Hype Cycle for Software Engineering, 2021

Published 27 July 2021 - ID G00747398 - 106 min read

By Analyst(s): Abhishek Singh, Mark O'Neill

Initiatives: Software Engineering Technologies; Software Engineering Strategies

Software engineering is a fast-evolving discipline underpinning the digital products of many businesses. Software engineering leaders can use the technologies and practices in this Hype Cycle to facilitate agility and innovation within their organizations.

Strategic Planning Assumption(s)

Analysis

What You Need to Know

Software engineering leaders need to keep up with the pace of emerging technologies, while at the same time extracting value from existing solutions. This places significant demands on their teams.

For software engineering teams, areas of interest include:

- Talent
- Security
- Team morale
- Open-source software
- Low code options
- Managing business stakeholders
- New application development
- Modernizing legacy solutions

Some of these areas complement each other, while others are in conflict. Software engineering leaders must use the right skills, tools, technologies and practices to deliver greater customer value. They must ensure that their teams, and the applications they deliver, become more resilient, adaptable and effective.

The Hype Cycle

Gartner's Hype Cycle for software engineering captures key technologies and practices that help software engineering leaders achieve agility and innovation as part of their business transformation initiatives. The 30 innovation profiles in this Hype Cycle represent the evolution of software engineering over the near term and beyond. Key trends include:

- Incorporating AI-augmented software engineering practices throughout the development life cycle.
- Designing delivery assets by using the principles of DesignOps.

- Incorporating observability tools for effective monitoring and troubleshooting.
- Supporting multiexperience development for modern applications — desktop, mobile, wearable, and conversational.
- Incorporating graph thinking, event stream processing and other modern approaches to software architecture.
- Improving application quality during each DevOps phase by using shift-left and shift-right testing practices.

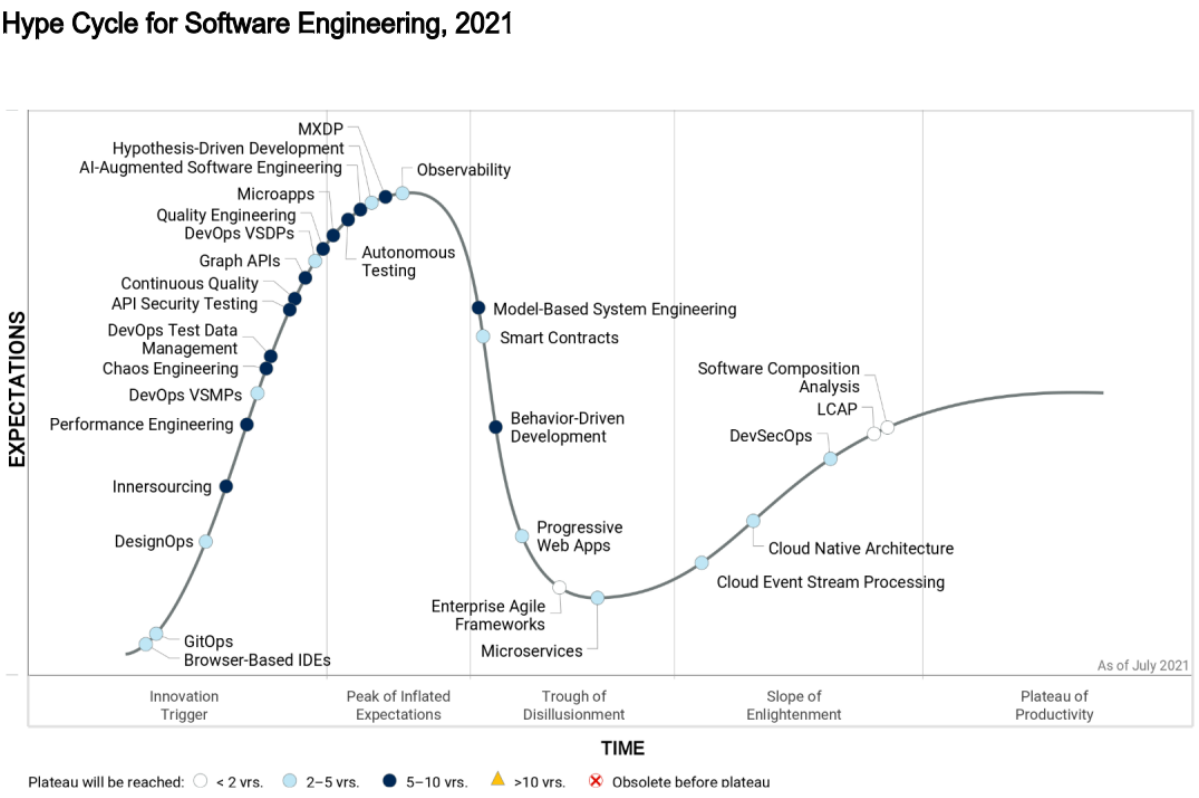
At the Innovation Trigger stage of the Hype Cycle, we see new technologies including GitOps and browser-based integrated development environments (IDEs). These increase automation and productivity for software engineering teams. At the Peak of Inflated Expectations, we see a group of technologies including observability and AI-augmented software engineering. Their position on the Hype Cycle indicates that they show great promise, but that there is currently significant hype around these technologies. Microservices has reached the bottom of the Trough of Disillusionment, reflecting the disappointment Gartner has heard from many organizations whose software engineering teams have created overly-complex microservices architectures.

Looking to the rightmost side of the Hype Cycle, approaching the Plateau of Productivity, we see software composition analysis (SCA) tools, which automate open-source software (OSS) risk management. These tools are making it safer for application leaders to encourage their teams to experiment with OSS-powered innovations. With many enterprises successfully adopting multiple low-code application platform (LCAP) solutions and cloud native architectures, we are seeing these technologies advance through the Plateau of Productivity.

In addition to technology innovation, software engineering practices are evolving quickly. Software engineering leaders are evaluating practices including performance engineering, continuous quality, and quality engineering to inject the discipline of quality and performance improvement across all the stages of the software development life cycle (SDLC). Increased user expectations for application design, quality and experience are pushing forward adoption of development techniques such as hypothesis-driven development and behavior-driven development.

Incidents of attacks on APIs are forcing organizations to put greater emphasis on API security testing during software development. Finally, smart contracts provide the ability to develop software right on the blockchain, but are falling down toward the Trough of Disillusionment due to complexity and lack of legal readiness.

Figure 1: Hype Cycle for Software Engineering, 2021



Source: Gartner (July 2021)

Downloadable graphic: Hype Cycle for Software Engineering, 2021

The Priority Matrix

The Priority Matrix shows the benefit levels and the number of years to mainstream adoption for the innovation profiles presented in this Hype Cycle.

"Transformational" innovations can have a significant impact on an organization's business models, driving a need for new strategies and tactics. AI-augmented software engineering, for example, will change the way modern software engineers will deliver code. Smart contracts have the potential to automate the business transaction by executing transactions under mutually agreed terms embodied in the code. Practices such as DevSecOps and the use of observability tools will help ship reliable software without compromising on security and compliance.

High-benefit innovations are less likely to change an organization's business model, but will have a significant impact in software engineering. Many of these innovations, such as DesignOps, DevOps, the use of browser-based IDEs and microservices adoption — will provide a lot of benefits during the next two to five years.

Although it may take five to 10 years for them to achieve mainstream adoption, innovations such as chaos engineering, performance engineering and quality engineering are more strategic for software delivery and require long-term planning and investment.

Table 1: Priority Matrix for Application Architecture and Development, 2020

(Enlarged table in Appendix)

Benefit	*Years to Mainstream Adoption*			
↓	*Less Than 2 Years* ↓	*2 - 5 Years* ↓	*5 - 10 Years* ↓	*More Than 10 Years* ↓
Transformational		DevSecOps Observability Smart Contracts	AI-Augmented Software Engineering Model-Based System Engineering	
High	Enterprise Agile Frameworks LCAP Software Composition Analysis	Browser-Based IDEs DesignOps DevOps VSDPs DevOps VSMPs GitOps Microservices	API Security Testing Autonomous Testing Chaos Engineering Continuous Quality MXDP Performance Engineering Quality Engineering	
Moderate		Cloud Event Stream Processing Cloud Native Architecture Hypothesis-Driven Development Progressive Web Apps	Behavior-Driven Development DevOps Test Data Management Graph APIs Innersourcing Microapps	
Low				

Source: Gartner (July 2021)

On the Rise

Browser-Based IDEs

Analysis By: Manjunath Bhat

Benefit Rating: High

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

Browser-based integrated development environments (IDEs) are consumed “as a service.” They enable browser-based remote access to a complete development environment, which obviates the need for local installation/configuration. This decouples the development workspace from the physical workstation, enabling a consistent experience across devices. The workspace comprises code editors, debuggers, code review, collaboration tools, source control repositories and integrated CI/CD pipelines.

Why This Is Important

Browser-based IDEs provide consistent, secure access to preconfigured development workflows to developers. This frees them from setting up their own environments, eliminating the need to install and maintain prerequisites, software development kits (SDKs), security updates and workstation plug-ins. Browser-based IDEs 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

Browser-based IDEs are becoming popular, due to their native integration with Git-based repositories and continuous integration/continuous delivery (CI/CD) tools, accelerating DevOps adoption. Browser-based IDEs enable IT to centrally manage and secure access to development environments, even from personal devices, minimizing code exfiltration from user machines. Browser-based IDEs are important to self-service platforms that enhance developer productivity, ensuring consistency and governance.

Drivers

Gartner predicts that, by 2026, 60% of cloud workloads will be built and deployed using browser-based IDEs. Five factors are driving their increased adoption:

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

2. Faster time to market requires consistency in development workflows and reduced toil to bootstrap environments. Environment setup issues can impede productivity and hurt the onboarding experience.

3. Cloud-native (e.g., Kubernetes) deployments require new tooling that either isn't available or is inconvenient to set up on-premises. Likewise, low-code development tools are web-first — 89% of low-code development vendors support a web-based IDE, compared with 24% for desktop-based IDEs.

4. The ability to centrally manage, govern and secure development environments becomes especially important with the increasing threat of software supply chain attacks. In addition, browser-based IDEs make it easier to support and secure bring your own PC (BYOPC) use cases.

5. Automating DevOps workflows introduces more plug-ins, extensions and API integrations, which make it cumbersome to manage on local machines.

Browser-based IDE providers are innovating rapidly. Representative providers include Amazon Web Services Cloud9, Codeanywhere, GitHub Codespaces, Gitpod, Red Hat CodeReady Workspaces and Replit.

Obstacles

Browser-based IDEs incur costs in addition to what an organization may already be paying for DevOps tooling. The cost can be prohibitive for teams that rely only on open-source tools for application development and delivery needs on local machines. The pay-per-use subscription costs add up when you have a large team, and developers may leave the IDE open with misconfigured settings for autohibernation or timeout.

Connectivity presents another obstacle. Poor or inconsistent internet speeds adversely affect developer productivity. Developers cannot write, debug and test code without a stable internet connection.

Security and compliance policies can prevent the use of cloud for development needs in some organizations. This can rule out the use of browser-based IDEs. Browser-based IDEs often depend on specific source control repositories or a specific Kubernetes implementation. This requires organizations to change their development practices before adopting browser-based IDEs.

User Recommendations

Software engineering leaders leading product and platform engineering teams should:

- Pilot the use of browser-based IDEs when their teams are developing with cloud-hosted source repositories. Even when developers prefer local environments on their machines, browser-based IDEs can provide a baseline as a "reference" environment.
- Ensure that browser-based IDEs are part of an overall developer self-service platform with central governance. The benefit of centralized governance and developer agility can be a win-win for platform and product teams.
- Use browser-based IDEs as one of the "quick wins" to improve the onboarding experience for new developers and reduce ramp-up time.
- Work with security leaders and enforce strong authentication and authorization policies to mitigate security risks. Browser-based IDEs present an additional, high-value attack vector, as they become the pipeline through which intellectual property flows.

Sample Vendors

Amazon Web Services; Codeanywhere; GitHub; Gitpod; Red Hat; Replit

Gartner Recommended Reading

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

Expert Insight Video: Improve Remote Worker Productivity by Measuring Performance Fairly

Accelerating Agile and DevOps Adoption Post-COVID-19

The Future of DevOps Toolchains Will Involve Maximizing Flow in IT Value Streams

GitOps

Analysis By: Paul Delory, Arun Chandrasekaran

Benefit Rating: High

Market Penetration: Less than 1% of target audience

Maturity: Embryonic

Definition:

GitOps is a technique for operating a cloud-native application using only declarative constructs stored in Git. It is the latest name for extending CI/CD to software deployment and infrastructure management. Beyond that, GitOps remains ill-defined and the subject of many debates. The CNCF's GitOps Working Group is formulating what it hopes will become the industry standard. The ideas behind GitOps are not new, but emerging tools and practices now promise to make this longtime dream a reality.

Why This Is Important

GitOps would be transformative for the I&O organization. GitOps workflows convert a Git pull request into a verified container image ready for production. Infrastructure is code. All changes flow through Git, where they are version-controlled, immutable, auditable and reversible. Developers interact only with Git, using abstract, declarative logic. I&O uses it to deploy infrastructure. It extends a common control plane across K8s clusters, which is increasingly important as clusters proliferate.

Business Impact

- **Agility and speed:** GitOps can be used to introduce version control, automation, collaboration and compliance. Artifacts are consistent and reusable. All of this translates directly to business agility and faster time to market.
- **Transparency and visibility:** GitOps artifacts are centralized and version-controlled, making them easy to verify and audit.
- **Resilience:** By implementing and centralizing infrastructure as code, GitOps enhances availability and resilience of services.

Drivers

- **Kubernetes adoption and maturity:** GitOps must be underpinned by an ecosystem of technologies, including tools for automation, infrastructure as code, CI/CD, observability and compliance. Kubernetes has emerged as a universal common substrate for cloud-native applications. This provides a ready-made foundation for GitOps. As Kubernetes adoption grows within the enterprise, so, too, can GitOps.
- **Need for increased speed and agility:** Speed and agility of software delivery is a critical metric that CIOs care about. As a result, IT organizations are pursuing tighter coupling of I&O and development teams to drive shorter development cycles, faster delivery and increased deployment frequency; 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 with process transparency and workflow commonality 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 better 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. The Working Group's draft architecture expressly requires Kubernetes and implicitly assumes the presence of a host of other technologies built on top of K8s. GitOps is necessarily limited in scope.
- **Lack of consensus:** For now, GitOps means different things to different people. This situation is likely to persist unless the GitOps Working Group succeeds in promulgating a definition that gains industrywide acceptance.
- **Cultural change:** GitOps is 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. Technology is not magic.
- **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

- **Use GitOps for cloud-native workloads:** Your initial target for GitOps must be a containerized, cloud-native application that is already using both Kubernetes and a continuous delivery platform such as Flux.
- **Embed security into GitOps workflows:** GitOps practices can and should embed security into the workflow. Security teams need to "shift left" so that the organization can build holistic CI/CD pipelines that impact software delivery and infrastructure configuration, with security embedded in every layer of that workflow.
- **Be wary of vendors trying to sell you GitOps:** GitOps isn't a product you can buy; it is a workflow and a mindset shift that needs to be part of your overall DevOps culture.
- **Use GitOps to sell automation initiatives:** I&O leaders can use the GitOps buzzword to get buy-in for adopting general automation best practices that the organization should already be doing. In the absence of a standard, you can define what GitOps means to your organization, then deploy it iteratively.

Sample Vendors

CloudBees; GitLab; Harness; Red Hat; Weaveworks

DesignOps

Analysis By: Brent Stewart

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

DesignOps is a set of operational practices that enables design team management and product-level delivery of design assets. The team management side focuses on strategic alignment to the business, operations for the central design function and career development. The product delivery side combines UX, product management and technology operations to enable efficient and DevOps-compatible plans, estimates and processes that increase quality, enable collaboration and feed 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 as digital product companies (for example, Airbnb, Adobe and InVision) and agencies alike discover the tremendous value of a proven operational approach for UX team management and design delivery on product teams.

Business Impact

The growth of DesignOps is due, primarily, to the value it creates during the delivery of design assets. Here, DesignOps does not alter the core skills and activities of a UX team; rather, it reorganizes them in a way that supports ongoing feature enhancement and idea generation without interrupting the continuous workflow of development teams. DesignOps represents the first widespread implementation of operational methods and techniques created not only for designers, but also for developers.

Drivers

Modeled to be compatible with DevOps and agile practices, DesignOps structures and organizes design work to enable early and frequent feedback via collaboration between the user, the designer and the developer, as well as ongoing, iterative delivery of assets and design decisions to the development team. This allows product teams to run parallel tracks of work (dual-track agile) in which UX teams employ “continuous discovery” to understand the user, engage in research, explore various design directions, test possible solutions and document outcomes. It also allows them to progressively support early development activities such as tech design and story creation.

There are three key drivers behind DesignOps:

- **Innovation:** When coupled with DevOps, DesignOps leads to more innovative solutions. As a practice, DesignOps employs dual-track agile that 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 sooner than before, and with much 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

To a large extent, the growth of DesignOps is inhibited by key gaps in planning, estimation and tracking knowledge:

- 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.

- Popular enterprise agile planning (EAP) tools are not designed with UX practitioners, activities and deliverables in mind (though this is changing).

User Recommendations

Software engineering leaders should:

- Educate themselves about the practice of DesignOps
- Train their UX teams in the basics of agile
- Pilot the approach with a high-performing, multidisciplinary feature 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 persons to the role of design manager — essentially, a UX-focused product manager.

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

Gartner Recommended Reading

DesignOps: Organize, Collaborate and Innovate Product UX at Speed

Technology Insight for Digital Product Design Platforms

3 Key Practices to Enable Your Multiexperience Development Strategy

Software Engineering Technologies Primer for 2021

Build Links Between Customer Experience, Multiexperience, User Experience and Employee Experience

Strategic Roadmap for Becoming a Digital Product Delivery Organization

Innersourcing

Analysis By: Arun Chandrasekaran, Mark Driver, Anne Thomas, Arun Batchu, Fintan Ryan, Mark O'Neill

Benefit Rating: Moderate

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

Innersourcing is the practice of bringing the open-source software principles of collaboration and openness into the organization for any form of software development.

Why This Is Important

Gartner has seen client inquiries on innersourcing increase 20% year over year. This coincides with a renewed interest in open source, in part driven by the COVID-19 pandemic. Organizations such as Comcast have been successful in adopting an innersourcing model (see the Comcast case study in A CTO's Guide to Top Practices for Open-Source Software). This has driven the interest of other organizations to similarly apply open-source practices internally in their organizations.

Business Impact

- Innersourcing encourages strategic thinking of software usage rather than trying to solve tactical problems.
- Organizations can use innersourcing to more effectively align with external open-source communities and foundations to tap into their knowledge.
- Embracing open-source principles can encourage employee retention through recognition of contributions.
- Applications created in an innersourcing model are suitable for being released as publicly available open-source projects.

Drivers

- Automated software deployment is a driver for innersourcing, meaning that DevOps adoption is a prerequisite for an innersouring program.
- The need for communication and collaboration between disparate software development teams within organizations drives the need for innersourcing.
- Software developers are now overwhelmingly familiar with the practices of open source, including through interacting with code repositories, using well-known open-source tools and even, in some cases, through contributing to open-source projects. This creates the foundation for the effective use of open-source principles inside organizations.
- Code quality can be improved through the usage of automated quality and consistency checking, which is often part of the technical infrastructure of an innersourcing effort, often via the work of an open-source program office (OSPO).
- Increased cross-team collaboration and reuse of functionality, as code libraries or via APIs, are also key drivers of innersourcing initiatives.

Obstacles

- Creating an effective innersourcing program requires sponsorship from senior leadership.
- Governance of innersouring is a challenge, but can be met through establishing an open-source program office and communities of practice.
- Many organizations do not yet have self-service access to deploy code, or access APIs or shared libraries.

User Recommendations

- Emphasize adequate documentation, cross-team collaboration and code contribution to other internal teams' repositories.
- Create smaller and flatter product teams with an emphasis on egalitarian decision making and constant communication on product direction.
- Establish and communicate best practices on coding standards, preferred tooling and common repositories, and code release process.
- Enable self-service access to infrastructure as a service (IaaS), and automate the build, test and release pipelines.
- Emphasize documentation and a simplified on-boarding experience for newcomers.
- Conduct hackathons to stimulate the creative process, and listen to developer feedback to remove bureaucracy.

Gartner Recommended Reading

A CTO's Guide to Top Practices for Open-Source Software

Building a Platform for Product Team Productivity (adidas)

Create API Portals That Drive API Adoption Among Internal and External Developer Communities

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 for continuous application performance improvement that involves practices, techniques and activities during each DevOps phase to achieve the performance quality goals of the business. It focuses on the architecture, design and implementation choices that will affect application performance and encompasses the practices that help mitigate performance 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 delight customers by exceeding their performance expectations. It provides the framework for application performance excellence that drives value and supports the realization of business outcomes for customers.

Drivers

- Raised end-user expectations for application quality, specifically nonfunctional 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 hyperdynamic microservices-based architectures, multicloud deployments, and 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 AWS, Google or 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

- **Focusing only on tools and technology:** Performance engineering requires a change in organizational culture. Tools enable quality but won't solve problems on their own.
- **Organizational inertia:** Departmental silos, traditional top-down management structures and a lack of experience with managing quality continuously can impede adoption.
- **Lack of clear goals:** Successful performance engineering requires clear goals that are aligned with the priorities of the business.
- **Internal pushback:** 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.
- **Limitation to performance testing only:** 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 for nonfunctional characteristics such as performance efficiency. ISO/IEC 25010 provides a template for understanding quality characteristics and includes performance efficiency as one of the top-level nonfunctional domains.
- 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 required skills needed for performance engineering by identifying the required roles, technologies and practices.
- Establish relevant 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 by leveraging performance information from production and real users.

Sample Vendors

AppDynamics; Broadcom; Dynatrace; Eggplant; Micro Focus; Tricentis

Gartner Recommended Reading

Adopt a Performance Engineering Approach for DevOps

Innovation Insight for Continuous Quality

Improve Software Quality by Building Digital Immunity

Innovation Insight for Autonomous Testing

DevOps VSMPs

Analysis By: Hassan Ennaciri, Akis Sklavounakis

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Early mainstream

Definition:

DevOps value stream management platforms (VSMPs) enable organizations to manage their software delivery process as a value stream to maximize the delivery of customer value. They provide visibility and traceability to every process in software delivery — from ideation through development to release and production, and extending to documenting feedback from customers. DevOps VSMPs continuously measure flow, surface constraints and non-value-added work, to improve customer value delivery.

Why This Is Important

As organizations scale their agile and DevOps practices, higher-level metrics that assess performance and efficiency of their product delivery is essential. DevOps 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 to correct course as needed. The visualization capabilities of DevOps VSMPs help product teams analyze customer value metrics against the cost required to deliver that value.

Business Impact

DevOps VSMPs help organizations bridge the gap between business and IT by enabling stakeholders to align their priorities to focus on delivering customer value. DevOps VSMPs can provide CxOs with strategic views of product delivery health and pipelines, allowing them to expedite data-driven decisions about future investments in products. 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

- Scaling agile and DevOps initiatives.
- Need for visibility into business, development and operational metrics.
- Optimization of delivery flow by mapping end-to-end processes involved in software delivery to reduce waste and bottlenecks due to cross-team dependencies.
- Need to improve quality and velocity of product deployments.
- Visibility to security and compliance status of products.
- Need for orchestration capabilities to have better traceability.

Obstacles

- DevOps VSMPs do not include native continuous integration/continuous delivery (CI/CD) capabilities; execution of the delivery pipeline requires use of a custom toolchain or a value stream delivery platform (VSDP).
- VSMPs may lack native plugins for some existing tools and that will require custom integration.
- Rationalization of acquiring another tool with more dashboards acts as an obstacle.
- VSMPs are still evolving and not all vendors have all the critical capabilities.

User Recommendations

- Ensure all stakeholders in the value streams are involved in the selection of a VSMP: business leaders, owners, application leaders and I&O leaders.
- Examine your existing EAP tools or VSDPs as some vendors are expanding their capabilities to include VSMP functionality.
- Pilot the VSMP with a product that has mature DevOps practices to best evaluate what insight can be generated.

Sample Vendors

CloudBees; Digital.ai; Opsera; Plandek; Plutora; Tasktop

Gartner Recommended Reading

The Future of DevOps Toolchains Will Involve Maximizing Flow in IT Value Streams

Agile and DevOps Primer for 2020

Flattening the Application Organization — Everyone Must Be Part of the Agile Value Stream

Predicts 2021: Value Streams Will Define the Future of DevOps

Infographic: Top 10 Technology Trends Impacting DevOps

Chaos Engineering

Analysis By: Jim Scheibmeir, Dennis Smith

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Chaos engineering is the use of experimental and potentially destructive failure testing or fault injection testing to uncover vulnerabilities and weaknesses within a complex system. It is systematically planned, documented, executed and analyzed as an attack plan to test components and whole systems both before and after implementation. Chaos engineering is often utilized by site reliability engineering teams to proactively prove and improve resilience during fault conditions.

Why This Is Important

Many organization's stake success on test plans that overemphasize software functionality and underemphasize the importance of validating the system's reliability. Chaos engineering (CE) moves the focus of testing a system from the "happy path" — instead, testing how the system can gracefully degrade or even continue to be useful while under various levels of impact. CE can also help identify where product documentation is less than sufficient or understanding of a system is lacking or siloed.

Business Impact

With chaos engineering, we minimize time to recovery and change failure rate, while maximizing uptime and availability. Addressing these elements helps improve customer experience, customer satisfaction, customer retention and new customer acquisition.

Drivers

- Gartner's 2020 Achieving Business Agility with Automation, Continuous Quality and DevOps Survey found that 18% of respondents were currently using or planning to use chaos engineering to improve software quality.
- Complexity in systems and increasing customer expectations are the two largest drivers of this practice and the associated tools. As systems become more rich in features, they also become more complex in their composition, and more critical to business success.
- Overall, chaos engineering helps organizations create more reliable systems and prove that systems are reliable.

Obstacles

- The first obstacle to chaos engineering is perception. Within many organizations, the predominant view is that the practice is random, done first in production, and due to these leads to more risk than less.
- Another obstacle to chaos engineering is organizational culture and attitude toward quality and testing. When quality and testing are only viewed as overhead, then there will be a focus on feature development over application reliability.
- Another common obstacle is simply gaining the time and budget to invest in learning the practice and associated technologies. Organizations must reach minimum levels of expertise where value is then returned.

User Recommendations

- Utilize a test-first approach by practicing chaos engineering in preproduction environments. Then move findings into production environments.
- Incorporate chaos engineering into your system development or testing process.
- Build-out incident response protocols and procedures, as well as monitoring, alerting, and observability capabilities in tandem with advancement of the chaos engineering 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 chaos engineering in production to facilitate learning and improvement at scale as the practice matures. However, be warned that Gartner believes that there are still very few organizations purposely using chaos engineering 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

Alibaba Cloud; Amazon Web Services; ChaosIQ; Gremlin; steadybit; Verica

Gartner Recommended Reading

Improve Software Quality by Building Digital Immunity

Innovation Insight for Chaos Engineering

How to Safely Begin Chaos Engineering to Improve Reliability

DevOps Teams Must Use Site Reliability Engineering to Maximize Customer Value

DevOps Test Data Management

Analysis By: Dale Gardner

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 data to evaluate the performance, functionality and security of applications. It typically includes copying production data, anonymization or masking and virtualization. In some use cases, specialized techniques, such as synthetic data generation, are appropriate. Given potential compliance and privacy issues, the process more frequently involves members of application and data security teams.

Why This Is Important

Test data management is inconsistently adopted across organizations, with many teams 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, technology. However, the reality is the combination of the increased pace of development from DevOps and a growing number of privacy mandates and constraints have strained traditional approaches — yielding new approaches and technology.
- More traditional test data management has been inconsistently adopted, with many organizations either simply using copies of production data or generating "dummy data" (distinct from emerging synthetic data generation techniques). 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 a growing number of 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. The 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 themselves are technologically vastly more complex), applications, and APIs to function, testing has become more complex. Such complexity demands 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 staffing.
- 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 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 — application development and test teams (to understand consumption patterns and needs), data management, privacy and security teams, compliance teams, other security teams, and legal counsel — as appropriate.
- Document existing test data management practices so tools and processes can be evaluated against data protection mandates.
- Coordinate 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.
- For DevOps use cases where frequent updates to test data are required, data virtualization may also be useful. Virtualization can speed the process of providing copies of safe data.

Sample Vendors

Actifio; BMC Compuware; Broadcom (CA Technologies); Delphix; Hazy; Informatica; MENTIS; Micro Focus; Solix Technologies

Gartner Recommended Reading

Market Guide for Data Masking

Elevating Test Data Management for DevOps

API Security Testing

Analysis By: Dale Gardner

Benefit Rating: High

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

API security testing is a specialized type of application security testing (AST) aimed at identifying vulnerabilities in application programming interfaces (APIs). Checks should include both traditional application vulnerabilities (e.g., injection attacks) and API-specific issues (e.g., broken-object-level authorization). Availability of an API definition (e.g., OpenAPI) is often, but not always, a prerequisite for effective testing; discovery helps ensure that unknown APIs are identified.

Why This Is Important

Attacks on applications are shifting to APIs, and their pace is increasing. By 2022, API abuses will be the most-frequent attack vector, and will result in data breaches. DevSecOps teams are focusing attention on the need for improved API testing in development, looking to a mix of traditional tools — e.g., static AST (SAST) and dynamic AST (DAST) — and emerging solutions focused on the requirements of API testing to identify the optimal approach to API testing.

Business Impact

APIs are a foundational element of many organizations' digital transformation efforts. Hence, securing APIs from attack and misuse is a growing concern for many security and risk management (SRM) professionals. API-specific testing, pre- and post-deployment, builds a solid foundation for an overall API security strategy. API discovery is needed, because many organizations struggle to maintain an inventory of APIs and need help locating them and ensuring they are tested and managed.

Drivers

- APIs enable information flow and support transactions between processes, programs and systems, so APIs have become common in application architectures, especially in support of digital transformation efforts. However, that growth has brought increased attention from attackers, and APIs have become the primary attack surface for many systems. Gartner has long noted that attacks and abuse against APIs are a common attack vector.
- These attacks have resulted in an endless stream of data breaches and other security incidents, yielding significant damage to organizations and individuals. As a consequence, DevSecOps teams — along with the business leaders whose applications are supported by APIs — are increasingly interested in API testing and security.
- Because the creation, development and deployment of APIs may be loosely managed, security teams often have to contend with unknown APIs, which exist outside normal processes and controls. Such APIs may be especially deficient in secure design and testing, posing an even greater risk. Hence, the discovery of APIs deployed to production is another important need.
- Given their increasing importance in application architecture and their sensitivity, identifying and remediating exploitable vulnerabilities in APIs has become an obvious goal. API security testing helps avert the tangible and intangible costs associated with breaches and other types of security incidents.
- Traditional AST tools — SAST, DAST and interactive AST(IAST) — were not originally designed to test for vulnerabilities associated with typical attacks against APIs, or for newer types of APIs (e.g., GraphQL). Although vendors have added support, these gaps have created an opportunity for specialist API security vendors, focused on testing, discovery of running APIs and protection from threats — or some combination of the three.

Obstacles

- Existing tools must be extended to address API-related vulnerabilities, and incorporate new testing approaches. APIs are susceptible to most, if not all, traditional attacks against applications; however, newer attack types have also emerged. For example, improper authorization checks around object identifiers (such as a user identifier) have been cited in a number of breaches.
- APIs in a number of forms are using various protocols. Simple Object Access Protocol (SOAP) remains in widespread use, although has been supplanted by Representational State Transfer (REST) APIs. GraphQL (a combination of query language and runtime) is now challenging REST. This requires additional support from tool vendors for effective tests.
- There is confusion in the marketplace, with various types of companies (traditional AST vendors, along with API testing and protection vendors, etc.) offering products. This complicates evaluation and selection efforts.

User Recommendations

- Do not invest in tools without a solid inventory of APIs — unless the tool is specifically intended to help address shortcomings in this area.
- Assess the overall role APIs play in your application portfolio —their criticality to the organization, their security and business risk, and the technical requirements they pose — to begin evaluation and selection efforts.
- Examine the testing capabilities provided by existing tools in your application security portfolio. Many have added support for APIs, although actual tests may still focus primarily on traditional application vulnerabilities.
- Where gaps are found, evaluate newer alternatives. They may also offer additional options, including audit of design-stage API specifications, as well as vulnerability scans and threat protection.
- API security tool capabilities — including discovery, testing and threat protection — overlap, so evaluate the ability of a given tool to address multiple requirements.

Sample Vendors

42Crunch; APIsec; Checkmarx; Contrast Security; Data Theorem; Imvision; NTT Application Security; Salt; Synopsys; Veracode

Gartner Recommended Reading

API Security: What You Need to Do to Protect Your APIs

Solution Path for Forming an API Security Strategy

Magic Quadrant for Application Security Testing

Critical Capabilities for Application Security 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 achieve the goal of making “quality” the responsibility of all. It synchronizes quality assurance and testing with DevOps processes and encompasses the practices that help mitigate risks before progressing to subsequent stages of the software development life cycle.

Why This Is Important

Many DevOps organizations are practicing continuous integration and continuous deployment, yet a continuous approach to quality is often missing. The ability to consistently deliver business value with high quality has become critical for organizations seeking to mature their DevOps processes. Continuous quality encourages a holistic and proactive approach with functional and nonfunctional requirements driving the design, development and delivery of products.

Business Impact

The adoption of a continuous quality strategy significantly improves an organization's ability to serve and delight its customers. Continuous quality enables solutions to be delivered at a greater release rate and with fewer defects than traditional quality control practices. It provides the framework for operational excellence that drives value, supports the realization of business outcomes for customers and streamlines operational processes.

Drivers

- Raised end-user expectations for application quality, which require a shift to a more holistic view of what constitutes superior quality that delights users.
- The pressure to innovate rapidly in order to launch differentiated products in the market quickly without compromising on quality.
- The ability to consistently deliver business value with consistently high quality to mature DevOps processes.
- The need to ensure that teams are equipped to create a superior user experience, build features that fit the market's timing, and enable the characteristics of an application that deliver value faster than they create technical debt.

Obstacles

- Lack of clear goals: Successful continuous quality requires clear goals that are 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 restrictive and requires consensus on usage across all team members.
- Loss of productivity: Changing organizational culture and engaging in new practices require significant investment and time. This will impact current timelines and can cause a decrease in productivity prior to reaching steady productivity.
- Limitation 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 development.
- Focusing only on tools: Continuous quality requires a change in organizational culture. Tools are enablers of quality but tools on their own won't solve problems.

User Recommendations

- Move away from the traditional application- or project-centric model of quality to a holistic quality approach by adopting an ecosystem-centric view of quality and a focus on business outcomes.
- Accelerate product delivery by championing a continuous quality mindset and involving stakeholders across the organization.
- Allocate ownership and appoint staff with the required 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.
- Task teams with developing continuous quality practices before choosing tools.

Gartner Recommended Reading

Innovation Insight for Continuous Quality

Adopt a Performance Engineering Approach for DevOps

Improve Software Quality by Building Digital Immunity

Maverick* Research: Software Testing and the Illusion of Exterminating Bugs

Innovation Insight for Autonomous Testing

Graph APIs

Analysis By: Mark O'Neill

Benefit Rating: Moderate

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

A graph API models its data as an object graph (a set of objects and relationships) described using nodes and edges. It enables users to query and manipulate the data by following and filtering on the relationships, in addition to the objects. This contrasts with a traditional REST API, which models data as resources without relationships. Many graph APIs, such as the Facebook Graph, provide an abstraction over multiple distinct data sources and APIs. GraphQL may be used to define graph APIs.

Why This Is Important

Graph APIs are gaining in importance because developers benefit from flexible self-service access to data. User and consumer needs drive API usage in multiple unpredictable ways, so dedicated APIs become harder to define, compared to graph APIs. Finally, the value of data and the need to exploit information held in data relationships drives the importance of graph APIs.

Business Impact

For API providers, graph APIs unlock business value in data, and in the relationships between data in particular. Graph APIs also provide more efficient use of resources, compared to the overhead of individual REST API calls. For API consumers, graph APIs can simplify integration because developers do not need to consume multiple different APIs from the same API provider in order to build an application or an integration.

Drivers

- The need for developer self-service access to data is a key driver for graph APIs.
- API marketplaces such as RapidAPI have begun to include graph APIs, which offers another point for the publication and discovery of graph APIs.
- Vendors including Microsoft, with Microsoft Graph API accessing multiple products including Microsoft Azure AD and Exchange Online, and SAP, with SAP Graph API accessing multiple SAP applications, including SuccessFactors and S4/HANA, now provide graph APIs across multiple products.

Obstacles

- Graph APIs place significantly more burden on teams supporting APIs and require investment in new design, security, monitoring and governance skills. In addition, vendor support for GraphQL is still not widespread. Security, governance and performance optimization practices are still being developed.
- Privacy concerns have arisen for graph APIs, in particular related to Facebook's original Graph API and its usage by Cambridge Analytica, leading Facebook to address these concerns in later versions.
- Performance is a consideration due to the amount of processing required to traverse a graph.
- Design skills for graph APIs are currently behind the skills for REST APIs.

User Recommendations

- Evaluate Graph APIs when they are available from their platform providers, and adopt them to provide flexible self-service access to data, including for applications used by their users such as dashboards and mobile apps.
- Replace dedicated "experience APIs" with graph APIs that tailor API experiences to different API consumers. This is because graph APIs offer the consumer more flexibility and control over how related data is accessed than traditional APIs.
- Adopt a community of practice approach to spread both Graph API and GraphQL skills in your organization as for both API consumers and API providers, graph APIs introduce new skills.

Sample Vendors

Apollo; Facebook; Google; Hasura; IBM; Microsoft; RapidAPI; SAP; StepZen; Tyk; WSO2

Gartner Recommended Reading

Assessing GraphQL for Modern API Delivery

Ensure Your API Management Solution Supports Modern API Trends Such as Microservices and Multicloud

DevOps VSDPs

Analysis By: Manjunath Bhat

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Early mainstream

Definition:

DevOps value stream delivery platforms (VSDPs) provide fully integrated capabilities that enable continuous delivery of software. These capabilities include planning, version control, continuous integration, test automation, release orchestration, continuous deployment (and rollback), monitoring, security testing and analyzing value stream metrics. DevOps VSDPs integrate with infrastructure and compliance automation tools to automate infrastructure deployment and policy enforcement.

Why This Is Important

Value stream delivery platforms enable organizations to simplify building and managing DevOps pipelines. They reduce complexity involved in orchestrating, integrating and governing pipeline activities. In addition, the prebuilt integration between different components of the platform leads to improved visibility, traceability and observability into the complete application development value stream. The benefits extend beyond IT.

Business Impact

VSDPs constitute the pipeline for continuous delivery of digital value. The seamless integration and shared visibility between development and operations workflows helps bridge the silos that exist between development and operations teams. Using a common platform for development, security and operations accelerates agile transformation and helps organizations adopt a product and platform team operating model. VSDPs are a foundational component of self-service platforms for development teams.

Drivers

- Continued cloud adoption and cloud-native architectures.
- Agile and DevOps ways of working with automation and collaboration being the focal points.
- Need for improved developer experience, satisfaction and productivity.
- Need for deeper collaboration and shared visibility across all IT teams (dev, sec and ops).
- Emergence of digital product and platform engineering teams to drive digital transformation initiatives.
- Competitive pressure that demands faster time to market for digital products and services.
- Proliferation of tools and the ensuing complexity of integrating those tools as well as the lack of skills to do so at scale.
- Industry traction around value stream management — VSDPs will increasingly bundle capabilities to analyze value stream metrics and extend the usability of the platforms to business stakeholders.
- Rise of vendors providing a managed open-source software (OSS)-based VSDP. Customers get the best of innovation from the OSS community and the guaranteed SLAs from a technology provider.
- Increased maturity of “as code” approaches to automate security and compliance (via policy as code) and infrastructure (via infrastructure as code).

Obstacles

- Organizations that want to unlock the full benefits of VSDPs must be willing to replace an existing toolchain — either completely or in-part. However, this is an impediment for teams that are resistant to change or view the change as a disruption to their ways of working.
- Organizations accrue technical and skills debt over time due to legacy automation workflows. That hinders teams from adopting new tools.
- The potential for vendor lock-in presents a constraint as well. Dependency on a single provider for a majority of their software development needs increases concentration risk and lowers bargaining leverage. Gartner cautions VSDP buyers that none of the vendors currently provide the full set of application delivery capabilities.
- VSDPs also fall short of providing the required value stream analytics that organizations can use to improve “flow” in the software delivery life cycle.

User Recommendations

- Scale DevOps initiatives by providing VSDPs as a self-service platform to reduce overhead, lower complexity, and ensure consistent and templatized workflows across multiple teams.
- Improve the flow of work by streamlining the software delivery life cycle with VSDPs that provide enhanced visibility, traceability, auditability and observability across the DevOps pipeline.
- Support remote development teams by adopting cloud-based VSDPs and using their integration and collaboration capabilities for code reviews, code sharing and issue tracking.
- Adopt an “as code” approach to managing IT workflows to improve operational efficiency and consistency — pipelines as code, infrastructure as code and policy as code.

Sample Vendors

Atlassian; Calculi; CloudBees; CodeNOW; Copado; Digital.ai; GitHub; GitLab; Harness; JFrog

Gartner Recommended Reading

Market Guide for DevOps Value Stream Delivery Platforms

Market Guide for DevOps Value Stream Management Platforms

The Future of DevOps Toolchains Will Involve Maximizing Flow in IT Value Streams

Why DevOps Success Requires Platform Teams

Platform Teams and AIOps Will Redefine DevOps Approaches by 2025

Quality Engineering

Analysis By: Susanne Matson, Jaideep Thyagarajan

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Quality engineering is the discipline of injecting quality across all stages of the software development life cycle (SDLC). It is the application of lessons learned and IP generated through quality assurance to engineer better business, IT or OT processes, products, solutions, services and applications from the outset of a development project.

Why This Is Important

Quality engineering (QE) is more widely practiced in organizations leveraging agile practices and DevOps methodologies. By enabling continuous quality, QE supports organizations to advance their continuous integration, continuous deployment and continuous delivery initiatives and programs. QE is supporting the continued improvement work for better agility and integration.

QE is essential to ensure differentiated user experiences in today's competitive digital markets.

Business Impact

- QE service offerings can help improve differentiation in developing or improving business, IT or OT processes, products, solutions or services.
- Benefits to be gained in development projects include: time to market; improved overall quality; decreased costs due to automation and elimination of manual labor; and improved consumer experience.

Drivers

- Organizations look to apply a QE approach to help them gain efficiency and business benefits by not separating the ideation and requirements phases from design, build and test. These are often owned by different entities — business versus IT.
- As organizations focus on what needs they have after the accelerated digital transformation following the pandemic, more users identify QE as a domain to look into. The intent is to help them reach quality needs at an increased pace.
- QE as a competence is focused on applying lessons learned on what works within a domain, for a specific purpose and for a specific user base. Applying analytics and intelligent automation during the development life cycle as part of QA services will enable organizations to engineer better processes, products, solutions or services without having to go through multiple iterations.
- Organizations use quality engineering to implement quality across the entire SDLC through standardization and more effective use of tooling.

Obstacles

- Although QE is commonly embedded as an integral component in business or technology consulting, challenges within the application services domain remain.
- The skill level needed for a quality engineer is more complex than traditional testing work.
- Cultural issues such as lack of collaboration between teams such as testing, development and operations teams can hinder QE.
- Finding the right QE vendor partner can be a challenge.

User Recommendations

- Formalize the inclusion of QE in the development process, as the default mechanism to feed relevant product, solution or service artifacts into the ideation phase.
- Feed best practices into the functional and technical design, and do not wait for QA to identify flaws during development or, even worse, in production.
- Leverage service providers' ability to apply an integral approach. These different capabilities can then be combined across development, agile, DevOps, advisory and industry teams.
- Utilize automation, predictive analytics and machine learning to support QE as they can help improve focus, effectiveness and insight into the testing process.
- Apply an integral approach from business, IT and OT perspectives, and actively focus on identifying, consolidating and maintaining QE artifacts as part of any development project.

Sample Vendors

Accenture; Capgemini; Cognizant; Deloitte; IBM; Infosys; TCS; Wipro

Gartner Recommended Reading

Application Testing Services must Evolve to Meet Client Demands in Agile and DevOps

Magic Quadrant for Application Testing Services, Worldwide

Critical Capabilities for Application Testing Services, Worldwide

Optimize Application Testing Quality and Speed With Embedded Intelligent Automation Services

At the Peak

Microapps

Analysis By: Jason Wong

Benefit Rating: Moderate

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

A microapp is a discrete, yet reusable and portable, app function, process or workflow that operates within the context of a larger app or application — and possibly across multiple apps or applications. The microapp must be tightly scoped and is composed of user interface (UI), logic and data components typically bound to back-end microservices through a mediated API layer. Microapps can be used as part of a micro-frontend architecture to compose a fit-for-purpose user experience (UX).

Why This Is Important

The term “microapp” is not new. But the architectural concepts of microapps as part of a mesh app and service architecture (MASA), and applied to multiexperience development (spanning web, mobile and conversational apps), are important to designing for composable business and achieving composable experiences. Microapps help scale front-end development because they are built and run as self-contained activities that may rely on common back-end services.

Business Impact

- Microapps deliver composable experiences, enabling more dynamic and contextual user engagement.
- They improve agility and reuse, with an ability to develop functionality independently by different developers.
- They promote consistent UX, especially across a multiexperience user journey.

Drivers

- Microapps have gained traction as part of mobile app development, but have yet to be more broadly embraced for multiexperience. However, there has been a steady increase in the use of micro-frontend application architecture that aligns with the use of microapps, such as creating a library of React-based modules or components. Microapps also support event-based scenarios to present interactions based on context, which aligns with event-driven architecture trends.
- In recent years, the total number of vendors using the term “microapp” to describe capabilities of their offerings has steadily increased. Enterprise software providers, such as SAP and Citrix, are also enabling and delivering microapps for greater configuration of a multiexperience UX.
- Some providers have started to create “super apps,” which are the runtime mechanism for distributing microapps from their ecosystem and developers. The WeChat and Alipay apps in China are examples of super apps that allow for third parties to create and deploy miniprograms within their apps. Enterprise collaboration and messaging platforms, such as Microsoft Teams and Slack, are taking cues from these consumer super apps to also enable third parties to create and distribute microapps within their main mobile or web apps.
- On the enterprise side, Tata Digital announced the creation of its own super app to bring all the Tata Group consumer businesses onto a single platform. This is an evolution of the apps economy, driven by digital business ecosystems. Microapps align with agile development practices and support the shift to smaller units of development work that can be delivered continuously. Organizations must align to agile and DevOps practices to effectively deliver microapps, and shift to cloud-centric development to streamline self-service development and DevOps processes. Security and governance is a concern when multiple microapps, from different providers, run in a larger application context.

Obstacles

- One main challenge is in designing for microapps. This requires understanding the user journey and breaking down interactions into microjourneys. This mature UX design approach allows for microapps to be used across mobile apps, as well as in web apps and supporting conversational apps, such as a card UI within a chatbot interaction.

- The development effort requires a high level of skills and coordination. Some organizations rely on specific client-side frameworks or proprietary platforms, such as low-code application platforms (LCAPs) or multiexperience development platforms (MXDPs), to implement an architecture supporting microapps. Using platforms and frameworks could lead to vendor lock-in risks; therefore, development teams must use appropriate isolation techniques to separate their code.

User Recommendations

Application and software engineering leaders should encourage their teams to:

- Address the need to support ever-increasing digital experiences by using microapps to facilitate multiexperience development running on the MASA.
- Identify suitable development frameworks or technologies for microapp enablement and orchestration across your target touchpoints (e.g., web, mobile app, chatbots).
- Avoid functional conflicts by managing governance of the microapp runtime container's capabilities (such as permissions, user consent and location service).
- Learn and strengthen modern application development skills that improve architecture, development and delivery.

Gartner Recommended Reading

3 Key Practices to Enable Your Multiexperience Development Strategy

Essential Skills for Modern Application Development

One Versus Many — When to Consolidate Your Enterprise's Apps

Autonomous Testing

Analysis By: Joachim Herschmann, Jim Scheibmeir

Benefit Rating: High

Market Penetration: 1% to 5% of target audience

Maturity: Adolescent

Definition:

Autonomous testing comprises AI- and ML-based technologies and practices to make software testing activities 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 which includes all phases of the testing life cycle. As such, autonomous testing is a driver for a holistic approach to automating the broader set of quality and testing activities related to requirements quality, design quality, code quality, release quality, and operational resilience in an integrated way. This increases the degree of autonomy for those activities.

Business Impact

The adoption of autonomous testing has the potential to significantly improve an IT organization's ability to serve and delight its customers. It can be an enabler for adjusting testing scenarios and overall software quality parameters as part of a continuous quality initiative aimed at optimizing end-user experience. It will also help to constitute a closed-loop system that provides continuous feedback about critical quality indicators.

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 in maintaining the automation as applications under test (AUT) evolve.
- The pressure to innovate quickly in order to differentiate in the market without compromising on quality relies both on 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

- Risk of doing nothing: Waiting until feature-complete autonomous testing solutions are available leads to a loss in competitive advantage, and reduced agility and innovation.
- Unrealistic goals: Underestimating the time required to acquire new skills and setting wrong expectations about the time required to become successful.
- Managing data quality: Gathering, cleaning and processing of data and training of the model are not trivial tasks and require adequate skills. Moreover, they are not yet autonomous processes.
- Internal pushback: Autonomous testing requires significant investment in new areas such as data science and analytics. While this will motivate some team members, others may see the approach as a threat to their known way of working and they may be afraid to adjust.
- Immaturity of tools: Currently available tools are still relatively new, have a narrow scope of technology coverage and still need to prove their value.

User Recommendations

- Set the right expectations about where autonomous testing can provide value, what its current limitations are and what is needed for it to be successful.
- Maximize the impact of autonomous testing by leveraging it as an enabler of a systematic approach to achieve the quality goals of business and development. Focus on key business value enablement and determine where it can help with revenue, cost, and risk management.
- Familiarize yourself with advanced analytics and ML. Invest in augmented analytics tools that employ ML algorithms to mine existing data for current and future projects.
- Allocate ownership and appoint staff with the required skills such as data analytics by identifying the required roles, technologies and practices.
- Select tools that best match available skills and development style by assessing application profiles and different teams' needs.

Sample Vendors

Applitools; Diffblue; Functionize; mabl; ProdPerfect; Testim

Gartner Recommended Reading

Innovation Insight for Autonomous Testing

Innovation Insight for Continuous Quality

Improve Software Quality by Building Digital Immunity

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

AI-Augmented Software Engineering

Analysis By: Arun Batchu

Benefit Rating: Transformational

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

AI-augmented software engineering (AIASE) is the use of AI technologies such as machine learning (ML), natural language processing (NLP) and similar technologies to aid software engineering teams in creating and delivering applications faster, more consistently, and with higher quality. AIASE commonly integrates with an engineer's existing tools to provide them with real-time, intelligent feedback and suggestions.

Why This Is Important

Today's software engineering methods involve writing boilerplate code that saps expert engineers' creativity, limiting their productivity. Novice developers are challenged by the complexity of modern software they need to deliver. Moreover, engineers need to master multiple fit-for-purpose languages that are used to realize a complex software system. AIASE helps in resolving these issues, by acting as an intelligent assistant, a pair-programmer, an expert coach and quality control inspector.

Business Impact

AIASE is enabling creative business problem-solving by automating boilerplate software engineering tasks. It is increasing developer velocity by recommending highly relevant code and library recommendations in a fraction of the time it would take otherwise. It is augmenting quality and testing engineers by allowing tests to self-heal and by automatically creating tests. AIASE models continuously improve their utility by learning from its regular interactions with the engineers and environment.

Drivers

- Increasing complexity of software systems to be engineered.
- Increasing demand for developers to deliver high-quality code faster.
- Continuous modernization of existing systems via replacement or refactoring.
- Increasing impact of software development on business models.
- Development of language server protocols for easy integration into widely adopted integrated development environments.
- Application of deep learning language models to software code.
- Increase in usage of high-quality open source software as model-training data.
- Application of machine learning to models, used within model-driven development platforms.
- Increasing availability of GPUs and TPUs needed to run machine learning workloads in the cloud.
- Increasing computational power of developer machines, especially for running and training machine learning models.

Obstacles

- Lack of deep understanding of generated solutions.
- Limited awareness of the existence of production-ready tools.
- Minimal evidence proving ROI of AIASE.
- Resistance from software engineers who fear job obsolescence
- Lack of transparency and provenance of data used for model training.
- Trade-off between relevance and latency of real-time suggestions.

User Recommendations

- Make the engineering teams aware of, evaluate, and adopt, AIASE tools with a sense of urgency.
- Identify the programming languages within the enterprise best suited to start with.
- Evaluate funding needed to obtain necessary licenses for AIASE, and develop a business case to obtain them.
- Establish a method to measure productivity gains.
- Check the maintainability of AI-generated code, tests, and models (in development platforms).
- Reassure their software engineers about AIASE augmenting — not replacing them.
- Track the rapidly evolving and highly impactful market to identify new products that minimize development toil and improve experience of software engineers.

Sample Vendors

Codota; Diffblue; IBM; Kite; Mendix; Microsoft; NerdVision; OutSystems; SeaLights; Tabnine

Gartner Recommended Reading

Emerging Technologies: Critical Insights Into AI-Augmented Software Development

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

Hypothesis-Driven Development

Analysis By: Jim Scheibmeir

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Hypothesis-driven development (HDD) is software development based on running a series of tests to find the best solution. The core principle is that every release is a test of a hypothesis, and is constructed to be the simplest way to prove or disprove that hypothesis.

Why This Is Important

When HDD is combined with agile and DevOps practices, it allows for the opportunity to accelerate the rate of innovation while reducing waste and managing risk. HDD enables an organization to define requirements and create software solutions in problem spaces where little is known about the best solution. It can prevent developing a solution that has no customers. HDD allows teams to test the market to see if the offering and its features are valued.

Business Impact

HDD is not trying things in an ad hoc way. It is a disciplined approach to innovation that requires functional decisions to be based on verified results, not the opinions of forceful or highly ranked individuals. The greatest benefits are seen where hypotheses can come from different people and can be quickly tested. Organizations using HDD need to understand that proving a hypothesis false is not a failure.

Drivers

Popularized by the book “The Lean Startup” by Eric Ries in 2011, HDD has been widely adopted in software product and game development. It is less common in IT development projects, and its position on the Hype Cycle reflects IT adoption. Gartner's 2020 Agile in the Enterprise survey, conducted online from 8 April to 22 May 2020 (n = 353 IT and IT-business professionals globally), found that 31% of respondents would be utilizing HDD by the end of 2020. HDD adoption is being driven by:

- Need to ensure business growth with changing user patterns, behaviors and new integration opportunities
- Increasing user expectations for application design, quality and experience
- Need to optimize development resources and minimize user fatigue due to application and process changes

Obstacles

- Experimental nature of HDD initiatives, as many IT organizations refuse to start a project where the solution, time and cost, and the user base are not well-understood.
- Frequent changes to user interfaces as well as supporting database and business logic code.
- Lack of talent proficiency in agile and DevOps practices that are required to support fast feature development and deployment to enable HDD.

User Recommendations

- Treat releases as experiments and validate that the new functionality achieves the desired outcome.
- Adopt HDD to guide development in customer-facing projects and internal projects where there is a large and diverse user base.
- Delay using HDD until teams have reached agile and DevOps proficiency.
- Adopt a hypothesis pattern for stories. Replace the traditional story format, i.e., “As an <identify user>, I need to do <task> in order to accomplish <outcome>” or the BDD/Gherkin format, i.e., “given <initial state>, when <event>, then <result>” with an HDD pattern such as: “We believe that <this capability> will result in <this outcome>. We will know we have succeeded when <we see a measurable outcome>.”
- Use lean startup principles to develop solutions with a four-stage HDD cycle: *think it, build it, ship it, tweak it.*

Sample Vendors

CloudBees; GitLab; LaunchDarkly; Optimizely; Split

Gartner Recommended Reading

Solution Path for Agile Transformation

Extend Agile With DevOps for Continuous Delivery

Quick Answer: What Are the Essential Shift-Right Practices for Testing in Production?

Guidance Framework for Adopting Lean UX

Learn From the Spotify Model for Better Enterprise Agile Scaling

MXDP

Analysis By: Arun Batchu, Jim Scheibmeir, Van Baker, Adrian Leow

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

A multiexperience development platform (MXDP) offers development teams an opinionated and integrated set of front-end development tools and "backends for frontends" services. Its purpose is to enable distributed and scalable development (both in teams and architecture) of fit-for-purpose apps across digital touchpoints and interaction modalities.

Why This Is Important

A MXDP unifies front-end application development activities across heterogenous types of apps so that the user experience (UX) across these apps is seamless, persistent and enjoyable. MXDPs ensure maximum reuse and accelerate development velocity.

Business Impact

- Delightful application user experiences that seamlessly and persistently span multiple devices and augment human senses (touch, vision, auditory).
- Simplified and composable construction of complex front-ends that speed-up application development and deployment.
- Smoother development experiences resulting in more engaged and loyal software engineers.
- Reduction of the number of development teams and personnel by allowing cross-platform development.

Drivers

- Need for consistent, seamless and distinctive application experiences that span an increasing number of end-user computing devices.
- Need for ambient (no-touch) experiences accelerated by the pandemic.
- Increased sophistication of multiexperience development platform technology.
- Increased demand for digitalization of industries using heavy machinery or operating in hazardous environments.

Obstacles

- Lack of portability of applications across MXDPs and vendor lock-in.
- Immaturity of immersive technologies.
- Inconsistent implementation of browser models across multiple devices and operating systems.
- Platform licensing costs, especially for consumer facing use cases.
- Learning curve for proprietary stack.
- Lack of and retention of trained personnel.
- Preference for open-source software stacks.

User Recommendations

- Guide teams to consider an MXDP as a supplement to open-source software stacks and low-code application platforms (LCAP).
- Drive innovation by having teams demonstrate the power of MXDPs to business and upper management stakeholders.
- Narrow down a consideration set by using Gartner's Critical Capabilities research for MXDPs.
- Use Gartner's Magic Quadrant research to pick a vendor that best matches the strategic vision and expectations of product execution.
- Evaluate licensing costs at scale, before increasing the impact of MXDPs.
- Develop plans for training and recruiting developers for the chosen MXDP.
- Mitigate lock-in risk by encouraging modularizing of applications and replacing them with open source stacks when appropriate.
- Utilize software engineering skills to extend the MXDP offerings with custom and reusable modules that citizen developers can utilize in no or low code development.

Sample Vendors

Appian; Builder.ai; GeneXus; HCL Software; Mendix; Neptune; Neutrinos; OutSystems; Pegasystems; Salesforce; SAP; ServiceNow; XOne

Gartner Recommended Reading

Technology Insight for Multiexperience Development Platforms

Magic Quadrant for Multiexperience Development Platforms

Critical Capabilities for Multiexperience Development Platform

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 allows them to be understood based on their outputs, and allows questions about their behavior to be answered. Tools that facilitate software observability enable observers to collect and explore high cardinality telemetry using techniques that iteratively narrow the possible explanations for errant behavior. The insights that observability provides are useful to operations, platform and DevOps teams.

Why This Is Important

The inherent complexity of modern applications and distributed systems, and the rise of practices such as DevOps, has left many organizations frustrated with legacy monitoring. These tools and techniques are unable to do more than collect and display external signals, resulting in monitoring that is only partially effective. Observability tools enable a skilled observer — often a software developer or site reliability engineer — to more effectively explain unexpected system behavior.

Business Impact

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

Drivers

The message of observability is resonating powerfully in the industry:

- The term observability is everywhere (e.g., in monitoring products, automation tools and enterprise suites) and 2021 is shaping up to be the year of observability. Although the tools are getting better, the message is being obscured by its ubiquity.
- OpenTelemetry's progress in 2021 as the "observability framework for cloud-native software" has further heightened the profile of observability and its toolchain.

- Traditional monitoring systems capture and examine (possibly adaptive) signals in relative isolation, with alerts tied to threshold or rate-of-change violations. To achieve this, one must be aware of possible issues beforehand, and instrument accordingly. Given the complexity of modern applications, it is unfeasible to do this for all potential issues.
- Observability tools enable a skilled observer — a software developer or a site reliability engineer — to more effectively explain unexpected system behavior, provided enough instrumentation is available. Integrating software observability with artificial intelligence for IT operations (AIOps) to automate subsequent determinations is a potential future development. However, such AIOps-based observability systems do not yet exist.
- Observability is an evolution of longstanding technologies and methods, and monitoring vendors are starting to include observability ideas inside their products, while new companies are building around the idea. Hence, time to Plateau of Productivity will be shorter than normal innovation profiles.

Obstacles

There are a number of obstacles to implementation of observability:

- Conservative nature of IT operations — 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 new technology is slow to be adopted.
- Many vendors, including those in the network and security domains, are using the term “observability” to differentiate their products. However, little consensus exists on the definition of observability and the benefits it provides, causing confusion among enterprises purchasing tools.
- Hyperscale cloud vendors — for example, Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform (GCP) — continue to improve their observability tools. These services may reduce or eliminate the future need for third-party tools.

User Recommendations

Users starting a DevOps journey should assess software observability tools to integrate into their continuous integration/continuous delivery (CI/CD) pipelines, feedback loops and retrospectives. They should also:

- Investigate problems that can't be framed by traditional monitoring by using observability to create time and site reliability engineering (SRE) error budgeting, adding flexibility to incident investigations.
- Enable observability by selecting vendors that use open standards for collection, such as OpenTelemetry.
- Tie service-level objectives (SLOs) to desired business outcomes, using specific metrics and leverage observability tools to understand variations.
- Avoid the implication that observability is synonymous with monitoring. At a minimum, observability represents the internal, rather than external, perspective. Ideally, observability solutions make identifying unknown unknowns less onerous by identifying correlations and patterns in streams of telemetry that may escape notice.

Sample Vendors

Amazon Web Services (AWS); Chronosphere; Google; Honeycomb; Lightstep; Microsoft

Gartner Recommended Reading

Monitoring and Observability for Modern Services and Infrastructure

Magic Quadrant for Application Performance Monitoring

Cool Vendors in Monitoring, Observability and Cloud Operations

Innovation Insight for Observability

Sliding into the Trough

Model-Based System Engineering

Analysis By: Marc Halpern

Benefit Rating: Transformational

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Model-based system engineering (MBSE) is a digital approach to creating models and simulating behaviors to understand systems and systems of systems. Those models prioritize the relationships among key variables within the systems, as well as environmental or operating conditions that influence behaviors, to understand and improve them.

Why This Is Important

MBSE — and its ability to cost-effectively model the behavior of systems — has become important in many industries and the government, as the performance of products, infrastructure, processes and organizations has become increasingly complex and interdependent. A growing number of roles addressing IT, manufacturing, transportation, infrastructure, and environmental and governmental issues look to MBSE as a means of understanding challenges and solving problems.

Business Impact

MBSE does the following:

- Replaces document-centric work environments with models
- Makes information easier to navigate and understand
- Cultivates more “cause and effect” thinking, resulting in better, more disciplined, cost-effective decisions
- Supports the planning and implementation of digital threads and digital twins
- Makes problem solving more efficient, since models connect and relate critical information instead of having to sift through disconnected documents

Drivers

- The need for MBSE among manufacturers has been heightened by the increased use of software as a part of physical systems.
- MBSE can improve the organizational structure in digital enterprises to enable highly collaborative work across different disciplines.
- Manufacturers seek to design and deliver individualized products more efficiently by designing product platforms instead of products. Product platforms are sets of product modules and features that can be mixed and matched to deliver variants of products. MBSE is an efficient means of designing product platforms, instead of products.
- MBSE is core to designing digital threads and digital twins. Digital threads and digital twins are central to digital business strategies among manufacturers.

Obstacles

- The time and cost to produce trustworthy models with sufficient fidelity to real-world systems
- Challenges in transferring the knowledge of best MBSE practices from a small knowledgeable community of experts to a broader market
- Poor adoption of standards that would make long-term use and reuse of the models more efficient
- Difficulty of changing the behaviors of organizations in ways that would make MBSE practice more productive

User Recommendations

- Seek support from service companies with strong practice and experience in system engineering.
- Start with a pilot program of small scale to improve on the design of a physical system, organization or process.
- Apply lessons learned to new initiatives, always with sensitivity that all businesses are systems of systems, as any system improved through MBSE can affect the performance of other parts of a business that the system impacts.

- When using MBSE to improve or solve challenges with each subsystem (for example, the ship), account for the other systems that it affects (for example, the shipyard or a supply chain).
- Ensure integration provides mapping between the requirements of the system being modeled and the technical specifications that meet those requirements.
- Plan for change management, including changes to organizations, roles, processes and practices, with experts in multidisciplinary roles with a focus on system thinking.

Sample Vendors

Ansys; Dassault Systèmes; IBM; Maplesoft; MathWorks; Modelon; PTC; Siemens; Sparx Systems

Gartner Recommended Reading

Innovation Insight for Model-Based System Engineering

Smart Contracts

Analysis By: Martin Reynolds, Avivah Litan, Adrian Leow

Benefit Rating: Transformational

Market Penetration: 1% to 5% of target audience

Maturity: Adolescent

Definition:

A smart contract is a type of blockchain record that contains externally written code, and controls blockchain-based digital assets. When triggered by a specified blockchain write event, a smart contract immutably executes its code and may result in a cascading blockchain event. Smart contracts can only read from, and write to, the blockchain. All off-chain interactions with smart contracts must be handled by agents that map between off-chain assets and on-chain digital assets.

Why This Is Important

Smart contracts can automate business transactions, executing transactions under mutually agreed terms embodied in the code. This approach eliminates human delays and costs, and allows companies to execute far more transactions. This increased capacity comes at minimal incremental cost, thereby increasing efficiency, reducing operating costs and opening new business models. Smart contracts enable competitors to collaborate over shared resources without compromising their independence.

Business Impact

Once smart contracts and business blockchain technologies are functional and effective, smart contracts can significantly improve capital efficiency by cutting administrative overhead and reducing working capital. We estimate that in, for example, manufacturing, smart contracts could increase cash efficiency by 5%.

Drivers

- **Ethereum smart contracts represent proven technology.** Ethereum smart contracts run not just the Ether cryptocurrency, but also enable the derivative cryptocurrencies that run using the Ethereum blockchain. Ethereum smart contracts represent a valuable target for attackers, but attackers have only succeeded in exploiting errors in smart contract implementations, not the Ethereum smart contract system.
- **Oracles provide trusted contract triggers.** Smart contracts operate only on records committed to the underlying blockchain. Distributed oracle technologies that record smart contract trigger events on the blockchain provide the trust necessary for business operations.
- **Use cases demonstrate blockchain links to the real world.** In 2020 and 2021, markets emerged for collectible assets linked to nonfungible tokens (NFTs) through smart contracts (ERC-721 standard). Trading cards, images and art are three such applications. Although the long-term value may be questionable, the cash paid for NFTs demonstrates confidence in the underlying blockchain technology.

Obstacles

- **Legal systems not ready for blockchain.** Enforcing the link between the blockchain token and the real-world asset may not withstand legal challenges, particularly if an asset is incorrectly transferred and the current holder does not relinquish ownership. The problem compounds with disputes that cross legal systems, for example with one party in the U.K. and the other in Brazil.
- **Immaturity of off-chain systems and blockchain code.** Although multiple smart contract platforms exist, few have been tested to an extent sufficient to guarantee correct behavior. In many cases, operation in a controlled environment with a trusted adjudicator can compensate for any issues that emerge, but adjudicators may be unwilling to engage in resolving disagreements.
- **Complexity of smart contracts.** Smart contracts are code, and code has bugs. In a complex smart contract, a developer may even conceal code for personal profit that may be difficult to remove.

User Recommendations

- **Ensure legal agreements bound behavior of your blockchain partners.** Although enforcement may be difficult, a clear framework defining what the platform does and does not do, and how disputes are escalated and arbitrated, is essential.
- **Budget for unresolved errors in smart contracts.** There will inevitably be some failures in smart contracts, and the arbitrator may not be willing to step in if the issue is not significant and obvious. Therefore, you should expect some losses and gains on smart contracts. Focus only on major or systematic errors that are worth the effort of remediation.
- **Manage the risk of smart contracts by means of open code and third-party providers.** There are three mechanisms that can protect against errant smart contracts. The first is to request published source code. The second approach is to use a smart contract mutually sourced from a third party, with guarantees of correct operation. The last approach is to use a third party for indemnification only.

Sample Vendors

Chainlink; Ethereum; Solana; Valid Network

Gartner Recommended Reading

Managing the Risks of Enterprise Blockchain Smart Contracts

Predicts 2020: Blockchain Technology

Guidance for Blockchain Solution Adoption

Garbage In, Garbage Forever: Top 5 Blockchain Security Threats

Behavior-Driven Development

Analysis By: Jim Scheibmeir, Joachim Herschmann

Benefit Rating: Moderate

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

Behavior-driven development (BDD) expresses requirements as behaviors linked to scenarios and outcomes to enhance collaboration between business domain experts, product owners, developers and testers. BDD uses domain-specific language and is implemented with executable specifications from feature files. BDD is a close relative of test-driven development (TDD) and acceptance-test-driven development — practices that shift software testing left in software engineering activities.

Why This Is Important

BDD extends the concept of TDD closer to the business user. BDD also improves traceability and auditability by providing a clear trail from the feature to the code that implements it and vice versa. Because BDD scripts are accessible to a nontechnical audience, they can be developed in cooperation with the IT consumer, thus ensuring the software fulfills all the acceptance criteria. BDD also maximizes the benefits of writing automated tests. The 2021 Hyperautomation Survey found that 14% of business technologists indicated automated software development and deployment were among the three tools used most (based on frequency and volume) to support automation initiatives over the last 12 months.

Business Impact

Software engineering teams — fusion teams, especially — should use BDD to improve collaboration on software products in consideration of the application behaviors and the implication for user experience. Because BDD scripts are accessible to a nontechnical audience, they can be developed in cooperation with business technologists, thus ensuring the software fulfills the acceptance criteria. Writing the tests before the code maximizes the benefits of writing automated tests by ensuring the tests are available as early as possible.

Drivers

- The market demands outstanding user experience as well as fast time to market. Indeed, 30% of respondents to Gartner's 2020 Achieve Business Agility With Automation Continuous Quality and DevOps Survey reported they are currently using or planning to use BDD to improve software quality. BDD provides a mechanism to maintain the fidelity and traceability of artifacts.
- BDD offers a path to agile adoption for organizations in highly regulated industries where there is an assumption that agile does not prescribe detailed documentation.
- BDD facilitates the use of the specifications to drive acceptance test automation.

Obstacles

The impediments to BDD's wider adoption are related to QA process changes, involvement from business and technical roles, and the skill sets it requires. Software engineering leaders struggle to ensure wide adoption of BDD because they fail to adequately influence and manage:

- **Internal pushback:** BDD requires engaging stakeholders from the business and IT and empowering them to be more accountable.
- **Loss of productivity:** Changing organizational culture and engaging in new practices require significant investments of money and time.
- **Limited value and ROI:** Providing continuous quality includes designing a product with quality in mind, building it with clear quality objectives and facilitating the discovery of issues early in development. BDD helps but is only one practice among many software QA practices.
- **Focusing only on tools:** Providing continuous quality requires a change in organizational culture.

- **Learning curve:** The given-when-then expression of Gherkin does not fit the way every organization expresses software requirements.

User Recommendations

BDD is a best practice for agile development and can be useful regardless of your software development methodology. To implement it:

- Start BDD at the beginning of the product life cycle to improve requirements and acceptance criteria. Focus on quality in user stories through collaboration between technical and nontechnical staff. Specify application conditions and behaviors within feature files written in DSL such as Gherkin.
- Improve the development process and specifically the areas around software quality by involving the stakeholders, developers and testers in the creation of the scenario and feature files.
- Describe the desired functionality using the Gherkin given-when-then syntax before writing the code. Use BDD tools to build automated tests that are verifiable by the product owner and business domain experts.
- Invest in training to expand your agile teams' test automation skills.

Sample Vendors

froglogic; SmartBear; Tricentis

Gartner Recommended Reading

Choose the Right Metrics to Drive Agile, DevOps and Continuous Delivery

Application Testing Services Must Evolve to Meet Client Demands in Agile and DevOps

Innovation Insight for Continuous Quality

4 Essential Steps to Implement Test Automation

Progressive Web Apps

Analysis By: Jason Wong

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Progressive web apps (PWAs) are designed to deliver a native app experience without the need to install app binaries on devices. They combine an app shell with service workers installed on desktops or mobile devices, which enable HTML, JavaScript, Cascading Style Sheets and web content to be cached and synchronized for optimal performance. When users opt to enable them, PWAs deliver app-like features such as offline data access, push notifications and a home screen icon.

Why This Is Important

PWA-enabled web applications and sites deliver better performance and improve user engagement. Developers simply embed service workers, supported within the browser, to surface PWA functions. PWAs can be installed directly from a PWA-enabled website and can be shared through links and QR codes. Google and Microsoft allow PWAs to be listed on and deployed through Google Play and the Microsoft Store respectively; however, Apple does not allow this for its App Store.

Business Impact

The main benefits of PWAs include:

1. Users do not need to visit an app store and install an app in order to get the capabilities of PWAs.

2. PWAs can be changed and updated without the requirements to push revisions to an app store and force updates on users' devices.

3. PWAs offer fewer capabilities than native apps, but they can be achieved at a fraction of the cost. Plus, they can be delivered by existing web development skills and teams.

Drivers

- Most leading desktop and mobile browsers have embraced PWAs. Browsers such as Apple Safari, Google Chrome, Microsoft Edge and Mozilla Firefox enable developers to implement service workers (albeit not consistently across desktop versus mobile browsers and operating systems), so that websites can behave like apps.
- PWAs are also being utilized by Huawei, one of the world's largest smartphone makers. In 2020, the Huawei Browser, a Chromium-based browser, supported the installation of PWAs for Huawei Mobile Services-based devices, without Google Services. Additionally, Huawei launched the Quick Apps project with 11 other Chinese mobile phone manufacturers to drive installation-free apps developed on JavaScript, CSS and HTML.
- Since PWAs are based on web standards, development tools and frameworks are also starting to support the creation and enablement of websites as PWAs. These tools include web frameworks like Ionic to low-code development tools that generate responsive web and single page applications that are PWA-enabled.
- PWAs are slowly being adopted for consumer facing websites as the next step beyond responsive web design. Also, Gartner observes that PWA support for specific employee-facing web apps, in lieu of native desktop or mobile apps, is also becoming an option for certain app use cases.

Obstacles

- PWAs are in the Trough of Disillusionment due to the still fragmented user experience (UX) across browsers, fairly basic app capabilities being utilized, and a general lack of consumer awareness of PWAs. Also, Apple continues to take an "arm's-length" approach to supporting PWAs, instead using the term "HTML5 apps" to describe them.
- Penetration rates are still very low overall. At the end of 2020, approximately only 1% of websites included a service worker, and 2.2% had an installable Web App Manifest file (Evidence: Progressive Web Apps in 2021). In December 2020, Mozilla abandoned support of PWAs for the Firefox desktop browser, although it still plans to support it on Android.
- Moving to PWAs in place of native apps is becoming less likely the longer the fragmented nature of service-worker support across operating systems and web browsers persists. Some of the technical limitations have forced application leaders to recognize that PWAs are inadequate for advanced mobile app use cases.

User Recommendations

Application and software engineering leaders should:

- Apply PWA service workers now to improve performance and UX in browsers that support PWA. Don't disregard PWAs' value because the capabilities are not available to all users, but factor in the variance in UX across devices.
- Evaluate PWAs for employee-facing app use cases, such as extending employee portal functionality to a mobile-optimized interface.
- Utilize PWAs in digital commerce as a means of turning web users into mobile-first users by increasing engagement and conversion rates with high-value, frequent interactions and supporting offline browsing of product catalogs.
- Investigate the potential security limitations of PWAs in terms of securing data cached locally on devices that use default web security and encryption technologies, such as HTTPS.
- Utilize JavaScript tools, frameworks and platforms that provide support for PWA capabilities to speed up the process of implementing PWA capabilities in your web apps.

Gartner Recommended Reading

Assessing Progressive Web Apps to Provide Multiexperience Performance and Capabilities

Balance Web and App for a Winning Mobile Marketing Strategy

Key Considerations When Building Web, Native or Hybrid Mobile Apps

Enterprise Agile Frameworks

Analysis By: Mike West

Benefit Rating: High

Market Penetration: More than 50% of target audience

Maturity: Early mainstream

Definition:

Enterprise agile frameworks are a collection of one or more methodologies and associated principles, such as lean and systems thinking.It is structured in a way to enable the delivery of large, complex agile initiatives. They may be top-down, bottom-up or a combination. They are increasingly implemented by organizations scaling up to address enterprise initiatives for complex software or cyber-physical product releases.

Why This Is Important

Enterprise agile frameworks provide organizations with structures, processes and practices that enable delivery of complex products. They can be used to implement, upgrade, migrate and enhance enterprise software and cloud solutions. Their purpose is to make the management and coordination of complex agile releases and evolving solutions not only feasible, but routine and sustainable.

Business Impact

Enterprise agile frameworks (EAFs) provide organizations with:

- Formal approaches for managing the work and deliverables of multiple teams.
- Structures, processes and practices that enable delivery of complex products.
- The ability to manage development of cyber-physical systems.
- Portfolio management discipline for allocating resources and tracking financial benefits.

EAFs also make possible agile configuration, integration, delivery and ongoing management of ERP and other complex corporate solutions.

Drivers

- Trust in agile development has grown across industries and geographies.
- The proportion of those doing most or all in agile has grown to exceed the percentage doing some agile development.
- Organizations have discovered that it is possible to align multiple teams to the same development initiative, but scrum of scrums is not practical for most.
- Consistent practices have become necessary to manage some of the challenges of complex agile development, such as cross-team dependencies.
- Many organizations now require formal governance to manage these complex, multiteam initiatives.
- Regulatory compliance issues increase the need to manage not just development, but the communications around the process.
- Financial expectations of organizations undertaking large-scale development require formal policy, process and practices.

Obstacles

- EAFs require experienced agile scrum teams that have attained predictable and productive velocity with high software quality.
- It requires incorporation of additional practices from extreme programming (XP) and other methods to enhance team productivity.
- It requires DevOps practices and a pipeline (or platform) in operation to deliver production-ready releases.
- It requires support for implementation at the CIO level or above with adequate funds to ensure training, coaching and facilitation of planning processes.
- It requires an internal customer with a need to scale to implement a complex system across many teams of developers.
- It can take as long as three-to-four years for most organizations to mature their agile capability and culture to be ready to scale.
- Not every organization will need to scale across multiple teams or require an enterprise agile framework. Some organizations will adopt more than one framework.

User Recommendations

As an application or software engineering leader scaling agile, you should:

- Drive the framework selection process by using evaluation criteria based on the solutions you will build, rather than a framework's popularity.
- Deliver positive outcomes in the selection process by shortlisting only frameworks that are compatible with your organization's culture, maturity and stakeholder needs. There is no single approach to scaling agile development that suits all organizations.
- Enable successful implementation by selecting a framework with sufficient training, consulting and support in your geography. Training and related consulting services for enterprise agile frameworks are still evolving.
- Ensure success in scaling by building agile team capabilities, engaging leadership around outcome-based value propositions and utilizing a team of change agents to seed and scale implementation.

Gartner Recommended Reading

Adapt Spotify's Chapters and Guilds for Better Business Outcomes

10 Essential Practices for Success in Implementing the Scaled Agile Framework (SAFe)

Bust Silos, Focus on Customers and Enhance Business Outcomes Through Value Streams

Market Guide for Enterprise Agile Frameworks

Microservices

Analysis By: Anne Thomas

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

A microservice is a tightly scoped, strongly encapsulated, loosely coupled, independently deployable and independently operated application service. Microservices architecture (MSA) applies the principles of service-oriented architecture (SOA), DevOps and domain-driven design (DDD) to the delivery of distributed applications. MSA has three core objectives: Continuous delivery, precise scalability and improved stability.

Why This Is Important

Microservices architecture promises powerful application agility, scalability and resilience. It is a way to build cloud-native applications, and it facilitates continuous delivery practices. But the architecture is complex, with disruptive cultural and technical impacts. Misconceptions about microservices often push software engineering teams to use them indiscriminately, leading to overly complex architectures that fail to deliver anticipated benefits and often make things worse.

Business Impact

- Microservices increase business agility by enabling teams to incrementally deliver new features and capabilities in their software products in response to changing business requirements.
- Microservices improve the scalability of the software engineering organization by enabling small teams to work independently to deliver different services within an application.
- Microservices allow teams to change one part of an application, without the delay and cost of changing the entire application.

Drivers

- Software engineering teams adopt microservices architecture to facilitate a continuous delivery practice. The architecture must be combined with strong DevOps practices to enable teams to safely deploy small, independent features to production systems at the frequency at which they are delivered.
- When applied well, the architecture increases the independence of different parts of a large application, enabling multiple development teams to work autonomously and on their own schedules.
- Microservices architecture facilitates the building of cloud-native applications that support robust scalability and resiliency requirements.
- Microservices are frequently deployed in managed container systems, which can dynamically scale service instances in response to load requirements and automatically recover services that have failed.
- When combined with chaos engineering and resiliency practices, microservices architecture enables self-healing systems that can continue to operate through partial outages.

Obstacles

- Microservices architecture and its benefits are often misunderstood, and many software engineering teams struggle to deliver outcomes that meet senior management expectations. For example, microservices should not be shared, and they will not save you money.
- If you aren't trying to implement or improve your continuous delivery practice, you will almost certainly be disappointed with the microservices cost-benefit equation.
- Microservices architecture is complex. Developers must acquire new skills and adopt new design patterns and practices to achieve its benefits.
- Microservices disrupt traditional data management models.
- Microservices require new infrastructure.
- Microservices are related to but not the same as APIs or containers.
- Many software engineering leaders underestimate the cultural prerequisites. Success depends on applying mature agile and DevOps practices and changing team structures to align with service domains.

User Recommendations

- Set clear expectations by defining business goals and objectives for microservices architecture adoption based on realistic cost-benefit analysis of the architecture.
- Use microservices architecture as a tool to help you attain those goals. Don't view microservices as a destination.
- Avoid "microservice washing" conventional SOA, three-tier architecture and integration. Recognize the difference.
- Improve outcomes by creating guidelines for where and when software engineering teams should and should not use microservices architecture.
- Keep application architecture as simple as possible to achieve your goals.
- Address cultural concerns by aligning teams along business domain boundaries, investing in distributed computing architecture skills and improving DevOps practices.

Gartner Recommended Reading

Leading Teams to Success with Microservices Architecture

Designing Services and Microservices to Maximize Agility

10 Ways Your Microservices Adoption Will Fail — and How to Avoid Them

Climbing the Slope

Cloud Event Stream Processing

Analysis By: W. Roy Schulte

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Early mainstream

Definition:

Cloud event stream processing (ESP) services are platform as a service (PaaS) offerings that perform computation on streaming event data in near real time. An event stream is a sequence of event objects arranged in some order (typically by time). Cloud ESP services enable situation awareness and low-latency responses to threats and opportunities, or they can be used to store data streams for use in subsequent applications.

Why This Is Important

ESP's data-in-motion architecture is a radical departure from conventional data-at-rest architecture. It derives information from streaming data in near real time as it arrives. As data and applications migrate to the cloud, organizations are using more cloud ESP services. Providers developed cloud ESP services quickly by leveraging years of research and development for on-premises ESP platforms. Cloud ESP services trail the progress of on-premises ESP on the Hype Cycle by only a small margin.

Business Impact

Compared to on-premises ESP, cloud ESP services may reduce the cost of stream analytics and stream data ingestion, particularly if workloads vary greatly over time. Serverless deployment styles, using on-demand processing models, will further reduce infrastructure management overhead. Cloud ESP brings the generic benefits of cloud: elastic scalability, self-provisioning, continuous patching and versioning, and offloading the work required for high availability and disaster recovery.

Drivers

- Companies have ever-increasing amounts of streaming data from corporate websites, transactional applications, sensors, meters, digital control systems, social computing platforms, news and weather feeds, data brokers, government agencies and business partners.

- Cloud ESP services have become widely available. All of the hyperscale cloud providers and numerous smaller companies offer ESP services as part of their PaaS portfolios. Multiple vendors have bundled cloud ESP services into their IoT platforms, working in tandem with their distributed ESP platforms running in edge locations for industrial applications.
- Cloud ESP services have matured into stable, well-rounded products with hundreds of applications in reliable production.
- Cloud ESP services put ESP within reach of businesses with limited budgets, and replace capital expenditure with operating expenditure.
- The rapid increase in cloud data warehouses is driving demand for cloud ESP services that have specific capabilities to facilitate rapid ingestion and loading of data from streaming sources and real-time change data capture (CDC) sources.

Obstacles

- ESP technology is overkill for most applications that process low or moderate volumes of streaming data or do not require fast response times.
- Cloud ESP services will not replace on-premises ESP platforms for applications that require very low-latency computation with on-premises (including edge) data. Edge deployments in factories, field locations and moving vehicles, and some data center systems need on-premises ESP platforms to minimize latency and to provide better availability where network connections are slow or unreliable.
- Rearchitecting existing business processes to take advantage of cloud ESP requires significant redesign and evolution, which will take time to mature.
- ESP services required low-level programming in Java, Scala or proprietary event processing languages until recently. A new generation of low-code development paradigms is improving developer productivity but is still limited to a minority of cloud ESP services.

User Recommendations

Cloud ESP services are relevant for a growing number of stream analytics and stream data ingestion applications. Data and analytics leaders should:

- Implement ESP through the use of cloud ESP services or on-premises ESP platforms when conventional data-at-rest architectures cannot process high-volume event streams fast enough to meet business requirements.

- Use a SaaS offering that has embedded ESP logic if a product that targets their specific business requirements is available. Verify that the embedded logic can be customized to their specific use cases and SLAs.

- Develop their own ESP-enabled applications on cloud ESP services when appropriate SaaS offerings are not available.

- Use cloud ESP services in preference to on-premises ESP platforms when most or all of the data sources or targets are in the cloud. Also use cloud ESP services if networks are reliable, workloads are bursty and ultra-low-latency is not a requirement.

Sample Vendors

Amazon; Confluent; Google; IBM; Informatica; Microsoft; Salesforce; SAS; TIBCO Software

Gartner Recommended Reading

Market Guide for Event Stream Processing

Adopt Stream Data Integration to Meet Your Real-Time Data Integration and Analytics Requirements

Market Share Analysis: Event Stream Processing Platforms, Worldwide, 2020

Stream Processing: The New Data Processing Paradigm

Cloud Native Architecture

Analysis By: Anne Thomas

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Cloud native architecture is the set of application architecture principles and design patterns that enables applications to fully utilize the agility, scalability, resiliency, elasticity, on-demand and economies of scale benefits provided by cloud computing. Cloud native applications are architected to be latency-aware, instrumented, failure-aware, event-driven, secure, parallelizable, automated and resource-consumption-aware (LIFESPAR).

Why This Is Important

Many organizations are moving to cloud native architecture as they shift their application workloads to cloud native application platforms. Cloud native principles and patterns enable applications to operate efficiently in a dynamic environment and make the most of cloud benefits. Organizations that simply "lift and shift" legacy applications to cloud native platforms often find that the applications perform poorly, consume excessive resources and aren't able to fail and recover gracefully.

Business Impact

- Cloud native architecture ensures that applications can take full advantage of a cloud platform's capabilities to deliver agility, scalability and resilience.
- It enables DevOps teams to more effectively use cloud self-service and automation capabilities to support continuous delivery of new features and capabilities.
- It can also improve system performance and business continuity, and it can lower costs by optimizing resource utilization.

Drivers

- Organizations want to make the most of cloud computing to support their digital business initiatives, but they can't fully exploit cloud platform benefits without cloud native architecture.
- Software engineering teams are adopting cloud native architecture to support cloud native DevOps practices, including self-service and automated provisioning, blue/green deployments, and canary deployments. A basic set of rules known as the " twelve-factor app" ensures that applications can support these practices.
- Cloud native architecture includes practices such as application decomposition (following the mesh app and service architecture [MASA] structure), containerization, configuration as code, and stateless services.

Obstacles

- Cloud native architecture adds a level of complexity to applications, and development teams require new skills, new frameworks, and new technology to be successful.
- Without proper education, architects and developers can apply the principles poorly and deliver applications that fail to deliver the expected benefits. This leads to developer frustration in adopting the new patterns and practices.
- Not every cloud-hosted application needs to be fully cloud-native and developers may be confused about when they need to use particular patterns to address their specific application requirements.

User Recommendations

- Use the twelve-factor app rules and the LIFESPAR architecture principles to build cloud native applications.
- Incorporate cloud native design principles in all new applications irrespective of whether you currently plan to deploy them in the cloud. All new applications should be able to safely run on a cloud platform, even if it doesn't fully utilize cloud characteristics.
- Apply cloud native design principles as you modernize legacy applications that you plan to port to a cloud platform to ensure that they can tolerate ephemeral or unreliable infrastructure. Otherwise, they are likely to experience stability and reliability issues.
- Select an application platform that matches your cloud native architecture maturity and priorities. Recognize that low-code platforms enable rapid development of cloud-ready applications, but they won't provide you with the full flexibility to apply LIFESPAR and twelve-factor principles.

Sample Vendors

Amazon Web Services; Google; Microsoft; Red Hat; VMware

Gartner Recommended Reading

How to Help Software Engineering Teams Modernize Their Application Architecture Skills

How to Modernize Your Application to Adopt Cloud-Native Architecture

A Guidance Framework for Modernizing Java EE Applications

Guidance Framework for Modernizing Microsoft .NET Applications

DevSecOps

Analysis By: Neil MacDonald, Mark Horvath

Benefit Rating: Transformational

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

DevSecOps is the integration and automation of security and compliance testing into agile IT and DevOps development pipelines as seamlessly and transparently as possible, without reducing the agility or speed of developers or requiring them to leave their development toolchain. Ideally, offerings provide security protection at runtime as well.

Why This Is Important

As IT development processes become more agile (including shifts to DevOps operating models), security should be proactively and seamlessly integrated into them — DevSecOps. DevSecOps offers a means of effectively integrating security into the development process, eliminating or reducing friction between security and development, and pragmatically achieving a secure, workable software development life cycle (SDLC).

Business Impact

The goal is to enable development to move faster without compromising on security and compliance. Furthermore, the externalization of security policy enables business units and security organizations, not developers, to define appropriate policies. Policy-driven automation of security infrastructure improves compliance, the quality of security enforcement and developer efficiency, as well as overall IT effectiveness.

Drivers

- Adoption of DevOps and other rapid development practices requires security and compliance testing that can keep up with the pace of development.
- DevSecOps offerings are applied as early as possible in the development process whereas traditional application security testing (AST) tools associated with older development models are applied late in the development cycle, frustrating developers and business stakeholders.
- Testing results need to be integrated into the development process in ways that complement developer's existing workflows, not require them to learn skills seemingly unrelated to their goals.
- The use of open source has greatly increased the risk to the inadvertent use of known vulnerable components and frameworks by developers.

Obstacles

- Developers view security testing tools as slowing them down in getting new releases out.
- Implemented incorrectly, siloed and cumbersome security testing is the antithesis of DevOps.
- Developers don't want to leave their development (continuous integration/continuous delivery [CI/CD]) pipeline to view results of security and compliance testing tools.
- Historically, static application security testing (SAST) and dynamic application security testing (DAST) tools have been plagued with false positives or vague information, frustrating developers.
- The diversity of developer tools used in a modern CI/CD pipeline will complicate seamless integration of DevSecOps offerings.

User Recommendations

- Prepare security and risk management teams for automated integration with DevOps initiatives, and identify the primary skills and technology gaps.
- “Shift left” and make security testing tools and processes available earlier in the development process, ideally as the developers are writing code.
- As zero vulnerability applications aren’t possible, favor automated tools with fast turnaround times with a focus on reducing false positives and allowing developers to concentrate on the most critical vulnerabilities first.
- Prioritize the identification of open-source software (OSS) components and vulnerabilities in development (referred to as software composition analysis).
- Require your security vendors to support out-of-the-box integration with common development toolchain vendors and also support full API enablement of their offerings for automation.
- Require security controls to understand and apply security policies in container- and Kubernetes-based environments.
- Favor offerings that can link scanning in development (including containers) to correct configuration and protection at runtime.

Sample Vendors

Apiiro; Aqua Security; Contrast Security; IBM (Red Hat), NowSecure; Palo Alto Networks; ShiftLeft; Snyk; Sonatype; Veracode

Gartner Recommended Reading

12 Things to Get Right for Successful DevSecOps

Integrating Security Into the DevSecOps Toolchain

Market Guide for Cloud Workload Protection Platforms

Magic Quadrant for Application Security Testing

Critical Capabilities for Application Security Testing

Market Guide for Software Composition Analysis

How to Make Cloud More Secure Than Your Own Data Center

DevSecOps Security Metrics: Use Your Code Repository to Start a Virtuous Cycle

LCAP

Analysis By: Paul Vincent, Jason Wong, Yefim Natis

Benefit Rating: High

Market Penetration: More than 50% of target audience

Maturity: Mature mainstream

Definition:

A low-code application platform (LCAP) is an application platform that supports low-code declarative and often visual, programming abstractions, such as model-driven and metadata-based programming. An LCAP supports end-user interface creation, includes a database, and is used for rapid application development with simplified software development life cycle tooling.

Why This Is Important

LCAPs are one of the most popular types of development tools supporting the low-code paradigm. They support general web and mobile application development with high productivity while reducing the need for deep developer skills, and are mostly cloud-based. They are widely adopted for developer personas ranging from enterprise software developers to citizen developers. Over 200 vendors support a wide variety of business use cases and industry specializations for digital business automation.

Business Impact

Speeding up application development while reducing developer effort can be transformative for business IT. Businesses adopt LCAP to deliver more automation and reduce their application backlogs as well as enable more self-service application development. Mostly cloud-based, LCAP vendors are also accelerating development of new capabilities to increase their use-case coverage and justify their subscription costs.

Drivers

- LCAPs have evolved from rapid application development, business process management technologies and SaaS extension platforms through their evolution of common capabilities around user interface, database, business logic definition, process orchestration and integration of now-ubiquitous API services. The demand to deliver new business automations through applications continues to outstrip conventional application development capacity. This is despite the rise of SaaS usage for standard business services — indeed the latter has resulted in more demands for custom-made extensions that has resulted in a large part of the LCAP marketing being SaaS vendors' LCAP: the market is dominated by Salesforce.

- Through the requirement for LCAP to enable competitive SaaS and complete applications, they have evolved toward multifunction capabilities. LCAPs overlap with the business process automation/iBPMS market for workflow use cases, and the MXDP market for user-interface-driven use cases.

- Some vendors have recently focused on cloud-native scalability to support larger B2C deployments and deeper governance tooling to support remote and distributed developers to enable postpandemic business and IT development fusion team structures. Through support for composing applications from multiple API and service types, LCAPs can cover an increasingly large set of enterprise application requirements, with some enterprises starting to choose them as a strategic application platform.

Obstacles

- Current LCAP market share is heavily biased toward some very large hyperscalers and a few successful independent vendors. However, Gartner commonly speaks with clients that have multiple LCAP offerings deployed across the enterprise.

- LCAPs have been implemented by the main SaaS platform vendors whose market dominance and deep pockets could diminish the opportunities for a large number of small LCAP vendors. However, this really means that for most enterprises the question is not whether to adopt LCAP, but which LCAP(s) will they focus on and invest in.

- LCAP like most low-code trades productivity for vendor lock-in (of both applications and developer skills). Vendor cancellations (like Google App Maker) do occur.

- Licensing models vary across vendors and can be regularly updated by vendors, and may not scale for new use cases. This can lead to vendor disillusionment!

User Recommendations

Software engineering leaders, CIOs and CTOs should:

- Evaluate application lock-ins due to the lack of portability or standards for low-code models. This technical debt will accumulate fast, and means that vendor relationships (and contracts) need to be considered strategic. Architecture needs should be considered — for example whether to use the built-in database for all use cases.
- Weigh annual subscriptions against the productivity benefits during application development (and maintenance). Subscription costs for LCAP are typically per end user, encouraging maximum LCAP adoption per paid-up user.
- Ensure developers have access to the tools that make them productive, whether LCAP or others, and are governed accordingly. Different developers with different skill sets will vary in their successful adoption of LCAP.
- Assess LCAP vendors. The large number of vendors implies possible future market instability, although to date there have been few cases of LCAP retirements.

Sample Vendors

Appian; Betty Blocks; Kintone; Mendix; Microsoft; Oracle; OutSystems; Quickbase; Salesforce; ServiceNow

Gartner Recommended Reading

Magic Quadrant for Enterprise Low-Code Application Platforms

Critical Capabilities for Enterprise Low-Code Application Platforms

Identify and Evaluate Your Next Low-Code Development Technologies

Software Composition Analysis

Analysis By: Dale Gardner

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

Software composition analysis (SCA) products are specialized application security testing tools that detect open-source software (OSS) and third-party components known to have security and/or functionality vulnerabilities, and to identify potentially adverse OSS licensing terms. It is an essential element in strategies to ensure an organization's software supply chain includes secure and trusted components and, therefore, aids in secure application development and assembly.

Why This Is Important

SCA's ability to help ensure the software supply chain is current, free of known vulnerabilities, and properly licensed supports the use of OSS in application development. It is an essential element of application security testing, given the ubiquity of OSS in applications and the potential for significant risk.

Business Impact

- SCA is broadly applicable and beneficial to all types of organizations with application development efforts. It should be considered a foundational element of application security testing.
- Its primary users are application development and security teams, tasked with ensuring the integrity of open-source components — and less frequently commercial software — in development.
- License assessments, once the primary use case for SCA, remain an important function for legal and sourcing groups.

Drivers

- SCA has moved from a mature to an early mainstream position in this Hype Cycle. This unusual reversal is a consequence of multiple changes, including a number of significant technological innovations (e.g., greater integration with other types of testing, more sophisticated analysis of the potential impacts of a package upgrade) and its vastly broader applicability to all types of organizations.
- The underlying driver for the growing importance of SCA is the increased use of OSS in application development. The prevalence of OSS, both as individual packages and within container images, has led to SCA becoming a key control on CI/CD pipelines and containerized environments.
- Repeated instances of high-impact vulnerabilities in OSS, rendered pervasive because of the underlying components' broad use across organizations, along with ever-present supply chain attacks, demonstrate the need to better understand the risks posed by OSS and commercial software packages.
- SCA analyzes code in use and reports issues gathered from information collected from sources such as OSS communities, private research and publicly available security vulnerability databases. Information typically includes intellectual property (IP) ownership, known security and functionality vulnerabilities, and references to the most recent versions of components. Some tools supplement this information with guidance on a preferred update version, balancing stability, remediation of flaws and potential adverse impacts on the functionality of existing code, all of which boosts developer and security team productivity and efficiency. In a few cases, tools have begun to incorporate assessments of operational risk, in an effort to help prevent supply chain attacks.
- SCA technology can help ensure developers are meeting compliance requirements and ethical and legal standards for code use, reducing the likelihood of unwanted or unapproved code that creates risks to an organization's intellectual property.

Obstacles

- The quantity of OSS is inestimable. The number of packages, languages and interdependencies creates an intricacy impossible for a single tool to manage. SCA tools specialize in a subset of languages, limiting utility. Development teams with diverse portfolios will turn to multiple tools, increasing costs and complexity.
- SCA is — incorrectly — seen as a solution to software supply chain attacks. While the tools report known vulnerabilities, with few exceptions they don't test code to identify new issues. Vendors will expand the scope of their research to identify signs of package compromise, and expand testing to discover issues proactively.
- Commercial software libraries are often outside the scope of SCA, but pose many of the same risks. This leaves a protection gap.
- Tools are increasingly optimized for application development use cases. But, legal and sourcing teams still rely on them to identify licensing issues. Tools will develop alternative interfaces to support different users.

User Recommendations

- Examine SCA technologies as a key ingredient of every software security program, used in conjunction with other AST tools. AST vendors should provide this capability built-in or via a partnership with a dedicated SCA vendor, and through dedicated commercial and open source tools.
- Analyze SCA warnings on licensing issues to address the legal issues stemming from components' IP ownership or license terms. This should be done by legal experts.
- Use SCA tools on a regular basis to audit repositories that contain software assets to ensure the software developed and/or used by the enterprise meets organizational standards.
- Utilize SCA tools to inspect the components application developers plan to use. SCA tools fit well within DevSecOps workflows, where scanning can be automated as part of the rapid development processes.
- Use SCA tools in conjunction with a formal corporate IP strategy that has established clear responsibility across the company.

Sample Vendors

Checkmarx; GrammaTech; Hdiv Security; JFrog; Snyk; Sonatype; Synopsys; Veracode; WhiteSource

Gartner Recommended Reading

Market Guide for Software Composition Analysis

Critical Capabilities for Application Security Testing

Magic Quadrant for Application Security Testing

How to Manage Open-Source Software Risks Using Software Composition Analysis

Appendixes

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 2021)

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 (for example, improved user experience) that will be difficult to translate into increased revenue or cost savings

Source: Gartner (July 2021)

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 constrains replacement	Maintenance revenue focus
Obsolete	Rarely used	Used/resale market only

Source: Gartner (July 2021)

Acronym Key and Glossary Terms

AI	artificial intelligence
IDE	integrated development environment
LCAP	low-code application platform
OSS	open-source software
SCA	software composition analysis
SDLC	software development life cycle

Evidence

Gartner's 2021 Hyperautomation Study was conducted online during March 2021 among 558 business technologists from North America (n = 226), Europe (n = 146), Latin America (n = 78) and APAC (n = 108).

To be qualified to answer the survey, respondents were required to:

- Have created, built or coded analytics or technology capabilities, on their own or with input from others, in the last 12 months.
- Have used at least one of the tools to produce analytics or technology capabilities for work in the last 12 months. Twenty-one tools were considered under four categories — application development tools, automation tools, integration tools, and data science and AI tools.

Results of this study do not represent global findings or the market as a whole, but reflect the sentiments of the respondents and companies surveyed.

Recommended by the Authors

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

Understanding Gartner's Hype Cycles

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

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

Client Question Video: Which Platforms and Tools Should I Use to Deliver High-Quality Software?

Cool Vendors for Software Engineering Technologies

Reduce Friction to Boost Software Engineering Team Productivity

How to Establish a Reskilling/Upskilling Talent Development Program for Software Engineering

---

Table 1: Priority Matrix for Application Architecture and Development, 2020

Benefit ↓	*Years to Mainstream Adoption*			
	Less Than 2 Years ↓	***2 - 5 Years*** ↓	***5 - 10 Years*** ↓	***More Than 10 Years*** ↓
Transformational		DevSecOps Observability Smart Contracts	AI-Augmented Software Engineering Model-Based System Engineering	
High	Enterprise Agile Frameworks LCAP Software Composition Analysis	Browser-Based IDEs DesignOps DevOps VSDPs DevOps VSMPs GitOps Microservices	API Security Testing Autonomous Testing Chaos Engineering Continuous Quality MXDP Performance Engineering Quality Engineering	
Moderate		Cloud Event Stream Processing Cloud Native Architecture Hypothesis-Driven Development Progressive Web Apps	Behavior-Driven Development DevOps Test Data Management Graph APIs Innersourcing Microapps	
Low				

Source: Gartner (July 2021)

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 2021)

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 (for example, improved user experience) that will be difficult to translate into increased revenue or cost savings

Source: Gartner (July 2021)

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 constrains replacement	Maintenance revenue focus
Obsolete	Rarely used	Used/resale market only

Source: Gartner (July 2021)