated view of governance, security and compliance across all product lines.

Sample Vendors

Broadcom; ConnectALL; Digital.ai; HCLSoftware; IBM; OpenText; Opsera; Planview; Plutora; ServiceNow

Gartner Recommended Reading

Market Guide for Value Stream Management Platforms

Tools for Delivering Business Metrics to Software Engineering Teams

Market Guide for Value Stream Delivery Platforms

Use the Right Metrics in the Right Way for Enterprise Agile Delivery

Internal Developer Portal

Analysis By: Manjunath Bhat

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Internal developer portals serve as the interface that enables self-service discovery and access to resources in complex, cloud-native software development environments. They can include software catalogs, scorecards to benchmark software quality, scaffold templates, product documentation, plug-ins for extensibility and automation workflows. Developer portals help improve developer productivity, operational efficiency and enhance governance by providing shared visibility across multiple teams.

Why This Is Important

Internal developer portals help software developers navigate infrastructure complexity, understand service interdependencies and enable faster release cadence in at least three ways. First, they serve as a common viewpoint for multiple teams of developers. Second, they provide developers with self-service access to underlying platform components and environments. Third, they provide a centralized place to score applications and measure progress against reliability and security requirements.

Business Impact

Developer portals can have the following business impacts:

- Developer experience and productivity: Help development teams improve their delivery cadence by improving developer experience, reducing cognitive load and shortening the onboarding time.
- Reliability and resilience: Aim to provide visibility to application health and include scorecards to assess their production readiness.
- Security and governance: Include prebuilt toolkits, templates and curated libraries that help create "paved roads" with built-in compliance, security and audit policies.

Drivers

- Platform engineering: Organizations are adopting platform engineering principles and creating platform teams to scale cross-cutting capabilities across multiple development teams. Platform teams curate internal developer platforms to abstract away the complexity of siloed systems and processes. Internal developer portals serve as an interface through which developers can consume the capabilities of internal developer platforms.
- Backstage: Backstage is one of the first open-source frameworks for building developer portals. It was created at Spotify and is now a Cloud Native Computing Foundation (CNCF) incubating project. The thriving open-source community supporting Backstage has largely contributed to its enormous mind share and rapid adoption. Hundreds of organizations have adopted Backstage since it was open-sourced in 2020. Backstage's success continues to drive interest, momentum and competition in this space.
- Developer experience: With software at the core of all digital innovation today, a great developer experience that accelerates software development becomes a key competitive advantage. Therefore, software engineering leaders are increasingly focused on minimizing developer friction and frustration. The ability to curate and provide customizable, developer-friendly experiences within the developer portal and reign in complexity will drive their appeal for both product and platform teams.

- Innersource: To enable rapid innovation and facilitate greater collaboration and knowledge sharing, software engineering leaders are adopting innersource approaches to software development. However, innersource requires an easy way for other teams to discover and search for existing projects within their organization. This is why organizations adopting innersource are turning to internal developer portals to make projects available and discoverable by other teams. See InnerSource Portal.

Obstacles

- Prerequisites: The successful adoption of internal developer portals goes beyond deploying a tool and requires certain prerequisites to be in place. For example, application services and their dependencies must be organized with consistently defined metadata that helps track their usage, performance and team ownership.
- Absence of platform teams: A dedicated platform team to manage and evolve the portal as a product is necessary to ensure the portal meets desired objectives. The absence of a dedicated platform team, and more so, led by a platform product owner to manage the portal as a product results in a disconnect between developer expectations and the portal's capabilities.
- Lack of developer buy-in: Although the developer portal serves as the "window" to the underlying platform's capabilities, it should provide "paved roads" and not "forced marches" — portal use should remain the choice of the development team. Trying to force development workflows into organizationwide blueprints for building developer portals without involving developers is a recipe for failure.

User Recommendations

- Use internal developer portals to scale cross-cutting software engineering capabilities across multiple development teams and streamline the software delivery life cycle.
- Do not assume that internal developer portals are turnkey solutions — they require a lot of prerequisites to be in place and many cases involve several weeks of prework activities. For example, Backstage requires codification of service-related metadata in YAML files before the content shows up in the software catalog.
- Ensure that the platform team includes internal developer portals in their charter. Continuously innovate portal capabilities by appointing a platform product owner for the developer portal to manage its roadmap, gather feedback and market its capabilities.

Sample Vendors

Atlassian; Calibo; CodeNOW; configure8; Cortex; Mia-Platform; OpsLevel; Port; Roadie

Gartner Recommended Reading

Innovation Insight for Internal Developer Portals

Drive Innovation by Enabling Innersource

Adopt Platform Engineering to Improve the Developer Experience

Cool Vendors in Platform Engineering for Improving Developer Experience

Promise Theory

Analysis By: Roger Williams

Benefit Rating: High

Market Penetration: Less than 1% of target audience

Maturity: Embryonic

Definition:

Promise theory is an approach to modeling the interactions among entities to understand uncertainty, improve coordination and solve problems. Agents (that is, entities in a system that can independently change) make explicit, voluntary promises about unverified behaviors, which enables them to examine a system and assess the likelihood of a desired outcome. Examples include a commitment to complete work by a specified time, or that a system will respond to requests in less than 50 milliseconds.

Why This Is Important

Promise theory provides a common language for coordinating expectations among various entities, due to the bottom-up nature of the approach. It can enable consistent, declarative and repeatable methods for deploying and managing systems.

Business Impact

Promise theory's open structure, simple concepts and acknowledgment of uncertainty can enhance trust in digital business. Given how important building trust is to success with digital business, promise theory can be a catalyst for these efforts. Promise theory use can improve management and coordination among product teams, and enable flexibility in technologies such as digital platform conductor tools. The result is nimble, resilient, decentralized and distributed systems.

Drivers

- The use of promises differs from a command-and-control approach to uncertainty, which requires obligations and the use of punishments and rewards to enforce them. Command and control often fails, because it can't guarantee results (only consequences), and it is prone to passive resistance that results in actions, rather than desired outcomes. For instance, for reporting purposes, we may prohibit an incident record from being saved unless a category is selected. However, the report will be worthless if it leads to the default or first available option being selected for convenience, rather than its value. Similarly, we may insist on 99.999% uptime from a server, yet that, in and of itself, will not make it so.
- Allowing entities to interact through a public promise creates a declarative model (as opposed to a procedural model), which is able to work with a wider variety of systems and system states.
- It also offers the benefit of idempotency — repeatable actions that are safe to execute any number of times without causing a change in the state of the system.
- Organizations can benefit from autonomy and scale — for example, agents can manage the state of the system locally through delegation of control, instead of command and control.

Obstacles

- Lack of awareness of the concept, reducing the odds of widespread adoption unless interest in this concept noticeably increases.
- Rejection due to “not invented here” syndrome or an unwillingness to accept the uncertainty required for digital business and DevOps success.
- An incorrect perception that the topic is merely theoretical and an “ivory tower,” rather than practical.
- Pushback from experts on configuration management in various domains who are unwilling or unable to understand why new terminology is needed (as opposed to other domains simply adopting their approach to configuration management).
- “Promise washing” of work already done — using the language of promises while still intending control and obligation, without recasting in terms of business outcomes.
- Reluctance to set clear targets and provide transparency into performance against those targets.
- Conflict with other behaviors that undermine trust, such as command-and-control management techniques.

User Recommendations

- Begin by identifying the agents that relate to the situation at hand, including the devices and staff that enable the system to deliver the desired result.
- Determine the promises (either explicitly or inherent) that contribute to the result for each agent.
- Identify the important attributes. These include the intention and the agent undertaking it voluntarily, and how the intention is being communicated to another agent. Also, conduct an assessment of the level and intensity of commitment, and identify the benefits that other agents can expect to obtain when the promise is kept.
- Review the network of promises for gaps and conflicts. Agents can then identify new and changed promises that they are willing to make to fill gaps, reduce friction or eliminate conflicts.

Sample Vendors

Apache Software Foundation; Cisco; Cloudsoft; Northern.tech

Gartner Recommended Reading

Emerging Tech Impact Radar: Software Engineering

Market Guide for Digital Platform Conductor Tools

At the Peak

Chaos Engineering

Analysis By: Jim Scheibmeir, Hassan Ennaciri

Benefit Rating: Moderate

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

Chaos engineering is the use of experimental and potentially destructive failure testing or fault injection to uncover vulnerabilities and weaknesses within a distributed system. Chaos engineering tools provide the ability to systematically plan, document, execute and analyze an attack on components and whole systems throughout the life cycle of the system. This planning may include the injection of random timing or attack executions.

Why This Is Important

Many organizations rely on test plans that overemphasize functionality and underemphasize validating the system's reliability and resilience. The distribution and complexity of systems makes understanding them more difficult. Chaos engineering (CE) shifts the focus of testing a system from the "happy path" toward testing how it can degrade gracefully or continue to be useful and secure while under various levels of impact. Applying CE enables improvements to system knowledge and documentation.

Business Impact

CE is aimed to minimize time to recovery and the change failure rate, while maximizing uptime and availability. Addressing these elements helps improve customer experience, satisfaction, retention and acquisition. Gartner inquiries regarding CE increased by over 11% between 2021 and 2022.

Drivers

- Increased complexity of systems and increasing customer expectations are the two largest drivers of CE and the associated tools.
- As systems become more rich in features, they also become more complex in their composition and more critical to digital business success.

- Overall, CE helps organizations become more resilient across their processes, knowledge and technology.
- Teams often lack the confidence to handle failures and the psychological safety to take action to resolve incidents. CE can help build that confidence.

Obstacles

- Within many organizations, the predominant view of CE is that the practice is random, first implemented during production, and increases, rather than reducing, risk.
- Organizational culture and attitudes toward quality and testing can present barriers to the adoption of CE. When quality and testing are only viewed as overheads, there will be a focus on feature development over application reliability.
- It can be challenging just to secure the time and budget to invest in learning CE and associated technologies. Organizations must reach minimum levels of expertise so that value is returned.

User Recommendations

- Utilize a test-environment-first approach by practicing CE in preproduction environments.
- Incorporate CE into your system development, CI/CD or testing processes.
- Build out incident response protocols and procedures, as well as monitoring, alerting and observability capabilities, in tandem with the advancement of the CE practice.
- Utilize scenario-based tests — known as “game days” — to evaluate and learn about how individual IT systems would respond to certain types of outages or events.
- Investigate opportunities to use CE in production to facilitate learning and improvement at scale as the practice matures. However, Gartner believes that very few organizations purposely use CE in their production environments.
- Formalize the practice by adopting a platform or tool to track the activities and create metrics to build feedback for continuous improvements.

Sample Vendors

Amazon Web Services; ChaosIQ; Gremlin; Harness; Microsoft; Steadybit; Verica

Gartner Recommended Reading

Quick Answer: What Metrics Should We Use to Assess and Improve Software Quality?

Predicts 2023: Observing and Optimizing the Adaptive Organization

Top Strategic Technology Trends for 2023: Digital Immune System

Predicts 2023: How Innovation Will Transform the Software Engineering Life Cycle

Infrastructure Platform Engineering

Analysis By: Hassan Ennaciri, Paul Delory

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Adolescent

Definition:

Infrastructure platform engineering is the discipline of building internal software products that present IT infrastructure to users or other platforms in an easily consumable way. Infrastructure platforms are self-service tools that allow nonexpert users to deploy and manage infrastructure themselves while I&O retains governance, security and compliance. Infrastructure platforms are often used as the foundation of higher-order, self-service layers such as internal developer platforms.

Why This Is Important

Digital enterprises are pressured to innovate and deliver products faster to meet customer needs. This requires adopting new operating models and modern practices to deliver scalable, reliable platforms that enable faster product delivery. Infrastructure platform engineering provides automated delivery of curated secure, reliable and scalable infrastructure services that can be available via self services or APIs and reduce the effort and cycle time for users to request and access the products.

Business Impact

Infrastructure platform engineering abstracts the complexity of the digital infrastructure to deliver platforms that continuously evolve to meet customer needs. It is an agile approach necessary to enable software products' value streams to meet customer needs and expectations. It also provides on-demand, fast access to environments, services and tools that improve customer experience and productivity.

Drivers

- **Business agility and innovation:** Digital businesses are required to be responsive to customers' needs and changing market conditions. They must have the ability to quickly deliver products that meet these changing demands and requirements.
- **Cost optimization:** Infrastructure platform engineering teams leverage automation to deliver scalable, reliable and secure platforms. This helps to improve efficiency, reduce resource cost due to manual work and reduce downtime due to change failures. Standardizing tools and platforms also optimizes resource utilizations and reduces cost incurred in tool proliferation.
- **Digital infrastructure and platform complexity:** Public cloud IaaS and PaaS deliver extensive capabilities and are designed to be consumable by developers, but most enterprises need additional governance and management that is best delivered by a platform engineering team.
- **Improve developer experience and productivity:** Infrastructure platform engineering abstracts complexity from developers and provides them with quick access or self-service in the environments they need to develop and test their software. Services can be made via an internal developer portal (IDP) such as Backstage, Calibo or Humanitec.
- **Compliance and security:** Infrastructure platform engineering automates and integrates compliance and security controls into software delivery pipelines, improving the organization's security posture and reducing the burden from developers.

Obstacles

- **Confusion:** There is a lot of hype and confusion about platform engineering and what it means. Many vendors are defining it to help sell their products, causing uncertainty with teams trying to adopt it.
- **Cultural:** This operating model is a new, modern approach that requires a shift in how teams work and collaborate, which is the hardest obstacle to overcome for many organizations.
- **Lack of skills:** Infrastructure platform engineering requires software engineering and specialized skills that may not exist in the organization.
- **Structure of traditional I&O operating models:** The organizational structure of many I&O teams is set up by domain specializations, making it hard to develop and deliver end-to-end services.
- **IT service management approaches:** The current approaches are process-heavy and rely on tickets and handoffs.
- **Complexity:** Successful implementation of infrastructure platform engineering is challenging because it requires new roles and involvement from many stakeholders.

User Recommendations

- **Start small and evolve:** Define initial goals and objectives of the platform by understanding common user needs and delivering viable products that continuously evolve to meet those needs.
- **Build a dedicated team with the right skills:** Successful infrastructure engineering practice requires dedicated teams with diverse skills in infrastructure platforms and software engineering.
- **Identify and fill critical roles such as platform owner and platform architect.** Acquire new talent with the required technical skills, the right mindset and strong interpersonal skills. Develop existing resources by provisioning continuous learning opportunities.
- **Adopt a product mindset:** Thread platform users as customers and ensure that you talk to them and continuously get their feedback to meet their existing needs as well as anticipate their future needs. Enable users and reduce the level of effort required to use the platform products.

Gartner Recommended Reading

Adopt Platform Engineering to Improve the Developer Experience

Top Strategic Technology Trends for 2023: Platform Engineering

Innovation Insight for Internal Developer Portals

Quick Answer: How Can I Optimize the Use of Programmable Platforms for Effective Software Delivery?

Guidance Framework for Implementing Cloud Platform Operations

Continuous Compliance Automation

Analysis By: Daniel Betts, Hassan Ennaciri

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Continuous compliance automation (CCA) integrates compliance and security policy enforcement into DevOps delivery pipelines. CCA codifies and continuously applies compliance policies and controls, while monitoring, reporting on correcting and protecting against vulnerabilities resulting from coding defects and misconfiguration. It reduces the number of manual execution steps involved in adhering to regulatory requirements, enhancing consistency, traceability and auditability.

Why This Is Important

Increased focus on security and compliance improvements drives enterprise investments in compliance automation used to secure code and infrastructure. Traditional compliance practices are incompatible with continuous software delivery processes — leading to slower delivery and unexpected, expensive remediation work. CCA improves release velocity and reliability while simplifying compliance enforcement and reporting via policy-driven, automated controls.

Business Impact

Organizations' evolving DevOps/DevSecOps practices can minimize risks and penalties by embedding automated compliance and reporting into their delivery pipelines. CCA enables organizations to integrate compliance into all phases of the delivery pipeline and consistently enforces compliance policies without sacrificing operational agility. CCA tools can offer benchmarks, assessments and self-service reporting to enable efficiencies in compliance auditing.

Drivers

- As organizations face an increasing number of regulatory obligations and more stringent enforcement, automating compliance will become even more valuable to I&O leaders as they strive to maximize flow.
- Additional compliance requirements continue to be added and require support with limited delay.
- Compliance activities are increasingly executed through automated testing, which delivers increased efficiency for developers and reduces the risk of compliance audit failures.
- As cloud-native application architectures and development models become more pervasive, integrating compliance into the toolchain will become more feasible and common.
- Compliance reporting, benchmarking and assessments are often manual and slow.

Obstacles

- No vendor provides capabilities across all elements of the delivery value stream. DevOps teams must integrate multiple tools into their value streams to provide compliance coverage across development and delivery activities.
- Failure to engage with compliance and security subject matter experts (SMEs) early in the development life cycle can lead to problems.
- A lack of rule set understanding and consistent implementation can be an impediment to CCA. Failure to consistently involve organizational compliance teams in implementation leads to a failure in delivering maximum value.
- Poorly implemented CCA presents a business risk. If it is assumed that by implementing CCA, delivered software becomes compliant without additional effort, organizations will face increased risk of compliance failure.

User Recommendations

- Adhere to compliance, governance and security requirements while creating a leaner operating environment.
- Implement a shift-left approach to ensure compliance controls are understood and applied earlier in the development process. Implement automated compliance checks at every phase of the pipeline, demonstrating a “shift secure” approach.
- Invest in tools that enable CCA at scale and can provide a continuous approach to prevent, detect and correct audit failures, and remove manual reporting activities.
- Select tools that can integrate into DevOps delivery platforms to enable security and compliance checking.
- Enforce security and compliance across all domains, including databases, application code, infrastructure and open-source software. No single vendor tool covers all these domains, so DevOps teams must use multiple tools and integrate across all phases of the delivery pipeline.
- Enable efficient compliance policy checking through compliance automation tools to measure benchmarks, perform assessments and report on compliance policy controls.

Sample Vendors

Anitian; Contrast Security; JFrog; Mend.io; Rapid7; Redgate; RegScale; Snyk; Sonatype; Styra

Gartner Recommended Reading

Market Guide for Continuous Compliance Automation Tools in DevOps

3 Essential Steps to Enable Security in DevOps

How to Build and Evolve Your DevOps Toolchains

Market Guide for Value Stream Delivery Platforms

Platform Engineering

Analysis By: Bill Blosen, Paul Delory

Benefit Rating: Transformational

Market Penetration: 20% to 50% of target audience

Maturity: Adolescent

Definition:

Platform engineering is the discipline of building and operating self-service developer platforms for software development and delivery. A platform is a layer of tools, automations and information maintained as products by a dedicated platform team, designed to support software developers or other engineers by abstracting underlying complexity. The goal of platform engineering is to optimize the developer experience and accelerate delivery of customer value.

Why This Is Important

Digital enterprises need to respond quickly to customer and internal demands; therefore, flexible, complex distributed software architectures have become popular. Software product teams struggle to focus on features due to this complexity, which results in poor developer experience. Platform engineering provides a self-service, curated set of tools, automations and information driven by developer priorities to accelerate value delivery in line with internal stakeholders, such as security and architecture.

Business Impact

Platform engineering empowers application teams to deliver software value faster. It removes the burden of underlying infrastructure construction and maintenance and increases teams' capacity to dedicate time to customer value and learning. It makes compliance and controls more consistent and simplifies the chaotic explosion of tools used to deliver software. Platform engineering also improves the developer experience, thus reducing employee frustration and attrition.

Drivers

- Scale: As more teams embrace modern software development practices and patterns, economies of scale are created, whereby there is enough value to justify creating a platform capability shared by multiple teams.
- Cognitive load: Adoption of modern, distributed architectural patterns and software delivery practices means that the process of getting software into production involves more tools, subsystems and moving parts than ever before. This places a burden on product teams to build a delivery system in addition to the actual software they are trying to produce.
- Need for increased speed and agility: The speed and agility of software delivery is critical to CIOs. As a result, software organizations are pursuing DevOps which is a tighter collaboration of infrastructure and operations (I&O) and development teams to drive shorter development cycles, faster delivery and increased deployment frequency. This will enable organizations to respond immediately to market changes, handle workload failures better and tap into new market opportunities. Platform engineering can drive this type of cross-team collaboration.
- Emerging platform construction tools: Many organizations have built their own platforms, but to date, these platforms have been homegrown, individual efforts tailored to the unique circumstances of the organizations that build them. Platforms generally have not been transferable to other companies or sometimes even to other teams within the same company. However, a new generation of platform-building tools is emerging to change that.
- Infrastructure modernization: During digital modernization, some forward-looking I&O teams embrace a new platform engineering role as a way to deliver more value, increasing their relevance to the business.

Obstacles

- Lack of skills: Platform engineering requires solid skills in software engineering, product management and modern infrastructure, all of which are in short supply.
- Platform engineering is easily misunderstood: Traditional models of mandated platforms with limited regard for developer experience can easily be relabeled and thus not achieve the true benefits of platform engineering.
- Outdated management/governance models: Many organizations still use request-based provisioning models. Those need to give way to a self-service, declarative model, with the primary focus being the effectiveness of the end users developing and operating solutions using the platform.
- Internal politics: There are many intraorganizational fights that could derail platform engineering. Product teams may resist giving up control of their customized toolchains. There might also be no appetite to improve the developer experience. Enterprises may also refuse to fund platform engineering without a clear ROI.

User Recommendations

- Start small with cloud-native workloads: Begin platform-building efforts with thinnest viable platforms for the infrastructure underneath cloud-native applications such as containers and Kubernetes.
- Embed security into platforms: Enable shift-left security within DevOps pipeline platforms, which will provide a compelling paved road to engineers.
- Don't expect to buy a complete platform: Any commercially available tool is unlikely to provide the entirety of the platform you need. Thus, the job of the platform team is to integrate the components necessary for the platform to meet your needs.
- Implement a developer portal as part of your platform: An internal developer portal (IDP) serves as the user interface that enables self-service discovery and access to internal developer platform capabilities. Consider Backstage open-source or other commercial tools. Note: "IDP" has multiple meanings in this context, as well as in the industry.

Gartner Recommended Reading

How to Start and Scale Your Platform Engineering Team

Guidance Framework for Implementing Cloud Platform Operations

Adopt Platform Engineering to Improve the Developer Experience

Innovation Insight for Internal Developer Portals

AI-Augmented Software Engineering

Analysis By: Arun Batchu, Hema Nair, Oleksandr Matvitskyy

Benefit Rating: Transformational

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

The use of artificial intelligence (AI) technologies (e.g., machine learning [ML] and natural language processing [NLP]) to help software engineers create, deliver and maintain applications is designated AI-augmented software engineering (AIASE). This is integrated with engineers' existing tools to provide real-time, intelligent feedback and suggestions.

Why This Is Important

Today's software development life cycle includes such routine and repetitive tasks as boilerplate functional and unit-test code and docstrings, which AIASE tools automate. AI-powered automation enables software engineers to focus their time, energy and creativity on such high-value activities as feature development. Emerging AI tools discover the configurations that meet operational goals. Software builders who use these tools remain productive and engaged, and they stay longer in their jobs.

Business Impact

AIASE accelerates application delivery and allocates software engineering capacity to business initiatives with high priority, complexity and uncertainty, helping quality teams develop self-healing tests and nonobvious code paths. These tools automatically generate test scenarios previously created manually by testers, and detect test scenarios often missed by test teams. AIASE tools detect issues with code security, consistency or maintainability and offer fixes.

Drivers

Demand drivers include:

- The increasing complexity of software systems to be engineered
- Increasing demand for developers to deliver high-quality code faster
- Increasing numbers of application development security attacks
- Optimizing operational costs

Technology solution drivers include:

- The application of AI models to prevent application vulnerabilities by detecting static code and runtime attack patterns
- The increasing impact of software development on business models
- The application of large language models to software code
- The application of deep-learning models to software operations

Obstacles

- Hype about the innovation has caused misunderstandings and unrealistic expectations about the benefits of AIASE.
- There is a lack of deep comprehension of generated artifacts.
- There is limited awareness about production-ready tools.
- Software engineers who fear job obsolescence have shown resistance.
- There is a lack of transparency and provenance of data used for model training.
- Uneven, fragmented solutions that automate only some of the tasks in the software development life cycle (SDLC).
- AI skills such as prompt engineering, training, tuning, maintaining and troubleshooting models.
- High model training and inference costs at scale.
- Intellectual property risks stemming from models trained on nonpermissive licensed code.
- Privacy concerns stemming from code, and associated proprietary data leaking as training data for AI models.
- Technical employees' fear of jobs being automated by AI.

User Recommendations

- Pilot, measure and roll out tools only if there are clear gains.
- Verify the maintainability of AI-generated artifacts, including executable requirements, code, tests and scripts.
- Track this rapidly evolving and highly impactful market to identify new products that minimize development toil and improve the experience of software engineers, such as those that ease security and site operations burden.
- Reassure software engineers that AIASE is an augmentation toolset for human engineers, not a replacement.
- Pick providers (including open-source vendors) that supply visibility to training data and transparency on how the model was trained.
- Establish the correct set of metrics, such as new release frequency and ROI, to measure the success of AIASE.

Sample Vendors

Akamas; Amazon Web Services; Diffblue; Google; IBM; Microsoft; OpenAI; SeaLights; Sedai; Snyk

Gartner Recommended Reading

Innovation Insight for ML-Powered Coding Assistants

Infographic: Artificial Intelligence Use-Case Prism for Software Development and Testing

Market Guide for AI-Augmented Software Testing Tools

DevOps Platforms

Analysis By: Manjunath Bhat

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Early mainstream

Definition:

DevOps platforms provide fully integrated capabilities to enable continuous delivery of software using agile and DevOps practices. These span the software development life cycle (SDLC) and include product planning, version control, continuous integration, test automation, continuous deployment, release orchestration, automating security and compliance policies, monitoring, and observability. DevOps platforms support team collaboration, secure software development and software delivery metrics.

Why This Is Important

Organizations use DevOps platforms to minimize tool friction and operational complexity resulting from disparate toolchains, manual handoffs, and lack of consistent visibility throughout the SDLC. This enables product teams to deliver faster customer value without compromising quality. The DevOps platforms market reflects the consolidation of technologies across development, security, infrastructure and operations to streamline software delivery.

Business Impact

DevOps platforms are the software delivery pipelines that enable continuous delivery of business value. The seamless integration, automation, extensibility and shared visibility between development, security and operations workflows help bridge the silos that exist between these teams. Using a common platform for development, security and operations accelerates agile transformation, and helps organizations move toward a product and platform team operating model.

Drivers

- **Modernizing application architectures:** Modernizing applications to take advantage of emerging cloud-native architectures requires fundamental changes to underlying DevOps practices and tools.

- **Increased emphasis/focus on enhancing developer experience:** Improved developer experience, agility and the need to improve delivery cadence by reducing cognitive load due to constant context switches and repetitive low-value work.

- **An integrated approach to security and compliance:** Integrating and automating security, compliance and governance as part of the development and delivery process is becoming a priority. A few DevOps platform providers include SCA capabilities as features in their offerings. Example vendors include GitHub, GitLab and JFrog.

- **Improved visibility into the flow of work:** Organizations are under pressure to reduce friction and manual handoffs, and this requires complete visibility into software delivery pipelines from ideation to production.

Obstacles

- Organizations that want to unlock the full benefits of DevOps platforms must be willing to replace an existing toolchain — either completely or in part. Teams can view the change as a disruption to their established ways of working and resist any change to the tools they have been using.

- Organizations accrue technical and skill debt over time due to outmoded automation workflows and legacy applications. This hinders teams from adopting new tools.

- Dependency on a single provider for a majority of their software development needs increases concentration risk and lowers bargaining leverage.

- Most DevOps platforms currently fall short in providing the full set of software delivery capabilities that organizations require to build, deliver, measure and improve the flow of value in the software delivery life cycle.

User Recommendations

- To fully reap the business benefits of DevOps platforms, organizations must adopt agile methods and practices.
- Scale and deliver capability by providing DevOps platforms as self-service platforms to reduce overhead, lower complexity, and ensure consistent and templatized workflows across multiple teams.
- Improve the flow of value by streamlining the software delivery life cycle with DevOps platforms that provide enhanced visibility, traceability, auditability and observability across the DevOps pipeline.
- Support InnerSource efforts by building InnerSource portals using source control repositories available in DevOps platforms.
- Reduce inconsistency in CI/CD pipeline definitions between teams by leveraging declarative and shareable pipeline capabilities in DevOps platforms.

Sample Vendors

Atlassian; CircleCI; CloudBees; GitHub; GitLab; Harness; JetBrains; JFrog; Red Hat; VMware

Gartner Recommended Reading

Keys to DevOps Success

Research Roundup for DevOps, 2022

Software Supply Chain Security

Analysis By: Dale Gardner

Benefit Rating: Transformational

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Software supply chain security (SSCS) is the set of processes and tools used to curate, create and consume software in ways that mitigate attacks against software or its use as an attack vector. Curation focuses on assessing risks of third-party software and assessing its acceptability. Creation focuses on secure development and the protection of software artifacts and the development pipeline. Consumption validates integrity of software through verification, provenance and traceability.

Why This Is Important

Given triple-digit increases in software supply chain attacks, global regulatory mandates, the widespread use of open-source tools and exposures due to remote ways of working, securing the overall software supply chain is paramount for all parts of an organization. This is particularly critical given growing digitized processes and services, making software the central location where most intellectual property now originates and resides.

Business Impact

- SSCS enables management and mitigation of a variety of risks, such as compliance failures and a host of security incidents.
- SSCS is necessary to satisfy both regulatory requirements and buyer demands for increased transparency around a vendor’s application security measures.
- SSCS can help avoid increased friction and lost productivity from ad hoc efforts to secure the development environment and application artifacts.

Drivers

- Software supply chain attacks have become increasingly sophisticated, with malicious actors exploiting weaknesses at every stage in the software procurement, development and delivery life cycle. The number of attacks has increased dramatically, with the Sonatype report citing an average 742% year-over-year growth rate between 2019 and 2022. These attacks pose risks threatening the efficiency and productivity of business and organizations, and the basic ability of those organizations to operate. They may cause data breaches and other security incidents, and loss of intellectual property, and even pose threats to human safety.

- In response, governments and regulatory agencies throughout the world are turning their attention to ways in which organizations can be encouraged — or forced — to improve software supply chain security efforts. In the United States, a substantial focus is on the "power of the purse," with multiple initiatives across all sectors of the federal government focused on preventing vendors with poor security practices from reaching lucrative markets. These activities are not limited to the United States. Governments throughout North America, the European Union and the United Kingdom, and through Japan and Southeast Asia have taken actions ranging from the passage of legislation to efforts to improve awareness and understanding of the issues.

- Organizations have only just begun to tackle the issues associated with software supply chain risks, placing many in a "catch-up" position where action must be taken. They have been slow to take steps to secure their software supply chains. According to Sonatype's 8th Annual State of the Software Supply Chain Report, only about 7% of respondents have taken broad action. This is consistent with inquiry trends noted at Gartner. While there have been significant increases in inquiry around SSCS, many questions remain focused on specific subsets of the broader security challenge.

Obstacles

- The majority of organizations lack a complete understanding of SSCS, and thus have yet to adopt a comprehensive view of the software supply chain and all its risks. Such a view would encompass security through the curation and selection of secure software (during development and procurement), during creation, and in consumption as organizations track the usage of code and evolving threats. For example, many organizations are focused on acquiring software bills of material (SBOMs), but have yet to establish how those artifacts will be evaluated, stored, and used.
- Efforts to articulate secure means of ensuring the integrity and provenance of software artifacts throughout the software creation, curation, and consumption processes are emerging, but uneven in scope, execution and adoption. For example, it's common for organizations to evaluate open-source software used in development for vulnerabilities, but most organizations are not proactive when identifying potential risks.

User Recommendations

- Adopt a comprehensive view of software supply chain security and do not fall into the trap of focusing solely on specific aspects of the problem.
- When evaluating third-party software for use, proactively assess it for security, legal and intellectual property risks.
- Evaluate and authorize the use of specific components, and establish a trusted repository of vetted code. Track the usage and deployment of third-party code to ease remediation efforts when vulnerabilities are discovered.
- Protect the entire development tool chain and associated artifacts from attack. Include the protection of code and other artifacts (e.g., IaC or configuration scripts) from unauthorized access or alteration, defending the delivery pipeline from attack, and hardening the operating environment.

Sample Vendors

Arnica; BluBracket; Chainguard; Cybeats; Cycode; GitGuardian; Legit Security; Ox Security; Palo Alto Networks (Cider Security)

Gartner Recommended Reading

Emerging Best Practices to Manage Digital Supply Chain Risks

How Software Engineering Leaders Can Mitigate Software Supply Chain Security Risks

Innovation Insight for SBOMs

Performance Engineering

Analysis By: Joachim Herschmann

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Performance engineering is a systematic approach to developing software applications to ensure they meet the application performance objectives of the business. It focuses on the architecture, design and implementation choices that will affect application performance, and encompasses practices that help mitigate associated risks before progressing to subsequent phases.

Why This Is Important

The ability to consistently deliver products that satisfy end-user expectations of scalability, stability, quality of service (QoS), availability and performance has become crucial for digital businesses. Performance engineering promotes a holistic and proactive approach, with performance requirements driving the design, development and delivery of products as part of a continuous quality strategy.

Business Impact

Performance engineering includes both “shift left” and “shift right” testing practices and significantly improves an organization’s ability to consistently deliver solutions that exceed customers’ performance expectations. Organizations can improve application observability by using the insights gained through performance engineering. This includes adding metrics and telemetry, or deploying new application monitoring tools so alerts can be raised before a performance bottleneck impacts users.

Drivers

- Increased end-user expectations for application quality, specifically operational characteristics such as performance efficiency.
- The need to ensure business continuity under changing usage patterns, network topologies and data volumes.
- The need to optimize the use of modern microservices-based architectures, as well as the automation and integration capabilities of modern application platforms.
- Support for different migration scenarios, such as lift and shift, replatforming or refactoring the architecture of packaged or on-premises apps.
- The need to manage performance and scalability across different cloud providers, such as Amazon Web Services (AWS), Google or Microsoft Azure, to ensure the ability to shift from one operator to another without a change in user experience.
- Cost optimization for SaaS/PaaS services that makes use of dynamic infrastructure to spin up (and down) testing resources as needed.

Obstacles

- Many organizations focus only on tools and technology. Performance engineering requires a change in organizational culture. Tools enable quality but won't solve problems on their own.
- Departmental silos, traditional top-down management structures and a lack of experience with managing quality continuously can impede adoption.
- Successful performance engineering requires clear goals that are aligned with the priorities of the business. The absence of established service-level indicators (SLIs) and objectives (SLOs) leads to a lack of realistic, quantifiable service availability or performance metrics.
- Performance engineering requires engaging stakeholders throughout the organization and empowering them to be more accountable and to seek out opportunities for improvement. Such a holistic approach can be seen as restrictive and requires consensus across all team members.
- Performance engineering includes designing with performance in mind, building the product with clear performance objectives and facilitating the discovery of issues early in development.

User Recommendations

- Create awareness of nonfunctional or operational characteristics, such as performance efficiency or the Application Performance Index (Apdex), an open standard developed by an alliance of companies to measure the performance of software applications in computing. ISO/IEC 25010 provides a template for understanding quality characteristics and includes performance efficiency as one of the top-level nonfunctional domains.
- Establish SLIs and SLOs as part of a performance engineering strategy. An SLI is a realistic, quantifiable measure of service availability or performance. An SLO is the target for the SLI over a fixed period.
- Foster a proactive performance quality strategy that makes performance an explicit requirement and verifies that performance goals are met and user satisfaction meets expectations.
- Allocate ownership and appoint staff with the skills needed for performance engineering by identifying the required roles, technologies and practices.
- Establish performance quality metrics based on the joint objectives that the business and IT are trying to accomplish.
- Collaborate with I&O leaders and establish a feedback loop using performance information from production and real users.

Sample Vendors

AppDynamics; Dynatrace; Keysight (Eggplant); Micro Focus; Perforce Software; Tricentis

Gartner Recommended Reading

How to Identify and Resolve Application Performance Problems

Quick Answer: Which Non-Functional Software Quality Characteristics Can Make or Break Your Product?

Improve Software Quality by Building Digital Immunity

AI-Augmented Testing

Analysis By: Joachim Herschmann, Jim Scheibmeir

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

AI-augmented testing comprises AI- and machine learning (ML)-based technologies and practices to make software testing activities more independent from human intervention. It continuously improves testing outcomes by learning from the data collected from performed activities. It extends traditional test automation beyond the automated execution of test cases to include fully automated planning, creation, maintenance and analysis of tests.

Why This Is Important

Software engineering leaders seeking to release faster without degrading quality are looking for more efficient ways of testing across all phases of the software life cycle. AI-augmented testing enables the automation of a broad set of testing activities related to requirement quality, design quality, code quality, release quality and operational resilience. This increases the degree of autonomy of those activities.

Business Impact

The adoption of AI-augmented testing has the potential to significantly improve an IT organization's ability to serve and delight its customers. It can enable the adjustment of testing scenarios and overall software quality parameters as part of a continuous quality strategy aimed at optimizing the end-user experience. It will also help to constitute a closed-loop system that provides continuous feedback about critical quality indicators and helps to reduce the costs of creating and maintaining tests.

Drivers

- A high dependency on human expertise and interaction limits how quickly modern digital businesses can design, build and test new software.
- Where automated testing is already in place, current levels of automation often remain below expectations due to a continued dependency on human intervention to maintain the automation as applications under test (AUT) evolve.
- The pressure to innovate quickly for market differentiation without compromising on quality relies on both a higher velocity and a higher degree of autonomy of the related activities.
- While delivery cycle time is decreasing, the technical complexity required to deliver a positive user experience and maintain a competitive edge is increasing. The answer is not more testing, but more intelligent testing enabled by AI technologies.

Obstacles

- Currently available tools are still relatively new, have a narrow scope and still need to prove their value. Generative AI, in particular, is the latest and most disruptive example. Hallucinations (content that is nonsensical or untruthful in relation to certain sources), subpar training data, potential copyright violations and security issues are the main risks associated with that particular set of AI technologies.
- Waiting until better AI-augmented testing solutions are available leads to a loss in competitive advantage and fewer innovations. It also incurs greater testing costs and the risk of undertesting.
- Underestimating the time required to acquire new skills and setting wrong expectations about the time required to become successful can be obstacles.
- Gathering, cleaning and processing data and training the model are not trivial tasks, and require adequate skills. Moreover, they are not yet autonomous processes.

User Recommendations

- Set the right expectations about the potential and limitations of AI-augmented testing and ensure that humans are always in the loop to verify the results produced by AI-augmented testing tools. This is particularly relevant for tools employing generative AI to automatically create tests, as generated tests may be completely useless or create false positives or negatives.
- Build a pilot team to map the cases where AI can provide the biggest benefits for software testing to the areas of greatest need in your organization. Evaluate opportunities to use it for test planning and prioritization, test creation and maintenance, test data generation, visual testing and test and defect analysis.
- Maximize the impact of AI-augmented testing by using it to enable a systematic approach to achieve the quality goals of business and development. Focus on key business value enablement and determine where it can help reduce cost and manage risk.

Sample Vendors

ACCELQ; Applitools; Avo Automation; CodiumAI; Diffblue; Functionize; mabl; ProdPerfect; testRigor

Gartner Recommended Reading

Market Guide for AI-Augmented Software-Testing Tools

Quick Answer: How Can AI Provide Benefits for Software Testing?

Innovation Insight for Continuous Quality

Improve Software Quality by Building Digital Immunity

Infographic: Artificial Intelligence Use-Case Prism for Software Development and Testing

Continuous Quality

Analysis By: Joachim Herschmann, Jim Scheibmeir

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Continuous quality is a systematic approach toward process improvement to achieve the quality goals of business and development. A continuous quality strategy fosters a companywide cultural change to drive quality ownership and engage everyone in achieving the set quality goals. It shifts focus from testing (an activity) to quality and digital immunity (outcomes) and encompasses the practices that help mitigate risks before progressing to subsequent stages of the software development life cycle.

Why This Is Important

The ability to consistently deliver business value with high quality has become critical for organizations seeking to mature their software development processes. Continuous quality encourages a holistic and proactive approach with functional and nonfunctional requirements driving the design, development and delivery of products. Development teams must use a continuous quality strategy to create quick feedback loops that enable easy adaptation to changes and increase customer satisfaction.

Business Impact

Continuous quality provides a framework for operational excellence that shifts the focus from testing as an activity to a holistic proactive approach to quality. Designing with quality in mind, building in quality and detecting defects earlier allow organizations to release higher-quality products more often than traditional quality control approaches enable. The adoption of a continuous quality strategy can significantly improve an organization’s ability to serve and delight its customers.

Drivers

- Raised end-user expectations for application quality require a shift to a more holistic view of what constitutes superior quality that delights users.
- Organizations are under the pressure to innovate rapidly in order to launch differentiated products in the market quickly without compromising quality.
- Highly dynamic distributed cloud-based architectures require the ability to continuously monitor quality and deal with defects in real time.
- As software development scales, teams need consistent delivery practices and a clear understanding of the quality characteristics that make or break a product, especially about the nonfunctional (operational) aspects of the product.

Obstacles

- **Lack of clear goals:** Successful continuous quality requires clear goals aligned with the priorities of the business.
- **Internal pushback:** Continuous quality requires engaging stakeholders across the organization and empowering them to be more accountable. Such a holistic approach can be seen as too far-reaching and requires consensus on usage among all team members.
- **Loss of productivity:** Changing organizational culture and engaging in new practices require a significant amount of investment and time. This will impact current timelines and can cause a decrease in productivity prior to reaching steady productivity.
- **Limited to testing only:** Continuous quality includes designing a product with quality in mind, building it with clear quality objectives, and facilitating the discovery of issues early in the development process.

User Recommendations

- Move away from the traditional application- or project-centric- focused model to a holistic quality approach by adopting an ecosystem-centric view of quality and placing focus on business outcomes.
- Put a fundamental managed software testing approach in place with test policy and strategy being a focus area. Establish Test Maturity Model Integration (TMMi) Level 2 artifacts before assessing tools to understand opportunities for testing early and continuously.
- Accelerate product delivery by championing a continuous quality mindset and involving stakeholders across the organization. Allocate ownership and appoint staff with skills needed for continuous quality by identifying the required roles, technologies and practices.
- Enable collaboration with user experience (UX) designers and customer experience (CX) teams to infuse quality right from the inception of an idea. Establish relevant quality metrics based on the joint objectives that the business and IT are trying to accomplish.

Gartner Recommended Reading

Innovation Insight for Continuous Quality

Tool: Craft and Communicate a Continuous Quality Strategy

Infographic: Sketching a Continuous Quality Strategy Over Coffee

Improve Software Quality by Building Digital Immunity

Infographic: Essential Skills for QA Engineers

Observability

Analysis By: Padraig Byrne, Gregg Siegfried

Benefit Rating: Transformational

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Observability is the characteristic of software and systems that enables them to be understood, based on their outputs and enables questions about their behavior to be answered. Tools that facilitate software observability enable observers to collect and quickly explore high-cardinality telemetry using techniques that iteratively narrow the possible explanations for errant behavior.

Why This Is Important

The inherent complexity of modern applications and distributed systems and the rise of practices, such as DevOps, has left organizations frustrated with legacy monitoring tools and techniques. These can do no more than collect and display external signals, which results in monitoring that is, in effect, only reactive. Observability acts like the central nervous system of a digital enterprise. Observability tools enable a skilled observer to explain unexpected system behavior more effectively.

Business Impact

Observability tools have the potential to reduce both the number of service outages and their severity. Their use by organizations can improve the quality of software, because previously invisible (unknown) defects and anomalies can be identified and corrected. By enabling product owners to better understand how their products are used, observability supports the development of more accurate and usable software, and a reduction in the number and severity of events affecting service.

Drivers

- The term "observability" is now ubiquitous, with uses extending beyond the domain of IT operations. Although the 2020s are now the "decade of observability," care must be taken to ensure the term retains relevance when used beyond its original range of reference.
- OpenTelemetry's progress and continued acceptance as the "observability framework for cloud-native software" raises observability and its toolchain.
- Traditional monitoring systems capture and examine signals (possibly adaptive) in relative isolation, with alerts tied to threshold or rate-of-change violations that require prior awareness of possible issues and corresponding instrumentation. Given the complexity of modern applications, it is unfeasible to rely on traditional monitoring alone.
- Observability tools enable a skilled observer, a software developer or a site reliability engineer to explain unexpected system behavior more effectively, provided enough instrumentation is available. Integration of software observability with artificial intelligence for IT operations (AIOps) to automate subsequent determinations is a potential future development.
- Observability is an evolution of longstanding technologies and methods, and established monitoring vendors are starting to reflect observability ideas in their products. New companies are also creating offerings based on observability.

Obstacles

- In many large enterprises, the role of IT operations has been to "keep the lights on," despite constant change. This, combined with the longevity of existing monitoring tools, means that adoption of new technology is often slow.
- Enterprises have invested significant resources in their existing monitoring tools, which exhibit a high degree of "stickiness." This creates nontechnical, cultural barriers to adopting new practices such as those based on observability.
- Costs associated with observability tools have grown as companies struggle to keep up with the explosion in volume and velocity of telemetry.

User Recommendations

- Assess software observability tools to integrate into their continuous integration/continuous delivery (CI/CD) pipelines and feedback loops.
- Investigate problems that cannot be framed by traditional monitoring by using observability to add flexibility to incident investigations.
- Enable observability by selecting vendors that use open standards for collection, such as OpenTelemetry.
- Tie service-level objectives to desired business outcomes using specific metrics, and use observability tools to understand variations.
- Ensure IT operations and site reliability engineering teams are aware of updates to existing monitoring tools and how they may take advantage of them. Many traditional application performance monitoring vendors are starting to incorporate observability features into their products.
- Avoid the conclusion that observability is synonymous with monitoring. At minimum, observability represents the internal perspective, rather than external.

Sample Vendors

Chronosphere; Grafana; Honeycomb; Lightstep; Observe; VMware

Gartner Recommended Reading

Monitoring and Observability for Modern Infrastructure and Applications

Magic Quadrant for Application Performance Monitoring and Observability

DevOps Test Data Management

Analysis By: Andrew Bales

Benefit Rating: Moderate

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

DevOps test data management is the process of providing DevOps teams with sanitized data to evaluate the performance, functionality and security of applications. It typically includes copying production data, anonymization or masking, and sometimes, virtualization. In some cases, specialized techniques, such as synthetic data generation, are appropriate. Given potential compliance and privacy issues, the efforts frequently involve members of application and data security teams.

Why This Is Important

Test data management is inconsistently adopted across organizations, with many teams still copying production data for use in test environments. As organizations shift to DevOps and the pace of development increases, this traditional approach is increasingly at odds with requirements for efficiency, privacy and security, and even the increased complexity of modern applications. This opens organizations to a variety of legal, security and operational risks.

Business Impact

Quick provisioning of test data helps ensure the pace of development isn't slowed. It's also increasingly important to remain compliant with the growing number of privacy mandates to which organizations are subject. This helps avoid fines and remediation and mitigation costs, along with the inevitable delays associated with audits and investigations. Finally, by providing application teams with anonymized or synthetic data, the risk of data breaches is reduced.

Drivers

- Test data management is generally viewed as a mature, relatively uncomplicated, practice. However, the reality is the combination of the increased pace of development from DevOps and a growing number of privacy mandates and constraints have stressed traditional approaches — prompting the use of virtualization as well as alternative masking and protection techniques, including synthetic data generation.

- More traditional test data management has been inconsistently adopted, with many organizations either simply using copies of production data in unsafe environments or generating "dummy data" (distinct from emerging synthetic data generation techniques) that doesn't accurately reflect production data. The data privacy requirements and complexity issues noted have prompted organizations to revisit and update their processes with an eye toward scalability and automation. Updated technologies may also be a requirement. For example, requirements for speed and agility have created a need for data virtualization tools.

- Data protection is cited by most Gartner clients in inquiries regarding test data management. Privacy and data protection requirements mean it's no longer safe to simply provide development teams with a copy of production data since this practice leaves organizations open to increased risk of regulatory violations, data breaches and other security issues.

- With modern applications relying on an increasing number of interconnected data stores (many of which, technologically speaking, are vastly more complex) and applications and APIs to function, testing has become more complex. Such complexity demands that tools support the ability to coordinate and synchronize changes across different data stores to ensure relational consistency while still addressing security and speed mandates.

Obstacles

- In the absence of a strong culture of security, processes and technologies to protect sensitive information during development and testing will encounter friction. Conflicting needs for rapid development and privacy require attention to a mix of organizational and cultural issues to strike a balance across groups.
- Responsibility for test data management in organizations has been shared by application development and database administration. New technologies and processes may shift those responsibilities to include security, complicating organizational dynamics and potentially creating the need for additional resource allocation.
- Implementation can be a burden, especially where little or no data sensitivity classification has been done. This must be accomplished before teams can proceed with the required data transformation and masking. These efforts are typically combined with an analysis of data relationships so that relational integrity can be assured.

User Recommendations

- Involve stakeholders — such as data management, privacy and security teams, compliance teams, etc. — to understand consumption patterns and needs — as appropriate.
- Document existing test data management practices so tools and processes can be evaluated against data protection mandates.
- Coordinate with other teams to avoid duplication of effort and tooling since data masking tools may also be used by analytics teams (e.g., to provide data for machine learning or other purposes).
- Evaluate data masking tooling by considering support for databases and other stores, data discovery capabilities, types of masking supported, and the ability to coordinate change to ensure consistency (e.g., key fields) across multiple sources.
- Evaluate data virtualization for DevOps use cases where frequent updates to test data are required. Virtualization can speed up the process of providing copies of safe data.
- Determine whether synthetic data generation is appropriate in cases where suitable data doesn't exist or reidentification risks are high.

Sample Vendors

BMC Software; Broadcom; DATPROF Delphix; Hazy; Informatica; K2View; Mage; OpenText; Solix Technologies

Gartner Recommended Reading

Market Guide for Data Masking

Elevating Test Data Management for DevOps

3 Steps to Improve Test Data Management for Software Engineering

Innovation Insight for Synthetic Data

Sliding into the Trough

Communities of Practice

Analysis By: Thomas Murphy, Peter Hyde

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

A community of practice (CoP) is a stable, informal and voluntary structure designed to foster problem solving, knowledge sharing and organizational learning. These communities promote extensive collaboration and social learning which are crucial to enable the cultural transformation necessary to drive organizational change.

Why This Is Important

CoPs aid in the adoption of new practices and combat silos. The shift to hybrid work has introduced new challenges (and opportunities), and the constant press for productivity often means participation isn't an allocated activity. In addition, IT organizations often continue to rely on top-down, functionally aligned centers of excellence (COEs) that focus on policy adherence and governance instead of problem solving and learning. In contrast, CoPs are motivational if they provide a clear vision and space for the community to collaborate, learn and evolve.

Business Impact

Communities of practice enable knowledge management, professional development and capability improvement.

Our research finds that CoPs:

- Shorten the learning curve for employees and improve onboarding.
- Provide higher levels of employee satisfaction, leading to better retention, motivation and innovation.
- Respond more rapidly to customer needs and inquiries.
- Reduce duplication of effort.

- Spawn new ideas for products and services.
- Help members develop capabilities that align with organizational needs.

Drivers

- The members of a software development team must gain new skills and behaviors, and improve the pace of delivery. CoPs provide a community-focused approach for knowledge sharing and organizational learning.
- A CoP is a continued source of strength through interactions and learning that should be captured and shared digitally.
- CoPs are voluntary groups that enable practitioners to create common guidelines and practices based on proven practice.
- Transversal knowledge management through CoPs prevents unintentional functional and product silos from forming.

Obstacles

- Building a community in hybrid or remote situations can be a challenge and requires soft skills development.
- Hierarchical structures and existing reporting chains can feel threatened by the self-directed nature of CoPs, resulting in management wanting to utilize COEs and business as usual.
- While CoPs are a powerful concept, organizations will also need other structures and practices, such as product orientation, agile methodologies, DevOps Dojos or InnerSource, to drive the best long-term benefits.
- CoPs require management support and active community coaching to be effective, continuously create value and prevent rapid obsolescence.

User Recommendations

Software engineering leaders leaders should:

- Empower cross-functional communities to learn and discover through experimentation, collaboration and knowledge sharing by creating and communicating a clear vision for the communities with time and space.

- Encourage experimentation and risk taking by dedicating time to CoPs to build a culture where failure and new approaches are celebrated as learning opportunities.
- Complement CoPs by organizing job rotations, hackathons and technology conferences.

Sample Vendors

Atlassian; Microsoft; Salesforce (Slack Technologies); Stack Overflow; Zoom Video Communications

Gartner Recommended Reading

Community of Practice Essentials

Case Study: Kick-Starting a Low-Code/No-Code Community of Practice (Heathrow Airport)

Ignition Guide to Creating Communities of Practice in Software Engineering

Team Collaboration Attributes That Drive High Performance

How Should We Collaborate in a Hybrid World?

Site Reliability Engineering

Analysis By: George Spafford, Daniel Betts

Benefit Rating: Transformational

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Site reliability engineering (SRE) is a collection of systems and software engineering principles used to design and operate scalable resilient systems. Site reliability engineers work with the customer or product owner to understand operational requirements and define service-level objectives (SLOs). Site reliability engineers work with product or platform teams to design and continuously improve systems that meet defined SLOs.

Why This Is Important

SRE emphasizes the engineering disciplines that lead to resilience; but individual organizations implement SRE in widely varying ways such as a defined role or a set of practices. SRE teams can serve as an operations function, and nearly all such teams have a strong emphasis on blameless root cause analysis. This is to decrease the probability and/or impact of future events and to enable organizational learning, continual improvement and reductions in unplanned work.

Business Impact

The SRE approach to improving reliability and resilience is intended for products and platforms that need to deliver customer value at speed at scale while managing risk. The two primary use cases are to improve the reliability of existing products/platforms or to create new products or platforms that need reliability from the start.

Drivers

- Clients are under pressure to meet customer requirements for reliability while scaling their digital services and are looking for guidance to help them.
- While Google originated what became known as SRE and continued to evolve it, practitioners are developing and sharing new practices as well. Potential practitioners looking for pragmatic guidance to improve the reliability of their systems have a rich body of knowledge they can leverage that works well with agile and DevOps.
- Organizations are adopting highly skilled automation practices (usually DevOps), and usage of infrastructure-as-code capabilities (which usually requires a cloud platform) to deliver digital business products reliably.
- The most common use case based on inquiry calls with clients is to leverage SRE concepts to improve the reliability of existing systems that are not meeting customer requirements for availability, performance or are proving difficult to scale.

Obstacles

- Insufficient internal marketing to understand what agile, DevOps or product teams need or would value and then explaining how the value SRE can deliver will justify the costs and risks incurred. Without marketing its benefits, SRE adoption tends to be less certain or slower. The SRE concept by itself is insufficient — people must continuously believe it is worthwhile.
- Finding SRE candidates who have the right mix of development, operations and people skills is a big challenge for clients. Impacts on initial adoption and scaling efforts as well.
- Rebranding of a traditional operations team without changing to adopt SRE practices, only SRE in name.
- Clients have voiced problems with product owners who overly focus on functional requirements and not nonfunctional requirements thus slowing improvements and support of SRE within the organization.

User Recommendations

- Leverage practices pragmatically based on need. Don't feel that you must implement SRE exactly the way Google does it, learn what works for you.
- Detect an opportunity to begin that is politically friendly, will demonstrate sufficient value and has an acceptable risk profile.
- Start small, focus, learn, improve, and demonstrate value — do not try to change everything at once.
- Work with the customer or product owner to define clear, obtainable SLOs based on their needs.
- Implement monitoring and improve observability to objectively report on actual performance relative to the SLOs.
- Product owners must be accountable for functional and non-functional requirements of their products.
- Instill collaborative working between site reliability engineers, developers and other stakeholders to help them learn how to design, build and evolve their products to meet SLOs.
- Create a community, implement effective organizational learning practices and evolve SRE practices.

Sample Vendors

Atlassian; Blameless; Datadog; Dynatrace; New Relic; OpsRamp; PagerDuty; Splunk

Programmable Infrastructure

Analysis By: Philip Dawson, Nathan Hill

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Programmable infrastructure is the concept of using and applying methods and tooling from the software development area to management of IT infrastructure. This includes, but is not limited to, APIs, immutability, resilient architectures and agile techniques.

Why This Is Important

Programmable infrastructure ensures optimal resource utilization, while driving cost efficiencies. A continuous delivery approach requires continuous insight and the ability to automate application responses. Moving to an API-driven infrastructure is the key first necessary step to enabling anti-fragile and sustainable automation through programmatic techniques.

Business Impact

Greater value (rather than cost reduction) is achieved via programmable infrastructure's ability to drive adaptive automation — responding faster to new business infrastructure demands, driving service quality and freeing staff from manual operations. Programmable infrastructure reduces technical debt with investment and enables a sustainable and highly responsive IT infrastructure service to the business.

Drivers

- Programmable infrastructure strategies are applied to private cloud, hybrid cloud and infrastructure platforms as well as public cloud. Demand for programmable infrastructure grows as heterogeneous infrastructure strategies are embraced.

- Programmable infrastructure is needed to manage the life cycle of infrastructure delivery from provisioning, resizing and reallocation to reclamation, and in the case of external resources, manage elasticity and the termination of consumption.

- Programmable infrastructure is needed to optimize and reduce the dependency on the infrastructure life cycle. More importantly, it enables the desired (performance, cost, speed) infrastructure provisioning and orchestration in line with business demands.

Obstacles

- The ongoing cost of refreshing API-enabled infrastructure components on-premises after initial implementation adds financial pressure to organizations.
- Applying automation to existing monolithic infrastructure components fails due to the lack of platform agility and vendor lock-in.
- While APIs enable integration across different infrastructure platforms, the lack of open APIs/API compatibility across vendor platforms creates a siloed mentality.
- The implementation of programmable infrastructure is hampered by the early adoption of it within infrastructure and operations (I&O), and the shortage of skilled software engineering resources to comprehensively exploit it (especially in web technologies such as HTTP and JSON to develop these APIs).

User Recommendations

- Deploy a programmable infrastructure to further abstract application from infrastructure delivery and pursue an agile digital business outcome.
- Implement a programmable infrastructure by investing in infrastructure automation tools and continuous delivery (example vendors for these markets are listed below, but no single vendor or platform can enable an organizationwide programmable infrastructure strategy) leading to API-led programmable platforms.
- Invest in infrastructure and DevOps, and modernize legacy IT architectures to implement an API-driven infrastructure.
- Examine reusable programmable infrastructure building blocks leveraging programmable infrastructure strategy built on repeatable and available skills from providers.

Sample Vendors

Amazon Web Services; CU Coding; Google; IBM; Microsoft; Oracle; Quality Technology Services; RackN; Tencent; VMware

Gartner Recommended Reading

Market Guide for Servers

Predicts 2023: XaaS Is Transforming Data Center Infrastructure

Quick Answer: How Can I Optimize the Use of Programmable Platforms for Effective Software Delivery?

AIOps Platforms

Analysis By: Matt Crossley, Matthew Brisse

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

Gartner defines AIOps platform as the application of AI/ML and data analytics at the event management level in order to augment, accelerate and automate manual efforts in the event management process and associated procedures. AIOps platforms are defined by the key characteristics of cross-domain event ingestion, topology assembly, event correlation and reduction, pattern recognition, and remediation augmentation.

Why This Is Important

The combination of increasing application complexity, monitoring tool proliferation, and increasing volumes and varieties of telemetry has shifted complexity from gathering data to interpreting data. AIOps platforms apply machine learning (ML) and data analytics to classify and cluster cross-domain events in near real time, at scale, and in ways that can exceed human capacity. These inferences can augment human analysis, accelerate human response, or automate a process to resolve an issue.

Business Impact

AIOps platforms deliver value through:

- Agility and productivity: By reducing alert fatigue through identification and correlation of related events, operators can focus on fewer, more critical events.
- Service availability and triage cost: By reducing the time and effort required to identify root causes and augmenting, accelerating, or automating remediation.
- Increased value from monitoring tools: By unifying events from siloed tools and learning actionable event patterns across domains.

Drivers

Demand for AIOps platform capabilities is accelerating and is fueled by:

- **Increasing complexity:** Organizations use an increasingly complex mix of IT assets that rely on a highly integrated combination of on-premises assets, cloud IaaS/PaaS providers and SaaS platforms to deliver solutions.
- **Increasing monitoring expectations:** Investments and improvements in monitoring and the pursuit of observability are generating more data from more sources. Increasing demand and advances in monitoring trends, like application performance management (APM) and digital experience monitoring (DEM), present operators with extremely detailed views into their business applications and the end-user experience. Effective use of this additional data requires near-real-time analysis and rationalization of events from related assets and services.
- **Demands for reliability:** Shifts in roles and responsibilities driven by modern operating models, like DevOps and SRE, in the pursuit of greater availability and faster incident resolution. AIOps platforms enable agility by offloading some of the mechanical tasks of event triage, root cause analysis and solution identification. This both accelerates response for common issues and frees up human creative capacity for novel events and business priorities.

Obstacles

- **Unrealistic expectations:** Hype is a major obstacle to AIOps platform adoption. Clients struggle to separate claims of AI and magical automation from achievable use cases. This impacts demonstrating value of AIOps platforms, specifically quantifiable return on investment.
- **Maturity of dependencies:** Benefits of AIOps platforms beyond event correlation requires maturity in dependencies such as automation.
- **Time to value:** AIOps platforms learn through observation, modeling normal data patterns, and associate a solution with these patterns. This can take time depending on the frequency of occurrence. Developing accurate detection models for rare events can take months.
- **Market shifts and maturity:** Monitoring vendors are moving up the stack, AIOps platform vendors are reaching into monitoring domains, and ITSM vendors use AIOps capabilities to extend their reach. Expect further convergence and market shifts to change the definition of "state of the art."

User Recommendations

- Establish clear, realistic use cases for an AIOps platform pilot and validate them individually, rather than all at once. This approach helps reveal pockets of potential value that might be missed when evaluating only the aggregate impact. Ultimately, this fundamental step underpins an eventual strategy, while scoping the vendor landscape, clarifying technical and process dependencies, and separating hype from reality.
- Layer the AIOps features within monitoring tools with the cross-domain analysis of an AIOps platform. This approach enables efficient data ingestion and analysis, and the surfacing of insights across domains.
- Do not require automation outcomes for all AIOps applications. There is tremendous value in accelerating and augmenting human activity. These approaches often avoid the challenge of the probabilistic uncertainty combined with automated change in production environments.

Sample Vendors

BigPanda; BMC Software; Digitate; IBM; Interlink; Moogsoft; OpsRamp; PagerDuty; ServiceNow; Splunk

Gartner Recommended Reading

Market Guide for AIOps Platforms

Deliver Value to Succeed in Implementing AIOps Platforms

Infographic: Artificial Intelligence Use-Case Prism for AIOps

Infographic: AIOps Architecture for Analyzing Operational Telemetry

How Do I Plan for Migrating My Data Center Infrastructure Into an XaaS Model?

Container Management

Analysis By: Dennis Smith, Michael Warrilow

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

Gartner defines container management as offerings that enable the development and operation of containerized workloads. Delivery methods include cloud, managed service and software for containers running on-premises, in the public cloud and/or at the edge. Associated technologies include orchestration and scheduling, service discovery and registration, image registry, routing and networking, service catalog, management user interface, and APIs.

Why This Is Important

Container management automates the provisioning, operation and life cycle management of container images at scale. Centralized governance and security are used to manage container instances and associated resources. Container management supports the requirements of modern applications, including platform engineering, cloud management and continuous integration/continuous delivery (CI/CD) pipelines. Benefits include improved agility, elasticity and access to innovation.

Business Impact

Industry surveys and client interactions show that demand for containers continues to rise. This trend is due to application developers' and DevOps teams' preference for container runtimes, which use container packaging formats. Developers have progressed from leveraging containers on their desktops to needing environments that can run and operate containers at scale, introducing the need for container management.

Drivers

- The adoption of DevOps-based application development processes.
- The rise of cloud-native application architecture based on microservices.
- New system management approaches based on immutable infrastructure, which gives the ability to update systems frequently and reliably maintained in a “last known good state” rather than repeatedly patched.
- Cloud-based services built with replaceable and horizontally scalable components.
- A vibrant open-source ecosystem and competitive vendor market have culminated in a wide range of container management offerings. Many vendors enable management capabilities across hybrid cloud or multicloud environments. Container management software can run on-premises, in public infrastructure as a service (IaaS), or simultaneously in both.
- Container-related edge computing use cases have increased in industries that need to get compute and data closer to the activity (for example, telcos, manufacturing plants, etc.).
- AI/ML use cases have emerged over the past few years, leveraging the scalability capabilities of container orchestration.
- Cluster management tooling that enables the management of container nodes and clusters across different environments is increasingly in demand.
- All major public cloud service providers now offer on-premises container solutions.
- Independent software vendors (ISVs) are increasingly packaging their software for container management systems through container marketplaces.
- Some enterprises have scaled sophisticated deployments, and many more are planning container deployments. This trend is expected to increase as enterprises continue application modernization projects.

Obstacles

- More abstracted, serverless offerings may enable enterprises to forgo container management. These services embed container management in a manner that is transparent to the user.
- Third-party container management software faces huge competition in the container offerings from the public cloud providers, both with public cloud deployments and the extension of software to on-premises environments. These offerings are also challenged by ISVs that choose to craft open-source components with their software during the distribution process.
- Organizations that perform relatively little app development or make limited use of DevOps principles are served by SaaS, ISV and/or traditional application development packaging methods.

User Recommendations

- Determine if your organization is a good candidate for container management software adoption by weighing organizational goals of increased software velocity and immutable infrastructure, and its hybrid cloud requirements, against the effort required to operate third-party container management software.
- Leverage container management capabilities integrated into cloud IaaS and platform as a service (PaaS) providers' service offerings by experimenting with process and workflow changes that accommodate the incorporation of containers.
- Avoid using upstream open source (e.g., Kubernetes) directly unless the organization has adequate in-house expertise to support.

Sample Vendors

Alibaba Cloud; Amazon Web Services; Google; IBM; Microsoft; Mirantis; Red Hat; SUSE; VMware

Gartner Recommended Reading

Market Guide for Container Management

Infrastructure Automation

Analysis By: Chris Saunderson

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Mature mainstream

Definition:

Infrastructure automation (IA) enables DevOps and infrastructure and operations (I&O) teams to deliver automated infrastructure services across on-premises and cloud environments. This includes the life cycle of services through creation, configuration, operation and retirement. These infrastructure services are then made available through platform delivery, self-service catalogs, direct invocation and API integrations.

Why This Is Important

IA delivers velocity, quality, efficiency and reliability, with scalable, declarative approaches for deploying and managing infrastructure. These tools integrate into delivery pipelines targeting deployment topologies that range from on-premises to the cloud, and enable infrastructure consumers to build what is needed when they need it. Once deployed, IA provides day-2 and beyond operational automation, and extends to provide policy compliance and enforcement capabilities.

Business Impact

Implementing and maturing IA services will enable:

- **Agility** — continuous infrastructure delivery and operations
- **Productivity** — version-controlled, declarative, repeatable, efficient deployments
- **Cost improvement** — reductions in manual effort expended via increased automation
- **Risk mitigation** — compliance driven by standardized configurations
- **Collaboration** — delivering environments that product teams need with security, cost and compliance requirements baked in.

Drivers

I&O leaders must automate delivery through tool and skills investments to mature beyond simple deployments. The target should be standardized platforms that deliver the systemic, transparent management of platform deployments. This same discipline must be applied to the operation of these deployed platforms, ensuring that efficient operations (including automated incident response) can be achieved. IA tools deliver the following key capabilities to support this maturation:

- Multicloud/hybrid cloud infrastructure delivery
- Support for immutable and programmable infrastructures
- Predictable delivery enabling automated operations
- Self-service and on-demand environment creation
- Integration into DevOps initiatives (continuous integration/delivery/deployment)
- Resource provisioning, including cost optimization capabilities
- Operational configuration management efficiencies
- Policy-based delivery and assessment/enforcement of deployments against internal and external policy requirements
- Enterprise-level framework to enable maturing of automation strategies
- Skills and practice development inside infrastructure teams, enabling agile and iterative development and sustaining of services

Obstacles

- The combination of tools needed to deliver IA capability can increase tool count and complexity.
- Software engineering skills and practices are required to get maximum value from tool investments.
- IA vendor capability expansion overlaps and confuses the tool landscape, resulting in over-investment.
- Steep learning curves can cause developers and administrators to revert to familiar scripting methods to deliver required capabilities.

User Recommendations

- Identify existing IA tools in use to catalog capabilities, identify use cases and document overlaps to aid decision making.
- Assess existing internal IT skills to incorporate training needs that more fully enable IA, especially for an automation architect role to coordinate standards development and implementation.
- Baseline how managed systems and tooling will be consumed (e.g., engineer, self-service catalog, API or on-demand).
- Integrate security and compliance requirements into scope for automation and delivery activities.
- Develop an IA tooling strategy that incorporates current needs and near-term roadmap evolution.

Sample Vendors

Amazon Web Services; HashiCorp; Microsoft; Perforce; Pliant; Progress; Pulumi; RackN; Upbound; VMware

Gartner Recommended Reading

Market Guide for Infrastructure Automation Tools

Innovation Insight for Continuous Infrastructure Automation

To Automate Your Automation, Apply Agile and DevOps Practices to Infrastructure and Operations

How to Start and Scale Your Platform Engineering Team

ARO

Analysis By: Daniel Betts

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Mature mainstream

Definition:

Application release orchestration (ARO) combines deployment automation, pipeline and environment management with release orchestration capabilities to simultaneously improve the quality, velocity and governance of application releases. ARO enables organizations to automate and scale release activities across multiple diverse teams (e.g., DevOps), technologies, development methodologies (e.g., agile), delivery patterns (e.g., continuous), pipelines, processes and toolchains.

Why This Is Important

Demand is growing and will continue to grow, for new applications and features delivered faster to support business agility. The resulting tumultuous and transformative activity (often in the form of DevOps/DevSecOps initiatives) has created multiple buyers for ARO capabilities. These buyers often desperately need ARO's cohesive value, yet are challenged to articulate and/or gain consensus around the business criticality of release activities to drive adoption.

Business Impact

ARO tools provide increased transparency in the release management process by making bottlenecks and wait for states visible in areas such as infrastructure provisioning or configuration management. Once these constraints are visible and quantifiable, business value decisions can be made to address them and measure improvement. This speeds the realization of direct business value, as new applications and enhancements/bug fixes can be more quickly and reliably delivered.

Drivers

- Agility and productivity gains: Faster delivery of new applications and updates in response to changing market demands.
- Cost reduction: Significant reduction of manual interactions by high-skill and high-cost staff, freeing them to work on higher-value activities.
- Risk mitigation: Consistent use of standardized, documented processes and configurations across multiple technology domains.
- Improvement and remediation: Use of dashboard views over metrics outlining and predicting release quality and throughput.
- Improved visibility and traceability into the release process.

Gartner

Obstacles

- The need to map capabilities to the client's internal delivery challenges or opportunities.
- Vendors in the DevOps toolchain market are building ARO features into platform solutions, with ARO as a subset of capabilities that can hide their value.
- Challenges in capturing the release orchestration space, due to maturity (feature parity across supplier platforms) in the market.

User Recommendations

- Organize activities into three categories: deployment automation, pipeline and environment management, and release orchestration.
- Explore DevOps platforms that provide ARO along with other native capabilities.
- Prioritize capabilities for current and future needs prior to evaluating vendors. When evaluating ARO tools for selection, prioritize tool features and map capabilities to requirements. Where legacy environments exist, those should be weighted more heavily.
- Simplify and speed up the transition to automated workflows by documenting current application release procedures, activities and artifacts performed by both traditional and DevOps teams.
- Confirm the availability of an ARO platform dashboard, with release performance and underlying platform metrics.

Sample Vendors

CloudBees; Digital.ai; GitLab; Harness; Microsoft; Plutora

Gartner Recommended Reading

Market Guide for Value Stream Delivery Platforms

Market Guide for Value Stream Management Platforms

How to Build and Evolve Your DevOps Toolchains

Beware the DevOps Toolchain Debt Collector

Climbing the Slope

Feature Management

Analysis By: Keith Mann

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

A feature is a discrete unit of software that provides a single, valuable function or capability within a larger system. Feature management combines the selective enabling or disabling of features via feature toggles (also known as feature flags) with the monitoring, assessment, and comparison of features and feature variants.

Why This Is Important

Feature management enables the use of feature toggles on a very large scale without incurring technical debt, which in turn enables them to be used for more purposes. Although feature toggles were originally used to disable features that were not ready for deployment, they are now also used for experimentation, testing and progressive release. Features are increasingly being monitored and tested for value throughout their life cycle, creating feedback loops that progress feature design.

Business Impact

Almost all businesses can benefit from faster, more reliable delivery of software — the impact of feature management in that sense could be universal. Experimentation has already had a large impact on digital marketing and digital consumer product development, where human responses are hard to predict. The biggest potential impact comes from the improved ability to select valuable features for development, based on feedback loops from feature monitoring in production to development planning.

Drivers

- Software engineering teams are reaching high levels of maturity in their use of agile and DevOps practices. This enables them to deliver software even faster than it can be deployed and consumed. Software engineering organizations can use feature management to decouple delivery from deployment and avoid slowing down teams.

- Demand for continuously available systems is increasing. This drives a need for instant recovery from defective features, which in turn fuels a need for feature toggles and feature management.
- With new technologies and corresponding new user experience options comes uncertainty about which types of experience are most effective for users. Feature management enables experiments that reduce this uncertainty by presenting various experiences to various sample audiences and measuring their responses.
- Advanced software engineering organizations are finding that fast delivery alone does not satisfy stakeholders. They are beginning to realize that they must establish feedback loops from production to feature design so that they can learn which designs produce the greatest value. Feature management enables the value of features to be monitored, tracked and compared.
- Speed to market pressures can be met in part through progressive releases — which enable features to be rolled out as their capabilities, such as localization, expand to meet the needs of new user segments.

Obstacles

- Even years after its genesis, the feature management market is still dominated by fairly small vendors. This limits user awareness and adoption.
- Feature management vendors are just beginning a shift from a technical to a business focus in their messaging. Technology buyers, like software engineering leaders, are often responsible for justifying the investment.
- The vision of feature management as a tool for experimentation and value feedback is still new. The focus has long been on the speed and reliability of deployment, so buyers view that as the benefit of feature management. Shifting the message is proving to be hard.
- Although value feedback loops could be the most important application of feature management, the concept and corresponding practices are still emerging.
- As with many areas of software engineering, low talent availability is a problem. Data scientists and other professionals skilled in experimentation are in short supply and high demand.

User Recommendations

- Ensure that development teams are familiar with feature management techniques. Ideally, developers will already be using feature toggles, where appropriate, to improve deployment speed and reliability.
- Promote the use of feature management for experimentation. This may mean selecting a feature management vendor and training or hiring staff.
- Introduce the concept of value feedback loops to software designers. Establish value feedback loops as tools and practices mature.
- Have software designers clearly identify features during solution design, and define the value of each feature.
- Adopt progressive release practices to reduce change impacts and enable earlier delivery.

Sample Vendors

Amplitude; Kameleoon; LaunchDarkly; Optimizely; Split; Unleash

Gartner Recommended Reading

Software Engineering Teams Must Learn to Deliver More Value

Data-Driven DevOps: How to Rethink the Measurement Approach

Adopt Platform Engineering to Improve the Developer Experience

Immutable Infrastructure

Analysis By: Neil MacDonald, Tony Harvey

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Early mainstream

Definition:

Immutable infrastructure is a process pattern (not a technology) in which the system and application infrastructure, once deployed, are never updated in place. Instead, when changes are required, the infrastructure and applications are simply updated and redeployed through the CI/CD pipeline.

Why This Is Important

Immutable infrastructure ensures the system and application environment, once deployed, remains in a predictable, known-good-configuration state. It simplifies change management, supports faster and safer upgrades, reduces operational errors, improves security, and simplifies troubleshooting. It also enables rapid replication of environments for disaster recovery, geographic redundancy or testing. This approach is easier to adopt with cloud-native applications.

Business Impact

Taking an immutable approach to workload and application management simplifies automated problem resolution by reducing the options for corrective action to, essentially, just one — repair the application or image in the development pipeline and rerelease. The result is an improved security posture and a reduced attack surface with fewer vulnerabilities and a faster time to remediate when new issues are identified.

Drivers

- Linux containers and Kubernetes are being widely adopted. Containers improve the practicality of implementing immutable infrastructure due to their lightweight nature, which supports rapid deployment and replacement.
- The GitOps deployment pattern, which emphasizes continuously synchronizing the running state to the software repository, has become an effective way to implement immutable infrastructure in Kubernetes-based, containerized environments.
- Infrastructure as code (IaC) tools (including first-party cloud provider IaC tools) have increasingly integrated configuration drift detection and correction, improving the practicality of implementing immutable infrastructure across an application's entire stack and environment.
- Interest in zero-trust and other advanced security postures where immutable infrastructure can be used to proactively regenerate workloads in production from a known good state (assuming compromise), a concept referred to as "systematic workload reprovisioning."
- For cloud-native application development projects, immutable infrastructure simplifies change management, supports faster and safer upgrades, reduces operational errors, improves security, and simplifies troubleshooting.

Obstacles

- The use of immutable infrastructure requires a strict operational discipline that many organizations haven't yet achieved, or have achieved for only a subset of applications.
- IT administrators are reluctant to give up the ability to directly modify or patch runtime systems.
- Applying the immutable infrastructure pattern is most easily done for stateless components. Stateful components, especially data stores, represent special cases that must be handled with care.
- Implementing immutable infrastructure requires a mature automation framework, up-to-date blueprints and bills of materials, and confidence in your ability to arbitrarily recreate components without negative effects on user experience or loss of state.
- Many enterprise applications are stateful applications deployed on virtual machines. These applications are oftentimes commercial off-the-shelf and are not designed for fully automated installation when redeployed.

User Recommendations

- Reduce or eliminate configuration drift by establishing a policy that no software, including the OS, is ever patched in production. Updates must be made to individual components, versioned in a source-code-control repository, then redeployed.
- Prevent unauthorized change by turning off all administrative access to production compute resources. Examples of this might include not permitting Secure Shell or Remote Desktop Protocol access.
- Adopt immutable infrastructure principles with cloud-native applications first. Cloud-native workloads are more suitable than traditional on-premises workloads.
- Treat scripts, recipes and other codes used for infrastructure automation similar to the application source code itself, as this mandates good software engineering discipline.
- Include immutable infrastructure scripts, recipes, codes and images in your backup and ransomware recovery plans as they will be your primary source to rebuild your infrastructure after an infection.

Sample Vendors

Amazon Web Services; Google; HashiCorp; Microsoft; Perforce; Progress; Red Hat; Snyk; Turbot; VMware

Gartner Recommended Reading

Comparing DevOps Architecture to Automate Infrastructure and Operations for Software Development

2022 Strategic Roadmap for Compute Infrastructure

To Automate Your Automation, Apply Agile and DevOps Practices to Infrastructure and Operations

Innovation Insight for Continuous Infrastructure Automation

Market Guide for Cloud-Native Application Protection Platforms

Continuous Delivery

Analysis By: Hassan Ennaciri

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Adolescent

Definition:

Continuous delivery (CD) is a software engineering approach that enables teams to build critical software quickly, while ensuring the software can be released reliably anytime. Through dependable, low-risk releases, CD allows continuous adaptation of the software to incorporate user feedback, market shifts and business strategy changes. This approach requires the engineering discipline to facilitate complete automation of the software delivery pipeline.

Why This Is Important

The growing success of DevOps initiatives continues to drive investments in CD capabilities. CD improves software release velocity and reliability, while simplifying compliance enforcement via automation. It is a prerequisite and the first step to continuous software deployments for organizations that aspire to push changes with zero downtime.

Business Impact

CD is a key practice for a DevOps initiative as it reduces the build-to-production cycle time. As a result, it accelerates the positive impact of new applications, functions, features and fixes by increasing velocity across the application life cycle. The positive impacts include improved business delivery and end-user satisfaction, improved business performance and agility, and risk mitigation via rapid delivery of updates.

Drivers

- Increased adoption of Agile and DevOps practices to deliver solutions.
- Pressure from digital business to improve release velocity and reliability.
- Additional compliance requirements that require automation and orchestration of release activities for better traceability and auditability.
- The need to improve delivery outcomes to deploy application builds and updates more consistently, by extending the benefits of continuous integration (CI) and automated testing to continuously build deployable software.

Obstacles

- Organizational culture and collaboration between teams with different roles and skills are major barriers to CD success. Agile practices that helped bridge the gap between business and development must be extended to deployment, environment configuration, monitoring, and support activities.
- Lack of value stream mapping of product delivery hinders visibility and quick feedback loops for continuous improvements. Teams struggle to improve and focus on value work, as they don't have insights into the critical steps in the process, the time each step takes, handoffs, and wait states.
- Manual steps and processes involved in deploying to production environments impact software flow delivery.
- Other challenges impacting the success of CD include application architecture, lack of automation in all areas of testing, environment provisioning, configuration security and compliance.

User Recommendations

- Evaluate all associated technologies when you start a CD initiative and take an iterative approach to adoption. This will require collaboration with different stakeholders from the product, development, security and operations teams.
- Establish consistency across application environments for a higher likelihood of success and implement a continuous improvement process that relies on value stream metrics.
- Evaluate and invest in associated tooling, such as application release orchestration tools, containers, and infrastructure automation tools. These tools provide some degree of environment modeling and management, which can prove invaluable for scaling CD capabilities across multiple applications.
- Explore a DevOps platform that provides fully integrated capabilities and enables continuous delivery of software.

Sample Vendors

Broadcom; CloudBees; GitLab; Harness; JFrog; Red Hat

Gartner Recommended Reading

How to Build and Evolve Your DevOps Toolchains

Market Guide for Value Stream Management Platforms

Beware the DevOps Toolchain Debt Collector

Entering the Plateau

Continuous Configuration Automation

Analysis By: Chris Saunderson

Benefit Rating: High

Market Penetration: More than 50% of target audience

Maturity: Mature mainstream

Definition:

Continuous configuration automation (CCA) tools enable infrastructure administrators and developers to automate the deployment, configuration and operation of systems and software. They support the definition, deployment and maintenance of configuration states and settings. Most CCA tools have an open-source heritage, and most offer commercial support.

Why This Is Important

CCA tools are critical to delivering operational efficiency. They enable automation and DevOps initiatives by deploying and managing infrastructure elements, associated software and configuration changes as code. In combination with infrastructure automation tools, CCA tools form the core of infrastructure-as-code (IaC) capabilities. They also enable the automation of Day 2 operations of deployed systems, expanding their reach into networking, containers, compliance and security use cases.

Business Impact

By enabling automated deployment and configuration of systems, settings and software programmatically, organizations realize:

- **Agility improvements:** CI/CD enablement for infrastructure and operations (I&O) services.
- **Productivity gains:** Repeatable, declarative, version-controlled infrastructure deployment/operation.
- **Cost optimization:** Reductions in manual interventions by skilled staff.
- **Risk mitigation:** Compliance assessment and remediation using standardized processes Improved change success and reliability are also realized.

Drivers

- Organizations need a broader set of deployment and automation functions beyond configuration management, including IaC, patching, application release orchestration (ARO), configuration assessment and auditing (e.g., for regulatory or internal policy compliance) and orchestration of operational tasks.
- CCA provides an automation framework that enables deployment automation and Day 2 operational automation of changes, compliance, and response to incidents or problems.
- CCA tools are essential for I&O administrators to mature from task-based scripting to a more structured approach to automation and delivery.
- There is a need for repeatable, declarative, standardized deployments to be made available to end users or I&O administrators that enable quick delivery of infrastructure to meet end-user needs and I&O policies and baselines.

Obstacles

- Confusion around the capabilities and overlaps of automation tools causing conflict and overinvestment.
- Developers and administrators may use CCA in a silo, further inhibiting enterprisewide adoption.
- IT skill sets hinder the adoption of these tools, requiring source code management and software engineering skills to make full use of capabilities.
- The growing use of IaC tools has created confusion about the role of CCA tools; however, CCA tools are necessary to deliver effective and efficient IaC.

User Recommendations

- Clarify the role that CCA tools fulfill in their toolchain and make selections based on the tasks that are in scope.
- Evaluate the availability of content against organizational use cases. Prioritize CCA tools that provide out-of-the-box content that addresses current pain points and accelerates time to value.
- Include both professional services for enablement and training requirements in cost evaluations. Costs associated with CCA tools extend beyond just the licensing cost.

- Expect to invest in training beyond tool implementation to fully realize the benefits of these tools.
- Guard against developers and administrators reverting to known imperative scripting methods to complete specific tasks in place of using CCA capabilities.
- Maximize the value of CCA tool investments by ensuring that your organization's culture can embrace CCA tools strategically and automate toil (for DevOps and I&O leaders).

Sample Vendors

Inedo; Perforce (Puppet); Progress; Red Hat; Rudder; VMware

Gartner Recommended Reading

Market Guide for Infrastructure Automation Tools

To Automate Your Automation, Apply Agile and DevOps Practices to Infrastructure and Operations

Innovation Insight for Continuous Infrastructure Automation

APM

Analysis By: Padraig Byrne, Mrudula Bangera

Benefit Rating: High

Market Penetration: More than 50% of target audience

Maturity: Mature mainstream

Definition:

Application performance monitoring (APM) enables the observation of an application's behavior, dependencies, users and business KPIs throughout its life cycle. The application being observed may be developed internally, as a packaged application or as software as a service (SaaS).

Why This Is Important

The APM market continues to evolve beyond its core root of server-side application monitoring as organizations seek to optimize business outcomes, enhance user experience and improve application performance. It is no longer sufficient to monitor one aspect of the technology stack; nor is it enough to deploy proprietary technologies to collect performance data. Modern APM implementations are becoming more tightly integrated with observability platforms.

Business Impact

APM solutions enable businesses to examine modern applications' end-to-end performance, coupled with detailed inspections to quickly identify service-impacting outages. As organizations continue to embrace digital transformation, their need increases for agility in order to succeed with their transformation initiatives. APM solutions can be perceived as more than another monitoring tool, supporting the need for agility and aiding in its acceleration and effectiveness.

Drivers

- **Unified monitoring:** New application monitoring and observability tools are becoming more unified. This approach requires platforms that share common data models to conduct correlation analysis and other critical functions of application performance monitoring.

- **Holistic monitoring:** Modern tools are becoming more holistic in terms of the types of data they can ingest, analyze and integrate. The continued adoption of new application development and operations technologies requires monitoring teams to constantly test the limits of their monitoring products.

- **Integration with DevOps and site reliability engineering (SRE):** Testing in preproduction and integration with continuous integration/continuous delivery (CI/CD) tools have become the new norm, increasing the quality and robustness of the finished product.

- **Intelligent monitoring:** The use of logs, traces, metrics and multiple other types of telemetry is enabling operations and monitoring teams to find unexpected patterns in high-volume, multidimensional datasets using artificial intelligence for IT operations (AIOps) technologies.

- **Business monitoring:** APM tools can be used to derive business metrics, such as abandoned shopping carts or average spend per customer for retailers. Representing such critical business information means an increased likelihood of further investment in these tools.

Obstacles

- Traditional implementations of APM often fail to provide a complete solution, requiring organizations to pivot between tools, wasting time and resources, while struggling to find the root cause of the problem.
- Modern architectures such as containers and microservices and cloud-native environments are coming to IT operations environments faster than monitoring strategies are evolving to handle them. This is leading to visibility gaps and performance challenges.
- Clients increasingly cite cost as a significant challenge for implementation of APM tools. Costs for larger organizations can run into millions per year, representing a significant percentage of IT spend.
- Many IT monitoring teams still rely on manually invoked runbooks or scripts to remediate problems, hindering I&O leaders' ability to deploy and monitor new technologies. There is often a major disconnect between what I&O monitors and what the business cares about, which can have significant negative implications for the business.

User Recommendations

- Choose vendors that assist in relating application performance to business objectives and serve not only IT operations (ITOps), but also DevOps, application owners and lines of business, providing value throughout the application life cycle. Select a vendor that provides actionable answers and not just endless drill-downs to more data.
- Choose products based on their ability to support: mapping and monitoring of customer and business journeys; bidirectional integration with the DevOps toolchain; new emerging standards in instrumentation, such as OpenTelemetry; cloud-native monitoring with an API-first approach; application security; and integrations with your existing or planned IT service management (ITSM) and configuration management database (CMDB) tools.

Sample Vendors

Cisco; Datadog; Dynatrace; Elastic; Grafana; Instana; New Relic; Splunk

Gartner Recommended Reading

Magic Quadrant for Application Performance Monitoring and Observability

Cloud Management Platforms

Analysis By: Dennis Smith

Benefit Rating: Low

Market Penetration: 5% to 20% of target audience

Maturity: Mature mainstream

Definition:

Cloud management platforms (CMPs) enable organizations to manage private, public and multicloud services and resources. Their functionality combines provisioning and orchestration; service request management; inventory and classification; monitoring and analytics; cost management and resource optimization; cloud migration; backup and disaster recovery; and identity, security and compliance. Functionality can be provided by a single product or a set of offerings with some degree of integration.

Why This Is Important

Enterprises and managed service providers will deploy CMPs to increase agility, reduce the cost of providing services and increase the likelihood of meeting service levels. CMPs promote cost-effective governance and accountability through self-service interfaces, automation, and adherence to standard best practices. They also play an important role in creating a unified layer of consumption and abstraction for organizations adopting a hybrid and multicloud model.

Business Impact

CMPs address issues related to aggregation, integration, customization and governance in the adoption of hybrid multicloud by organizations. The recent CMP market continually focuses on preventing enterprises from overspending or leaving themselves vulnerable to security issues — two key items to avoid when adopting cloud services. CMPs also address the need of managed cloud service providers for multitenant customer operations and support through a single management portal.

Drivers

- Organizations are acknowledging the need to foster end-user self-service provisioning with embedded governance.
- Vendors are addressing the key issues enterprises face. Many vendors are looking to combine cost management and security functionality into governance tooling. A few vendors are also looking to provide infrastructure-as-code assistance by overlaying cloud management functionality to this capability.
- Many vendors have added container management to their CMP offerings. The ability to serve both application developers, and infrastructure and operations (I&O) personas is key. This requires CMPs to be linked into the application development process to enable I&O teams to enforce provisioning standards, without imposing a workflow that inhibits agility.
- Global system integrators (GSIs) and management service providers (MSPs) leverage CMPs to build a services layer in order to create multicloud managed services.

Obstacles

- Cloud management comprises a complex and varied set of activities, and no CMP platform has been successful at addressing all customer needs.
- The CMP market has been gradually fragmenting, moving from integrated all-in-one tools to specialty tools that focus on a subset of capabilities, such as security, operations or financial management.
- Challenges vendors face include interfacing with multiple public clouds, cost transparency with workload optimization to remediate cost overruns, handling newer functions (e.g., containers and serverless deployments), and edge computing.
- Market dynamics have caused confusion among potential buyers. There have been acquisitions by CMP vendors and vendors in adjacent markets, combining CMP functionality with their existing functionality. Cloud service providers (CSPs) and MSPs have entered the market, and many long-standing vendors have introduced new products that target gaps in their previous ones.

User Recommendations

- Identify your full vertical (infrastructure as a service [IaaS], platform as a service [PaaS] and SaaS) and horizontal (on-premises, public cloud and edge) needs.
- Choose a single-focused specialized tooling, if your requirements do not expand beyond a limited scope.
- Select between native cloud services if you value depth with a cloud provider, or CMPs if you want breadth across cloud providers and your on-premises deployment.
- Determine the utility of functionally-focused tools by defining the organization's functionality needs. Integrate cloud management or traditional management tools because no vendor has a total cloud management solution.
- Ensure staffing to operate CMP platforms by planning new roles (e.g., cloud engineers and/or operators).
- Develop skills in financial and capacity management.
- Align with GSIs and MSPs to embed your CMP in their offerings as a key component of their multicloud managed services offering.

Sample Vendors

Arrow Electronics; CloudBolt; Kion; Morpheus Data; Snow Software; Turbot; VMware; Wipro Enterprises

Gartner Recommended Reading

Market Guide for Cloud Management Tooling

Market Guide for Container Management

6 Best Practices to Create a Cloud Management Services Offering in the World of Multicloud and Hybrid Cloud

Appendixes

See the previous Hype Cycle: Hype Cycle for Agile and DevOps, 2022

Hype Cycle Phases, Benefit Ratings and Maturity Levels

Table 2: Hype Cycle Phases

(Enlarged table in Appendix)

Phase ↓	*Definition* ↓
Innovation Trigger	A breakthrough, public demonstration, product launch or other event generates significant media and industry interest.
Peak of Inflated Expectations	During this phase of overenthusiasm and unrealistic projections, a flurry of well-publicized activity by technology leaders results in some successes, but more failures, as the innovation is pushed to its limits. The only enterprises making money are conference organizers and content publishers.
Trough of Disillusionment	Because the innovation does not live up to its overinflated expectations, it rapidly becomes unfashionable. Media interest wanes, except for a few cautionary tales.
Slope of Enlightenment	Focused experimentation and solid hard work by an increasingly diverse range of organizations lead to a true understanding of the innovation's applicability, risks and benefits. Commercial off-the-shelf methodologies and tools ease the development process.
Plateau of Productivity	The real-world benefits of the innovation are demonstrated and accepted. Tools and methodologies are increasingly stable as they enter their second and third generations. Growing numbers of organizations feel comfortable with the reduced level of risk; the rapid growth phase of adoption begins. Approximately 20% of the technology's target audience has adopted, or is adopting, the technology as it enters this phase.
Years to Mainstream Adoption	The time required for the innovation to reach the Plateau of Productivity.

Source: Gartner (July 2023)

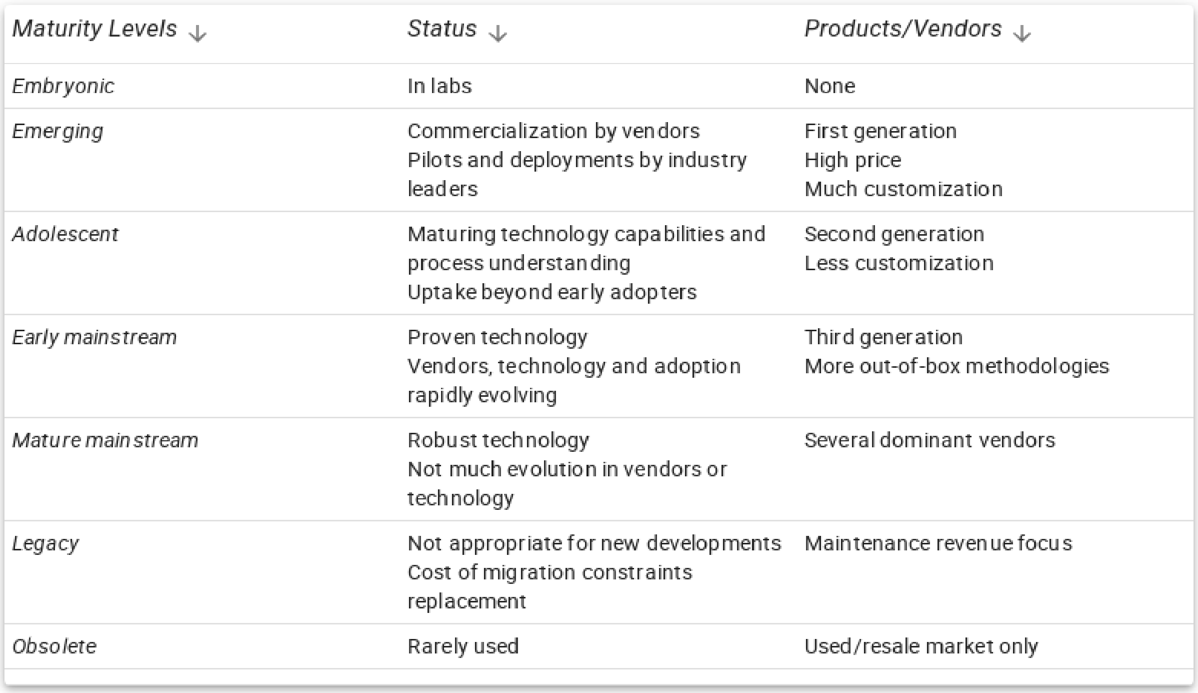
Table 3: Benefit Ratings

Benefit Rating ↓	*Definition* ↓
Transformational	Enables new ways of doing business across industries that will result in major shifts in industry dynamics.
High	Enables new ways of performing horizontal or vertical processes that will result in significantly increased revenue or cost savings for an enterprise.
Moderate	Provides incremental improvements to established processes that will result in increased revenue or cost savings for an enterprise.
Low	Slightly improves processes (e.g., improved user experience) that will be difficult to translate into increased revenue or cost savings.

Source: Gartner (July 2023)

Table 4: Maturity Levels

(Enlarged table in Appendix)



Maturity Levels ↓	*Status* ↓	*Products/Vendors* ↓
Embryonic	In labs	None
Emerging	Commercialization by vendors Pilots and deployments by industry leaders	First generation High price Much customization
Adolescent	Maturing technology capabilities and process understanding Uptake beyond early adopters	Second generation Less customization
Early mainstream	Proven technology Vendors, technology and adoption rapidly evolving	Third generation More out-of-box methodologies
Mature mainstream	Robust technology Not much evolution in vendors or technology	Several dominant vendors
Legacy	Not appropriate for new developments Cost of migration constraints replacement	Maintenance revenue focus
Obsolete	Rarely used	Used/resale market only

Source: Gartner (July 2023)

Acronym Key and Glossary Terms

AIOps	Artificial intelligence for IT operations
APM	Application performance monitoring
ARO	Application release orchestration
DORA	DevOps Research and Assessment
SDI	Software-defined infrastructure
VSMP	Value stream management platform

Document Revision History

Hype Cycle for Agile and DevOps, 2022 - 13 July 2022

Hype Cycle for Agile and DevOps, 2021 - 12 July 2021

Hype Cycle for Agile and DevOps, 2020 - 15 July 2020

Hype Cycle for DevOps, 2019 - 17 July 2019

Hype Cycle for DevOps, 2018 - 24 July 2018

Hype Cycle for DevOps, 2017 - 14 July 2017

Hype Cycle for DevOps, 2016 - 26 July 2016

Recommended by the Authors

Some documents may not be available as part of your current Gartner subscription.

Understanding Gartner's Hype Cycles

Tool: Create Your Own Hype Cycle With Gartner's Hype Cycle Builder

Why DevOps Success Requires Platform Teams

Market Guide for Value Stream Management Platforms

Magic Quadrant for DevOps Platforms

Quick Answer: What Is Site Reliability Engineering?

Table 1: Priority Matrix for Agile and DevOps, 2023

Benefit	*Years to Mainstream Adoption*			
↓	***Less Than 2 Years*** ↓	***2 - 5 Years*** ↓	***5 - 10 Years*** ↓	***More Than 10 Years*** ↓
Transformational		AI-Augmented Software Engineering Observability Platform Engineering Site Reliability Engineering	Software Supply Chain Security	

Benefit	*Years to Mainstream Adoption*			
↓	***Less Than 2 Years*** ↓	***2 - 5 Years*** ↓	***5 - 10 Years*** ↓	***More Than 10 Years*** ↓
High	APM Continuous Configuration Automation	AI-Augmented Testing ARO Communities of Practice Container Management Continuous Delivery Continuous Quality DevOps Platforms Feature Management GitOps Infrastructure Automation Internal Developer Portal Performance Engineering Policy as Code Programmable Infrastructure Value Stream Management Platforms	AIOps Platforms Cloud Development Environments DesignOps Infrastructure Platform Engineering Observability-Driven Development	Promise Theory
Moderate		Continuous Compliance Automation DevOps Test Data Management Immutable Infrastructure	Chaos Engineering	

Benefit	*Years to Mainstream Adoption*			
↓	***Less Than 2 Years*** ↓	***2 - 5 Years*** ↓	***5 - 10 Years*** ↓	***More Than 10 Years*** ↓
Low	Cloud Management Platforms			

Source: Gartner (July 2023)

Table 2: Hype Cycle Phases

Phase ↓	*Definition* ↓
Innovation Trigger	A breakthrough, public demonstration, product launch or other event generates significant media and industry interest.
Peak of Inflated Expectations	During this phase of overenthusiasm and unrealistic projections, a flurry of well-publicized activity by technology leaders results in some successes, but more failures, as the innovation is pushed to its limits. The only enterprises making money are conference organizers and content publishers.
Trough of Disillusionment	Because the innovation does not live up to its overinflated expectations, it rapidly becomes unfashionable. Media interest wanes, except for a few cautionary tales.
Slope of Enlightenment	Focused experimentation and solid hard work by an increasingly diverse range of organizations lead to a true understanding of the innovation's applicability, risks and benefits. Commercial off-the-shelf methodologies and tools ease the development process.
Plateau of Productivity	The real-world benefits of the innovation are demonstrated and accepted. Tools and methodologies are increasingly stable as they enter their second and third generations. Growing numbers of organizations feel comfortable with the reduced level of risk; the rapid growth phase of adoption begins. Approximately 20% of the technology's target audience has adopted, or is adopting, the technology as it enters this phase.
Years to Mainstream Adoption	The time required for the innovation to reach the Plateau of Productivity.

Phase ↓	*Definition* ↓

Source: Gartner (July 2023)

Table 3: Benefit Ratings

Benefit Rating ↓	*Definition* ↓
Transformational	Enables new ways of doing business across industries that will result in major shifts in industry dynamics.
High	Enables new ways of performing horizontal or vertical processes that will result in significantly increased revenue or cost savings for an enterprise.
Moderate	Provides incremental improvements to established processes that will result in increased revenue or cost savings for an enterprise.
Low	Slightly improves processes (e.g., improved user experience) that will be difficult to translate into increased revenue or cost savings.

Source: Gartner (July 2023)

Table 4: Maturity Levels

Maturity Levels ↓	*Status* ↓	*Products/Vendors* ↓
Embryonic	In labs	None
Emerging	Commercialization by vendors Pilots and deployments by industry leaders	First generation High price Much customization
Adolescent	Maturing technology capabilities and process understanding Uptake beyond early adopters	Second generation Less customization
Early mainstream	Proven technology Vendors, technology and adoption rapidly evolving	Third generation More out-of-box methodologies
Mature mainstream	Robust technology Not much evolution in vendors or technology	Several dominant vendors
Legacy	Not appropriate for new developments Cost of migration constraints replacement	Maintenance revenue focus
Obsolete	Rarely used	Used/resale market only

Source: Gartner (July 2023)