Hype Cycle for Application Architecture and Integration, 2023

Published 1 August 2023 - ID G00792325 - 98 min read

By Analyst(s): Andrew Comes, Wei Jin

Initiatives: Software Engineering Technologies; Adopt Modern Architectures and Technologies

Digital business ambitions drive application architecture and integration innovation through modern architectures and experiences that unleash the value of technology investments via connectivity. Software engineering leaders use technologies in this Hype Cycle to deliver digital business ambitions.

Analysis

What You Need to Know

Demand for innovation, agility and scalability drives software engineering leaders to pursue technologies that deliver their organization's ambitions, without exceeding its risk appetite. Generative AI introduces opportunities for incorporating AI-augmented capabilities into the developer experiences of platforms and tools.

However, increasingly complex application portfolios make it more and more challenging for software engineering leaders to modernize and deliver new application capabilities. The need to achieve greater business agility, remain competitive and innovate hinges on integration infrastructure composed as part of a broad, cohesive strategy to deal with the disruptive changes associated with digital business transformation.

This Hype Cycle, along with Hype Cycle for Software Engineering, 2023, reflects the position, rate of adoption and speed of maturation of innovative technologies and practices. Many of these innovations can have a short-term to midterm impact on software engineering leaders' strategies and tactics; however, they collectively pave the way for the composable business revolution.

The Hype Cycle

This year's Hype Cycle for Application Architecture and Integration highlights these trends:

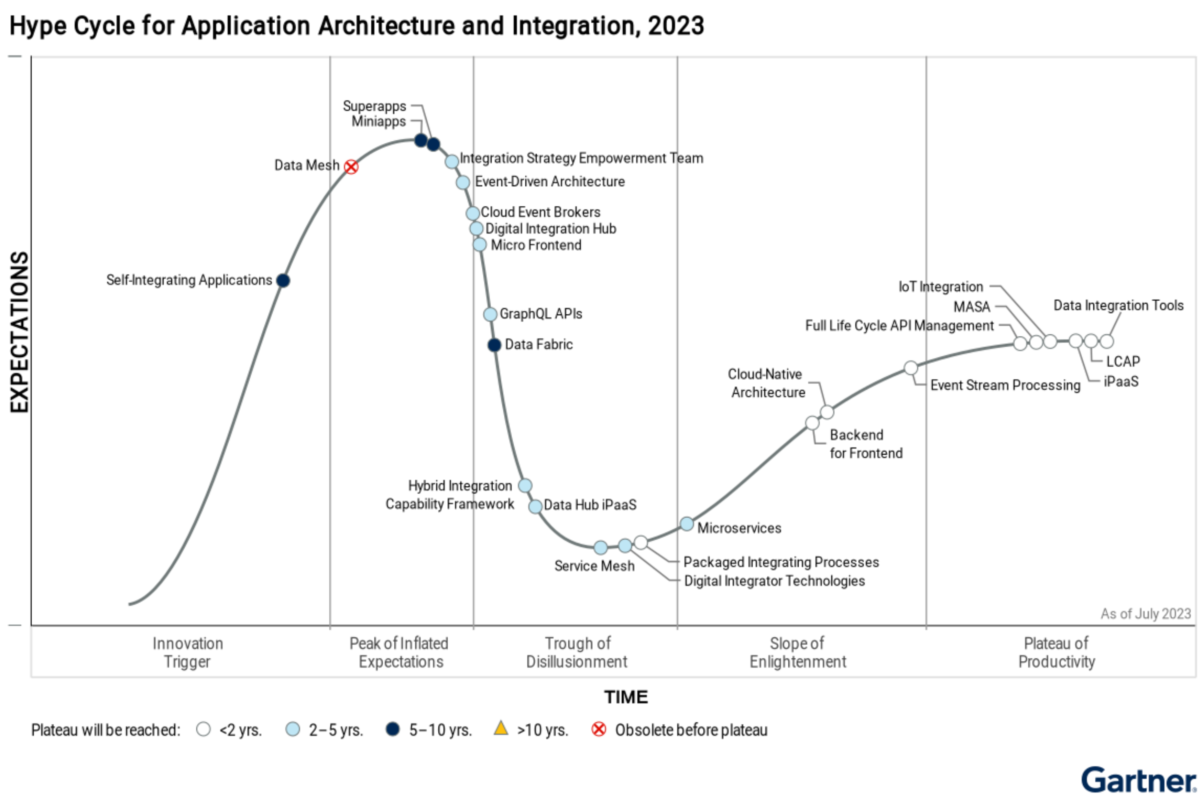
- **Assessment of artificial intelligence and digital integrators** — Software engineering leaders responsible for integration are assessing AI-augmented integration capabilities, especially those focused on generative AI. They seek to use this technology to distribute and accelerate integration development through a conversational interface, as well as optimize the existing integrations on the platform. Digital integrator technologies and self-integrating applications capitalize on opportunities for using generative AI and augmented integration to improve the quality of deliverables and speed of development.

- **Microservices emerges from the Trough of Disillusionment** — Since 2019, microservices have been in the Trough of Disillusionment because Gartner has seen microservices architecture initiatives fail, or run into significant problems, due to complexity and lack of skills. Now, with increased understanding of microservices architecture and when it is appropriate, as well as when it is not appropriate, microservices architecture has moved to the Slope of Enlightenment.

- **Seamless sharing of applications and data** — Capabilities for integration are advancing, with a growing emphasis on distributing low-code and no-code development, API enablement, and event-driven design. Implementing these capabilities lays the foundations for organizations' application portfolios to easily access, compose and provision a range of business functions and data. Such implementations led to developments in packaged integration processes, integration platform as a service (iPaaS), data integration tools and API management to support digital business ambitions.

- **Extensible platform architecture and deployment** — Maturing cloud-based and hybrid offerings make integration and application infrastructure broadly applicable and easier to build and manage. Leaders responsible for low-code approaches adopt application PaaS, low-code application platforms and various PaaS technologies for faster time-to-value and increased developer and integrator productivity. This trend spawns diverse opportunities to exploit PaaS options across a variety of cloud-based technologies featured in Hype Cycle for Cloud Platform Services, 2023.

Figure 1: Hype Cycle for Application Architecture and Integration, 2023



The Priority Matrix

Software engineering leaders should closely monitor the following innovations, which will provide the greatest benefits within the shortest times or at current mainstream adoption:

- Integration platform as a service
- Data integration tools
- Full life cycle API management
- Low-code application platform (LCAP)

Some innovations will take longer to achieve mainstream adoption relative to the ones above, but have proven to deliver high or even transformational value, including:

- Integration data hub
- Digital integration hub
- Digital integrator technologies

- Event stream processing
- Integration strategy empowerment teams (ISETs)
- Packaged integration processes

Adoption of these innovations requires investment in skills and presents some risks because of their intrinsic complexity or still-limited industry support. However, their market penetration is growing, due to many successful deployments and associated lessons learned.

Other innovations in this Hype Cycle are either relatively mature but have a moderate impact, or their low level of industry adoption dilutes their potentially high benefits.

Finally, a small number of innovations are still in the initial stages of their life cycle, so software engineering leaders should assess the risks and rewards associated with adoption.

Table 1: Priority Matrix for Application Architecture and Integration, 2023

(Enlarged table in Appendix)

Benefit	*Years to Mainstream Adoption*			
↓	*Less Than 2 Years* ↓	*2 - 5 Years* ↓	*5 - 10 Years* ↓	*More Than 10 Years* ↓
Transformational	Event Stream Processing		Data Fabric Self-Integrating Applications	
High	Data Integration Tools Full Life Cycle API Management IoT Integration iPaaS LCAP MASA Packaged Integrating Processes	Cloud Event Brokers Data Hub iPaaS Digital Integration Hub Digital Integrator Technologies Event-Driven Architecture Hybrid Integration Capability Framework Integration Strategy Empowerment Team Microservices	Superapps	
Moderate	Backend for Frontend Cloud-Native Architecture	GraphQL APIs Micro Frontend	Miniapps	
Low		Service Mesh		

Source: Gartner (August 2023)

On the Rise

Self-Integrating Applications

Analysis By: Keith Guttridge

Benefit Rating: Transformational

Market Penetration: Less than 1% of target audience

Maturity: Embryonic

Definition:

Self-integrating applications will use a combination of automated service discovery, metadata extraction and mapping, automated process definition, and automated dependency mapping to enable applications and services to integrate themselves into an existing application portfolio with minimal human interaction.

Why This Is Important

Integrating new applications and services into an application portfolio is complex and expensive. Gartner research shows that up to 65% of the cost of implementing a new ERP or CRM system is attributable to integration. The technology for enabling applications to self-integrate exists in pockets, but no vendor has yet combined all the elements successfully. As applications develop the ability to discover and connect to each other, the amount of basic integration work will dramatically reduce.

Business Impact

Self-integrating applications can:

- Improve agility, as the time to onboard applications and services is massively shortened.
- Cut costs by up to 65% when onboarding new applications and services.
- Reduce vendor lock-in, as platform migration becomes simpler.
- Improve the ability to focus on differentiation and transformational initiatives, as the “keeping-the-lights-on” burden is dramatically reduced.

Drivers

- Cloud hyperscalers provide features such as service discovery, metadata extraction, intelligent document processing and natural language processing.
- Automation or integration vendors provide features such as intelligent data mapping, metadata extraction, next-best-action recommendations, process discovery and automated decision making.
- SaaS vendors provide features such as process automation, packaged integration processes, portfolio discovery and platform composability.
- In the new era, intelligent application portfolio management is placed on top of augmented integration platforms in order to properly address the challenge.
- Generative AI simplifies the build process to create integration processes.

Obstacles

- Embedded integration features within SaaS are good enough to enable organizations to get started quickly, thus stalling investment in improving self-integration capabilities.
- Generally, organizations are not well aware of the availability of augmented integration technologies for enabling self-integrating applications. Many organizations still view integration as a complex issue requiring specialist tools.
- There is not a clear market leader that is looking to push this technology forward as the major application vendors look to protect their customer bases.
- Complex scenarios across multiple datasets and service interfaces are too challenging for the current technology. Organizations place too much trust in the solution to do the right thing. Ownership and visibility of the integrations might become contentions within the organizations.

User Recommendations

Software engineering leaders responsible for integration should:

- Ask your major application vendors about the interoperability of applications within their portfolios. This is the area where self-integrating applications are most likely to emerge first.
- Investigate integration vendors that have augmented artificial intelligence features to automate the process of onboarding applications and services into a portfolio.

- Manage your expectations for ease of integration. Self-integrating applications will provide just enough integration with the rest of the application portfolio to enable a new application to work efficiently.
- Keep track of governance capabilities. Who can authorize access? Has the appropriate observability been established? Is everything fully audited? Does something need to change? An organization's integration landscape is an ever-evolving environment, and each integration has a life cycle that needs to be maintained.

Sample Vendors

Boomi; IBM; Microsoft; Oracle; Salesforce; SAP; SnapLogic; Tray.io; Workato

At the Peak

Data Mesh

Analysis By: Roxane Edjlali, Ehtisham Zaidi, Mark Beyer, Michele Launi

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

Data mesh is a data management approach. Though not an established best practice, it supports a domain-led practice for defining, delivering, maintaining and governing data products. Data products are a packaging and delivery mechanism for data that must be easy to find and use by data consumers (business users, data analysts, data engineers or other systems). Data products must also fulfill a contract (terms of service and SLAs) between the provider and the consumer.

Why This Is Important

The definition of data mesh is evolving as the market explores the approach. Data mesh represents a potential alternative or complement to centralization-only data management strategies for analytics and other use cases. Organizations continuously seek a means to balance data requirements for repeatability, reusability, governance, authority, provenance and optimized delivery of data. Data mesh is a skills- and resource-intensive approach that shifts responsibility and authority back to subject matter experts (SMEs) in each data domain.

Business Impact

From a governance and authority perspective, data mesh relies on a federated governance approach that can delegate authority to the business and data domain SMEs. SMEs are assumed to exhibit the greatest experience in capturing and using data within their domain of expertise. They are responsible for determining guidance and processes for creating, managing and preventing unnecessary proliferation of data products. The goal of the mesh is to provide ready access to data products.

Drivers

- Data mesh provides a model that allows for decentralized data management, which aligns to organizational needs.

- Data mesh gives domains the flexibility they need to build data products that meet their required use cases. It also gives domains more control over the use of those data products across the enterprise.
- By leveraging existing assets instead of centralizing the data architecture, data mesh can reduce the time and effort required to enable data reuse throughout the enterprise. Data mesh asserts remediation for flexibility, scalability and accountability issues in approaches like centralized data warehouses, data lakes and data hubs.
- In Gartner client interactions, delays in data access and utilization are the most frequently reported issues from organizations seeking to deploy data mesh. Organizations question the success of data centralization, which can't meet all analytical use cases.
- Data mesh emerges as a compromise to respond to delivery issues, budget constraints, and misunderstandings between central teams and lines of business. Centralized approaches are often detached from the broader business domain requirements.

Obstacles

- Data management maturity and skills are required for data governance at the domain level, data completeness, application design and deployment, data quality, data provenance, systems architecture, and analytics data management.
- Data products must be able to meet the SLAs of the other groups sharing, reusing or accessing them. The associated skills may not be present in the BUs.
- Inappropriate identification of either data details or correct integrity for combining them may cause data product proliferation, thus increasing management and maintenance and necessitating reengineering to reconcile different interpretations of the data.
- Data mesh implementations and practices do not follow any specific guidelines. They cannot be vetted against standardized, or even competing, approaches. Implementations vary and may incorporate multiple approaches (e.g., marketplace experiences, virtualized views or subject-specific data marts).
- Data mesh will be obsolete before the plateau. The practice and supporting technology will evolve toward data fabric as organizations start collecting passive metadata.

User Recommendations

- Commit to building a distributed data management team, as the data mesh concept is highly dependent on the organizational model and the distribution of skills across central IT and LOBs.
- Assess data products for business domain alignment and efficiency gains upon delivery. Data strongly aligned to a single domain with broad utility across the enterprise may provide lower risk for initial data product efforts.
- Control data product proliferation by monitoring technical debt and ensuring that data products continuously evolve to meet changes in usage and scope.
- Start a metadata management program in parallel with data mesh, and collect passive metadata. This approach will allow data mesh to evolve toward metadata activation over time, guiding data product operationalization, value justification and greater transparency in the uses of data.
- Mitigate irresponsible data management by addressing management and governance contention issues in data product design within the data domains.

Gartner Recommended Reading

Quick Answer: What Is Data Mesh?

Quick Answer: Comparing Data Fabric and Data Mesh

Data and Analytics Essentials: Data Fabric and Data Mesh

Quick Answer: How Are Organizations Overcoming Issues to Start Their Data Fabric or Mesh?

2023 Planning Guide for Data Management

Miniapps

Analysis By: Tigran Egiazarov

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

A miniapp is a discrete, focused and autonomous application that deploys and operates within the context of a container application platform rather than a core OS (iOS or Android). It is designed to enhance an app's core capabilities without the need for separate installation or switching to external applications. A miniapp must be tightly scoped and composed of user interface, business logic and data components, which optionally interact with a back-end service via API.

Why This Is Important

Miniapps can be applied to multiexperience development by delivering a consistent and cohesive user experience (UX) across different platforms (mobile, web, immersive technologies). The miniapp pattern enables businesses to build a front-end experience that can run in a superapp platform — expanding an organization's reach to new audiences and enabling more customer convenience. It also solves superapp extensibility challenges, allowing application developers to integrate new features and services.

Business Impact

Miniapps can provide value in these ways:

- Miniapps enable a company to reach more customers. They also offer new monetization opportunities (in-app purchases, payment services, revenue sharing) for superapp platform owners who offer their superapp platform to other developers.
- With miniapps, organizations can provide enhanced UXs across multiple platforms (multiexperience).
- Miniapps can also improve productivity and reduce user flow complexity, because multiple miniapps can be used in one superapp.

Drivers

- Miniapps allow both superapps platform owners and third-party service providers to reach new audiences. Miniapps have gained traction as part of mobile app development, and also based on the growth of conversational platform technologies.

- The superapp trend for both consumer-facing and employee- (internal) facing use cases is also driving miniapps. Superapps provide the runtime mechanism for the creation and distribution of miniapps within their ecosystems. The WeChat and Alipay apps in China are examples of superapps that allow third parties to create and deploy miniapps (also known as miniprograms) within their apps. Enterprises targeting the Chinese consumer market have to build miniapps for both WeChat and Alipay, and this trend is spreading to Southeast Asia, Africa and Latin America. At the same time, enterprises are increasingly adopting internal-facing superapps to foster employee engagement.

- On the technology side, there are an increasing number of mobile frameworks like React Native, Flutter and Cordova, as well as the use of WebView inside superapps, rendering an HTML/JavaScript/Cascading Style Sheets micro-front-end. These frameworks make development of superapp platforms easier, which reinforces adoption of miniapps.

- Enterprise collaboration and messaging platforms, such as Microsoft Teams and Slack, are taking cues from consumer superapps to enable third parties to create and distribute miniapps within their main desktop or mobile apps. It is essential to distinguish the miniapp, which provides a complete new experience, from the plug-in, which extends the existing experience with new features or user flow.

- Miniapps are part of the evolution of the app economy, allowing extension of the existing UX and business capabilities to a new platform (e.g., a superapp platform) and elevating miniapps to the same level as iOS, Android and other applications.

Obstacles

- Low levels of standardization (e.g., the stagnant W3C miniapp standard) and interoperability within superapps do not enable miniapp portability. This increases the investment and effort required to support "yet another platform."
- Designing for miniapps requires understanding of the user journey and the breaking down of interactions into specific modular use cases for miniapps. This poses a significant UX challenge to maintain alignment across mobile apps, superapps and conversational apps such as chatbots.
- The lack of comprehensive miniapp frameworks significantly restricts options for developers. Moreover, some organizations use proprietary platforms like low-code application platforms (LCAPs) and multiexperience development platforms (MXDPs) to build cross-platform applications, which often lack support for specific superapps.
- Ensuring security and governance is a crucial concern for the miniapp-superapp relationship. A mutual verification process must be established to serve as a layer of trust and accountability between both parties.

User Recommendations

- Align with your company's strategy by presenting a solid business case before deciding to build miniapps for a particular superapp. This will avoid the risk of unnecessary expenditure.
- Analyze your users' superapp preferences and be ready to adapt more quickly to support ever-increasing digital experiences by using miniapps to facilitate multiexperience development.
- Identify suitable development frameworks or technologies for miniapp enablement across your target touchpoints (such as web, mobile apps, conversational platforms and immersive technologies, if supplied by superapp).
- Avoid functional conflicts by managing governance of the miniapp runtime container's capabilities (such as permissions, user consent and location service).
- Use miniapps to deliver functionality into third-party superapps.
- Isolate miniapp code from the superapp platform by investing in a superapp abstraction layer (SAL) to maintain flexibility and reduce the risk of vendor lock-in.

Sample Vendors

DCloud.io; DronaHQ; GeneXus; Ionic; KOBIL

Gartner Recommended Reading

How to Make the Right Technology and Architecture Choices for Front-End Development

Adopt a Mesh App and Service Architecture to Power Your Digital Business

Is React Native or Flutter The Best Fit For Building Mobile Apps?

Key Considerations When Building Web, Native or Hybrid Mobile Apps

Quick Answer: Are Cross-Platform Mobile Apps As Good As Native Apps?

Superapps

Analysis By: Jason Wong

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

A superapp is a mobile app that provides end users (e.g., customers, partners or employees) with a set of core features, as well as access to independently created miniapps. The superapp is an open platform to deliver a miniapps ecosystem. Users can choose miniapps from this ecosystem to activate for consistent and personalized app experiences. There is no separate app store or marketplace for miniapps; instead, superapp users discover, activate and deactivate miniapps in the superapp.

Why This Is Important

Users demand mobile-first experiences that are powerful and easy to use. Superapps have expanded outside China and South Asia to India (e.g., Tata Neu, MyJio and Paytm); Latin America (e.g., Rappi, PicPay, Mercado Libre); and the Middle East/Africa (e.g., M-PESA, Careem and Yassir). The superapp concept is rapidly expanding to employee-facing use cases, such as frontline workers, and employee communications and engagement, such as Walmart's me@Walmart and Wipro's MyWipro apps.

Business Impact

Organizations can create superapps to consolidate multiple mobile apps or related services that reduce user experience (UX) friction (such as context switching) and development effort. Superapps can achieve economies of scale and leverage the network effect of a larger user base and multiple providers. Superapps provide a more-engaging experience for their customers, partners or employees. They improve UX by enabling users to activate their own toolboxes of miniapps and services.

Drivers

- Superapps are gaining interest from forward-thinking organizations that embrace composable application and architecture strategies to power new digital business opportunities in their industries or adjacent markets.
- Superapps are growing beyond mobile apps for consumer use cases. Frontline and remote work trends are driving the evolution of employee communications apps into workforce superapps through the addition of plug-ins for HR, payroll, shift management and other miniapp functions.
- The superapp concept is expanding into enterprise software as a service (SaaS) applications and tools, such as workflow, collaboration and messaging platforms (e.g., Slack and Microsoft Teams, which already have a large number of add-on apps to their main applications). Superapps are starting to expand to support a wide range of modalities, including chatbots, Internet of Things (IoT) technologies and immersive experiences.
- To achieve agility and digital transformation, a superapp advances the concept of a composite application that aggregates services, features and functions into a single app. Multiple internal development teams and external partners can provide discrete services to users by building and deploying modular miniapps to the superapp. This development ecosystem also amplifies the superapp’s value, by providing convenient access to a broader range of services in the app.

Obstacles

- There are numerous technical ways to build a superapp, but creating the business ecosystem can become a bigger challenge than technology implementation. A superapp serves as a platform for internally developed miniapps across the business and for third-party, externally developed miniapps. Business partners are needed to create an extended ecosystem for monetization by deploying miniapps to an established user base.
- Another obstacle is getting the UX design of a superapp right for the audience, and also having consistency of the miniapps published to the superapp. Different user personas prefer to interact differently with miniapps — for example, some prefer single, task-focused miniapps versus others preferring everything at their fingertips. Inconsistent UX patterns in a superapp could negatively affect adoption and retention.
- Data sharing could be complex, including simple user authentication, such as single sign-on (SSO), and tracking user preferences or app usage history.

User Recommendations

- Educate partners on the innovations and business value a superapp strategy can drive to improve or consolidate mobile apps.
- Ensure that there is a sound business model and organizational structure to support the distributed development ecosystem for miniapps.
- Secure executive sponsorship by preparing the partnership strategy and financial case for funding as a digital business initiative.
- Identify core features in your superapp (e.g., commerce, communications or collaboration) that will drive a critical mass of adopters and create interest on the part of developers to serve those users
- Offer an easy developer experience and convenient developer tools (e.g., APIs, design guidelines, software development kits [SDKs] and frameworks) for partners to build, test, register and submit miniapps for potential monetization.
- Define security and data protection needs by establishing governance reinforced with common platform implementation to satisfy security and data protection constraints.

Sample Vendors

Alipay; DingTalk GeneXus; Ionic; KOBIL; LINE; Microsoft; PayPay; Paytm; Slack; WeChat

Gartner Recommended Reading

Quick Answer: What Is a Superapp?

Quick Answer: How Does a Superapp Benefit the Digital Employee Experience?

How Banks Can Take On Super-Apps

Top Strategic Technology Trends for 2023: Superapps

Integration Strategy Empowerment Team

Analysis By: Andrew Comes

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

Integration strategy empowerment teams (ISETs) operate as service providers, responsible for the design, implementation and delivery of integration strategies. ISETs help deploy the integration platform, disseminate best practices, provide training, support and consulting services, and lead the integration community of practice. An ISET serves many personas across an organization, such as integration specialists, software engineers, IT administrators, ad hoc integrators and citizen integrators.

Why This Is Important

Many organizations have an integration competency center (ICC) which centrally delivers integration projects for the organization’s units. This model can improve consistency and quality, but its centralized nature fails to deliver results at the pace the business demands, driving teams to bypass its function. An ISET increases the speed of delivery by empowering teams and individual users in business and IT to fulfill integration work through a shared technology platform.

Business Impact

An ISET enables business agility and reduces time to value by decentralizing integration work to accommodate affected domains, while keeping integration costs under control and ensuring compliance with integration governance. In the long term, it drives composability by empowering decentralized fusion teams to collaboratively build new applications by composing packaged business capabilities using an agile method via orchestration, integration and automation.

Drivers

A growing number of midsize and large organizations will pursue a decentralized, self-service integration approach, and establish an ISET. This trend is driven by:

- The increasing amount of work needed to support the competing integration use cases stemming from business-efficiency-focused initiatives, modernization initiatives, migration projects and from the proliferation of SaaS predominantly.
- Application teams' desire to incorporate self-service integration work in their agile, DevOps-enabled application delivery processes.
- Organizations' adoption of composability, which enables a wide range of personas to build focused and customized applications by assembling and integrating predefined "building blocks" (or "packaged business capabilities").
- The widespread vendors' commitment to improve the user experience on their integration platforms, appealing to a range of personas, including integration specialists, software engineers and IT administrators, as well as ad hoc and citizen integrators.
- The facilitated access to integration capabilities embedded in SaaS applications and the growing offering of packaged integration processes, which appeals to ad hoc and citizen integrators.
- The expanding availability of enablement teams implementation methodologies and services from integration platform providers, consulting companies and systems integrators (SIs).

Obstacles

- Creating, maintaining and evolving a useful HICF requires investments in technology and technical, methodological and organizational skills.
- It is complicated to support a large population of "integrators" (up to hundreds or even thousands) with highly differentiated IT skills — from advanced for integration specialists to minimal for citizen integrators. Limited industry experience makes it difficult to find support and skills to help the ISET set up and quickly overcome the organizational, methodological and technical learning curves.
- It is difficult to interpret policies and then manage governance and compliance in an adaptive way despite the highly decentralized environment, while avoiding hindering collaboration and agility.
- The distributive nature of the ISET complicates adherence to the strategy set by the ISET if the strategy is enforced by bureaucracy, as business units may see the ISET as an obstacle instead of a service provider.

User Recommendations

- Establish your ISET by taking its possible size into account. An ISET's size can vary from a few full-time employees to dozens (or more) depending on your organization (midsize or large) and the scale and complexity of your integration challenges.
- Position the ISET as a service provider focused on empowering self-service integration, instead of a bureaucratic organization which sets excessive rules, processes, procedures and policies, grants unnecessary approvals, and conducts redundant reviews. Staff the ISET with the skills and mindsets needed to act as an enabling entity, rather than an "integration factory" that creates delivery bottlenecks.
- Determine which integration personas are in scope and out of scope to ensure due diligence across distributed use cases. Establish KPIs that measure the ISET's ability to serve its constituents' digital ambitions.
- Implement the ISET model in steps, making it easier to justify the investments in terms of business value or technical benefits.

Gartner Recommended Reading

Integration Teams for the Digital Era Must Support Multiple Delivery Models

Integration Maturity Model

Event-Driven Architecture

Analysis By: Yefim Natis, Gary Olliffe, Max van den Berk

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Event-driven architecture (EDA) is a style of system design where components communicate indirectly by passing event notification messages via an intermediary (an event broker). EDA is a long-standing architecture model. The increased demands of digital business, including cloud-native architectures, globally distributed computing, stream analytics and real-time decision making, have reintroduced EDA as newly relevant to IT leaders and technology providers and placed it back onto the Hype Cycle.

Why This Is Important

EDA provides advanced opportunities for scale, extensibility and resilience in applications through its asynchronous, intermediated, pub/subdesign model. Monitoring business and technical events in real time enables continuous analysis of the context for advanced intelligence in decision management.

Business Impact

An event-aware organization is more responsive in its ecosystem — more empathetic in its customer experience and more intelligent in its decision making — than a purely transaction-centric business. Competence in EDA accelerates the transition to the modern, continuously intelligent and innovating adaptive-scale business. Lacking event awareness, organizations may struggle to support business requirements at competitive speeds, agility, context awareness and cost-efficiency.

Drivers

- Digital business demands real-time context awareness through stream analytics to support intelligent business decisions. Applications that adopt EDA become sources of such context and empower their business decision makers.
- Application designers that seek high agility and scalability turn to EDA to implement more flexible, extensible and scalable applications and data communications.
- Growing interest in composable architecture brings organizations to EDA to maximize the autonomy and efficient use of its modular components (packaged business capabilities).
- The popularity of Apache Kafka is creating greater awareness of EDA among mainstream organizations and their software engineering leaders.
- Many major application vendors, including Salesforce and SAP, upgraded support of EDA to their applications and application platforms in recent years, enabling more intelligent and introspective monitoring of business processes.
- All cloud hyperscalers have added or upgraded their support for EDA by adding and extending their messaging and event brokering services.
- Application integration continues to gain adoption in mainstream organizations, and EDA is a popular model for strategic integration design.

Obstacles

- The lack of productivity and governance tools dedicated to EDA limits the design of EDA-based applications to more advanced engineering teams, and thus delays broader adoption.
- The diversity of protocol, message and API formats and standards for event processing limits adoption and increases implementation costs.
- The design principles of EDA are less well-understood by most development teams because of the complex trade-offs associated with asynchronous communication and the familiarity bias in favor of the common and ubiquitous request/reply model, often implemented using REST APIs.
- Event-driven communications can deliver only eventual consistency. Applications that require synchronization of distributed database updates must choose a different architecture.

User Recommendations

- Develop an inventory of the currently deployed EDA-related technologies (such as queuing, message and event brokers) and practices; the existing technologies are likely sufficient for some or most of early EDA initiatives.
- Build a strategic roadmap for full adoption of EDA into your standard set of skills and capabilities.
- Adopt EDA gradually, as the industry develops required standards, best practices and improved product design and management tools.
- Aim to establish EDA, along with request-driven service-oriented architecture (SOA), as the common and complementary architecture patterns, both considered for future application initiatives.
- Combine technologies and practices of EDA with event streaming and analytics to amplify the business value of the architecture and justify the essential initial investment.
- Work with business stakeholders to coordinate the discovery and analysis of business events; aim for synergy in business and technical modeling of event-driven solutions.

Sample Vendors

Amazon Web Services (AWS); Confluent; Google; IBM; Microsoft; Oracle; Salesforce; Solace

Gartner Recommended Reading

Using Event-Driven Integration With Enterprise Applications

Essential Patterns for Event-Driven and Streaming Architectures

Maturity Model for Event-Driven Architecture

Innovation Insight for Event Thinking

Sliding into the Trough

Cloud Event Brokers

Analysis By: Yefim Natis, Gary Olliffe, Max van den Berk, Keith Guttridge

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Event brokers play the role of an intermediary in event-driven architecture (EDA), administering the event topics, registering event publishers and subscribers, and facilitating the capture and distribution of event notifications. Cloud event brokers expand these capabilities with cloud-native characteristics of elastic scalability and multitenancy. They are optimized to support cloud-based applications and operations.

Why This Is Important

Most of the outcomes of digital business transformation depend, in part, on an organization's continuous awareness of relevant business events and its ability to respond in real time. Event brokers facilitate the monitoring and distribution of event notifications to application services for an automated response, dashboards for human action, or data stores for further analysis. Cloud event brokers bring these capabilities to cloud-based applications and operations.

Business Impact

Organizations aware of their ecosystem events are better prepared to manage unexpected interruptions and capitalize on opportunities in business moments. They are equipped to broadcast notable events for simultaneous, multitargeted responses. Event brokers enable organizations' versatility in monitoring multiple sources of events and communicating to many responders in parallel with strong scalability, integrity and resilience. Cloud event brokers extend this functionality to on-cloud operations.

Drivers

- Increased demand for real-time insights drives organizations to manage event streaming and stream analytics, leading, in turn, to event brokers for governance and coordination of event traffic.

- Increased adoption of Apache Kafka, by both businesses and leading technology vendors, promotes organizational awareness of the benefits and opportunities that event-driven application design brings.
- The migration of business applications to the cloud demands new platforms and communication infrastructure, driving many organizations to evaluate and adopt event broker services, paired with integration and API management offerings.
- The availability of multiple vendors' cloud event broker services, based on open-source standards such as Advanced Message Queuing Protocol (AMQP), Apache Pulsar, Apache Kafka, MQTT and NATS, provides competitive and differentiated options in event broker services for a better-tuned fit to customers' use cases.
- Increased maturity of cloud event brokers supports more advanced capabilities in performance, data and process management, and optimization of event-driven applications.
- Most leading SaaS offerings support some event processing, increasing awareness of the benefits and opportunities of event-driven application design in a large number of mainstream business and government organizations.
- The increasing popularity of digital integration hubs and data virtualization approaches deliver near-real-time data to applications by consuming event streams, instead of direct database lookups.
- Growing demand for advanced use patterns of "change cloud-data capture" drive new adoption of cloud event brokers.

Obstacles

- Cloud event broker offerings become too expensive as more proprietary features are added to help differentiate from the competition.
- Event broker functionality, embedded in some platform and application services, fragments control of event streaming across the organization, while delaying a systematic investment in event brokering.
- Some software engineering teams use webhook and WebSocket tools to set up event notifications, delaying the full many-to-many experience of EDA that's implemented via an event broker technology.

- Lack of universally supported standards for protocols or APIs for EDA implementation increases the costs and complexity of managing a large event-driven application infrastructure.
- Some organizations choose the request/reply communication where EDA would be a more effective model, just due to the unmitigated familiarity bias and lack of required skills.

User Recommendations

- Mix up the complementary strengths of the request/reply service-oriented architecture (SOA) and EDA, and encourage new projects to consider the combined use of both, as appropriate.
- Pilot experimental projects using cloud event brokers to gain insight and skills for upcoming, more advanced projects. Even a basic pub/submiddleware service is sufficient as a precursor for a full-featured event broker.
- Give preference to cloud event broker vendors demonstrating the understanding of the full life cycle of event processing functionality and responsibility.
- Plan for coordinated use of an event broker and an event stream processing (ESP) platform (also called stream analytics platform). The technologies are different but contribute to a shared business outcome.

Sample Vendors

Amazon Web Services (AWS); Confluent; Google; IBM; Microsoft; Solace; TIBCO Software

Gartner Recommended Reading

Using Event-Driven Integration With Enterprise Applications

Essential Patterns for Event-Driven and Streaming Architectures

Apply Event-Driven Approaches to Modernize IoT Data Integration

How to Identify Your Event-Driven Architecture Use Cases to Select the Best-Fit Event Broker

Maturity Model for Event-Driven Architecture

Digital Integration Hub

Analysis By: Andrew Humphreys

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

A digital integration hub (DIH) provides low-latency, high-throughput API- and event-based data access by aggregating and replicating multiple application sources into a data management layer that synchronizes with the applications via event-driven patterns. A DIH enables scalable, 24/7 data access, reduces workloads on the systems of record (SORs) and improves business agility. Organizations can reap additional value by using a DIH in analytics, data integration and composition scenarios.

Why This Is Important

Digital initiatives leverage APIs and, increasingly, events to unlock core business applications data and business logic. Traditional integration architectures can face severe performance, scalability and availability issues stemming from excessive workload generated in the SORs from new digital requests. A DIH is an increasingly popular alternative architecture to overcome these issues and delivers additional benefits that improve business agility.

Business Impact

DIHs can have the following business impacts:

- Provide digital application clients with a richer and more responsive experience than is achieved by accessing applications directly.
- Reduce application workload and limit the fees paid to SaaS providers for API consumption.
- Help enable 24/7 operations. The hub is not dependent on application availability.
- Improve business agility by decoupling the API layer from the applications.
- Maintain an up-to-date picture of fast-changing data used for analytics, notification services and data integration.

Drivers

- DIHs can help organizations lower operational costs by reducing the high workload generated in SORs by digital application requests.
- By enabling data to be stored locally, DIHs reduce the API-limit fees paid to SaaS providers, as well as the number of calls that need to be made to SaaS providers.
- DIHs deliver a more responsive and data-rich user experience than traditional integration architectures, leading to improved customer satisfaction.
- Using a DIH can accelerate the transition to composable business, digital and API economy by maintaining an up-to-date picture of fast-changing data needed to support a comprehensive set of APIs and events.
- DIHs give digital teams greater flexibility in building new offerings, by enabling them to access any data in the hub without needing to involve the SOR teams.
- Many vendors are introducing packaged DIH offerings, at times focused on specific use cases.

Obstacles

- Assembling and managing the varied set of DIH building blocks (API gateways, application platforms, integration platforms, event brokers, data management and metadata management tools) is very complex.
- Dealing with an architecture that is not well-known in the industry implies a scarcity of know-how, experience and skills, leading to high costs.
- Keeping the data management layer in sync with the systems of record by leveraging event-based integration tools (for example, change data capture) is challenging.
- Addressing the data governance issues derived from the creation of yet another copy or data structure outside the systems-of-record data is cumbersome.

User Recommendations

- Adopt a DIH-enabled architecture to provide a rich, responsive omnichannel experience for large audiences (hundreds of thousands of users or greater) and to reduce the cost associated with sustaining the digital initiative generated workload hitting the SORs.
- Enable API "pull" and event "push" services to access data scattered across multiple back-end systems.
- Decouple digital services from SORs to enable more flexible and composable business applications.
- Maintain an up-to-date "single source of truth" for fast-changing data, which can be used to provide additional services (for example, custom analytics or search) or analyzed in real time to detect "business moments."
- Embed DIH initiatives into the overall data hub strategy for governance and integration to avoid ending up with yet another data silo.

Sample Vendors

Cinchy; GigaSpaces, IBM; Informatica; Mia-Platform; Microsoft; Oracle; SAP; Sesam; Software AG

Gartner Recommended Reading

How Software Engineering Leaders Can Overcome the 4 Key Packaged Business Application Challenges

Essential Patterns for Data-, Event- and Application-Centric Integration and Composition

Micro Frontend

Analysis By: Anne Thomas

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

The benefits that the MFE pattern delivers to the front-end architecture are similar to those that microservices deliver to the back end. An MFE is a cohesive, independently deployable front-end module that runs within a larger front-end app. An MFE typically includes some logic for interacting with the back-end functionality. An MFE can be used in multiple front-end apps to enable consistent experiences.

Why This Is Important

The MFE pattern is critical for the future of web applications. It solves many of the challenges associated with building and maintaining large web applications (like an e-commerce site, for example) by splitting the code into smaller, simpler modules that can be delivered independently. The MFE pattern is often combined with the backend for frontend (BFF) pattern to enable easier reuse and to increase team autonomy.

Business Impact

The MFE pattern shortens the time to market of new application features because teams can deploy MFEs independently. Teams can work autonomously, which reduces friction and accelerates delivery. Each team is focused on their mission to deliver value for its distinct area of business, generating better experiences. MFE can also improve consistency and reduce redundancy because an MFE can be used in multiple apps. The trade-off for these benefits, though, is increased complexity.

Drivers

- **Independent releases and incremental upgrades:** The MFE pattern breaks a large application into loosely coupled modules that can be deployed independently. This release independence can protect the release schedule of the parent app from potential delivery delays caused by modules that are being built by a different team or a third party.
- **Autonomous teams:** The MFE pattern enables the work to be divided up and assigned to separate teams to increase scale and velocity. Each team can work independently with their own delivery schedules, and can also work with their preferred frameworks and tooling. An MFE often has a direct binding to a corresponding back-end component. The MFE pattern enables independent teams to each build their slice of the application from end to end.
- **Maintainable codebases:** MFEs are smaller and simpler than a monolithic web application, and each component is easier to maintain.
- **Modular reuse:** An MFE can be reused in multiple front-end applications, ensuring better consistency across multiple experiences.

Obstacles

- **DevOps prerequisites:** The value of MFEs depends on having a mature continuous deployment practice. Teams may get better outcomes by just focusing on improving modularity and maintainability rather than adopting a more complex architecture.
- **Inconsistent frameworks:** MFE frameworks help developers tie together MFEs into a working application. However, the tools use different approaches to support intermodule communications and shared state.
- **Increased complexity:** MFE frameworks work well for applications with obvious domain boundaries. As an application becomes more complex, however, managing state, navigation, routing and intermodule communications becomes more challenging, and the frameworks can hinder rather than help.
- **Governance:** Given the interdependencies created by this pattern, teams must clearly define what is provided by the container app versus the MFE components. Further, they must exercise strong discipline in managing and versioning the independent components.

User Recommendations

- Use the MFE pattern to enable independent delivery of modules and to isolate modules from each other and from the parent app. Don't use MFEs if you don't need this type of independent delivery.
- Ensure that teams have the knowledge and skills to deal with the increased complexity of the MFE pattern. Frameworks can simplify some of the tasks, but they don't make the complexity go away.
- Align MFEs with business-domain boundaries or workflows, and give MFE teams responsibility for building the back-end counterpart components.
- Start with coarse-grained components, and increase granularity only when the benefits of doing so outweigh the increase in complexity.
- Adopt design systems to ensure better consistency of the multiple MFEs in an application.
- Adopt an MFE framework to simplify state management, navigation, routing and intercomponent communications.

Sample Vendors

Bit; Luigi; Qiankun; single-spa; smapiot (Piral)

Gartner Recommended Reading

How to Make the Right Technology and Architecture Choices for Front-End Development

Adopt a Mesh App and Service Architecture to Power Your Digital Business

GraphQL APIs

Analysis By: Andrew Humphreys, Dave Micko

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

GraphQL enables API consumers to use a standard syntax to specify the data attributes they want, meaning they can select to receive only the information they need. This contrasts with a traditional REST API, where the structure of the data returned is predefined. Many GraphQL APIs, such as the Salesforce GraphQL API, provide an abstraction over multiple distinct data sources and APIs. This way, they simplify retrieving data from multiple sources.

Why This Is Important

GraphQL enables flexible self-service access to data for front-end applications. It optimizes data movement between remote clients and back-end services by avoiding over- and underfetching issues. GraphQL also enables organizations to model, expose and use valuable data or metadata associated with important relationships between data entities. Organizations including Netflix and Airbnb are adopting GraphQL as a means of composing a large graph schema from independently managed subgraphs.

Business Impact

GraphQL enhances user experience development. It is a more efficient use of resources, compared to the overhead of individual REST API calls. Further, it enables consumers to explore and unlock data based on their individual needs, rather than the provider-defined REST APIs resource model. A GraphQL schema can also become a shared data model that is decoupled from the physical data and interface models of individual services. This can help establish a common understanding of business data.

Drivers

- Web developers — particularly ReactJS developers — are driving demand for GraphQL to enable self-service access to data and services. GraphQL allows API consumers to define the structure for the responses they require from APIs, enabling greater flexibility than REST APIs.
- GraphQL reduces the need for multiple API calls to access all the data a developer requires, and reduces the amount of unnecessary data returned in an API call.
- Vendors now provide GraphQL APIs that enable data access across multiple applications. API management products are increasingly adding GraphQL support, simplifying the adoption of GraphQL APIs.

Obstacles

- GraphQL 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 as widespread as support for REST APIs. Security, governance and performance optimization practices are still maturing.
- Performance is a consideration due to the processing required to process a query and efficiently retrieve the data specified. This may require multiple back-end API requests or database queries.
- The availability of design and development skills for GraphQL APIs is currently behind the availability of skills for delivering REST APIs. For example, error handling in GraphQL is different and may add complexity the developer needs to plan for.

User Recommendations

- Evaluate GraphQL APIs when they are available from their platform providers, and adopt them to provide flexible self-service access to data, including for applications such as dashboards and mobile apps.
- Consider building GraphQL APIs to meet specific demand from (mostly front-end) developers. For example, demand could include situations where multiple consumers need different data payloads, and when consumers need to span multiple API's.
- Replace dedicated "experience APIs" or backends for frontends (BFFs) with GraphQL APIs that tailor API experiences to different API consumers. GraphQL APIs offer the consumer more flexibility and control over how related data is accessed than traditional APIs.
- Adopt a community of practice approach, as well as mentoring, "Golden Paths" and guidelines on internal developer portals. These initiatives will help spread GraphQL skills in your organization, as GraphQL APIs introduce new challenges and ways of thinking for both API consumers and API providers.

Sample Vendors

Apollo GraphQL; AWS (AppSync); Hasura; Hygraph; IBM (StepZen); RapidAPI; Tyk; WSO2

Gartner Recommended Reading

Choosing an API Format: REST Using OpenAPI Specification, GraphQL, gRPC or AsyncAPI

Data Fabric

Analysis By: Mark Beyer, Ehtisham Zaidi, Roxane Edjlali, Sharat Menon, Robert Thanaraj

Benefit Rating: Transformational

Market Penetration: 1% to 5% of target audience

Maturity: Adolescent

Definition:

A data fabric is a design framework for attaining flexible and reusable data pipelines, services and semantics. The fabric leverages data integration, active metadata, knowledge graphs, profiling, ML and data cataloging. Fabric overturns the dominant approach to data management which is "build to suit" for data and use cases and replaces it with "observe and leverage."

Why This Is Important

Data fabric leverages traditional approaches while enabling the enterprise to adopt technology advances and avoids "rip and replace." It capitalizes on sunk costs and simultaneously provides prioritization and cost control guidance for new spending for data management. It leverages concepts and existing platforms/tools or implementation approaches. It offers flexibility, scalability and extensibility in infrastructure for humans or machines to assure data is consumable across multiple use and reuse cases on-premises, multicloud or hybrid deployments.

Business Impact

Data fabric:

- Increases identification, deployment and availability of data for reuse at scale.
- Provides insights to data engineers by standardizing repeatable integration tasks, improving quality, and more.
- Adds semantic knowledge for context and meaning, and enriched data models.
- Evolves into a self-learning model that recognizes similar data content regardless of form and structure, enabling connectivity to new assets.
- Enables observability across the data ecosystem.

- Reduces maintenance, support and optimization costs associated with managing data.

Drivers

- The dearth of new staffing or personnel seeking data management roles and the attrition of experienced professionals leaving the practice area has increased the demand for more efficient data reuse.
- Demand for rapid comprehension of new data assets has risen sharply and continues to accelerate, regardless of the deployed structure and format.
- Increased demand for data tracking, auditing, monitoring, reporting and evaluating use and utilization, and data analysis for content, values and veracity of data assets in a business unit, department or organization.
- Catalogs alone are insufficient in assisting with data self-service. Data fabrics capitalize on machine learning (ML) to provide recommendations for integration design and delivery, reducing the amount of manual human labor that is required.
- Significant growth in demand and utilization of knowledge graphs of linked data, as well as ML algorithms, can be supported in a data fabric to assist with graph data modeling capabilities and use-case generic semantics.
- Organizations have found that one or two approaches to data acquisition and integration are insufficient. Data fabrics provide capabilities to deliver integrated data through a broad range of combined data delivery styles including bulk/batch (ETL), data virtualization, message queues, use of APIs, microservices and more.

Obstacles

- Organizations will keep applying budget or staff to one-off and point-to-point integration solutions.
- Differing design and semantic standards used by various vendors to document and share metadata create challenges in its integration and effective analysis to support a data fabric design.
- Fabric needs analytic and ML capabilities to infer missing metadata. This will be error-prone at first with staffing and resources assigned to competing demands in advanced analytics, data science and AI near the data consumption layer.
- Active metadata management practices lag behind data fabric adoption but are critical to its implementation.
- Diverse skills and platforms demand a cultural and organizational change from data management based upon analysis, requirements and "design then build" to discovery, response and recommendation based upon "observability and leveraging."
- Improper split from data mesh implies choosing one approach over another and not a complementary relationship.
- Inexperience in reconciling a data fabric with legacy data and analytics governance programs will confound implementers.

User Recommendations

- “Active metadata” and leveraging the inherent practices to it is mandatory in a data fabric (covered separately).
- Invest in an augmented data catalog that permits multiple ontologies over top of business data taxonomies and is alerted to new use cases for data and the related business units utilizing data.
- Deploy data fabrics that populate and utilize knowledge graphs in targeted areas where adequate metadata and metadata management practices already exist.
- Ensure business process experts can support the fabric by enriching knowledge graph capabilities with business semantics.
- Evaluate all existing data management tools to determine the availability of three classes of metadata: design/run, administration/deployment and optimization/algorithmic metadata. When adopting new tools, favor those that share the most metadata.
- Do not permit SaaS solutions to isolate their metadata from access by PaaS solutions that orchestrate across solutions.

Sample Vendors

Cambridge Semantics; Cinchy; CluedIn; Denodo Technologies; IBM; Informatica; Semantic Web Company; Stardog; Talend

Gartner Recommended Reading

Data and Analytics Essentials: How to Define, Build and Operationalize a Data Fabric

Quick Answer: What Is Data Fabric Design?

Emerging Technologies: Critical Insights on Data Fabric

Hybrid Integration Capability Framework

Analysis By: Shrey Pasricha

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

The hybrid integration capability framework (HICF) defines a logical framework to amass integration and governance capabilities across an organization. It enables organizations to tackle multiple integration use cases across four dimensions — personas, endpoints, deployment model and domains. It defines how an organization can approach integration, assembling multiple integration tools from one or more providers and managing them as a cohesive, federated and integrated whole.

Why This Is Important

Organizations pursuing digital and composable business initiatives find that the integrations they need to build are required to:

- Connect systems and services residing on different endpoints.
- Implement a myriad of use cases and integration patterns.
- Enable delivery by various integrator personas.
- Deploy to different runtime environments.

The HICF helps software engineering leaders responsible for integration architecture to organize their portfolio to support a modern integration strategy.

Business Impact

Implementation will differ to reflect specific requirements, but in all cases, HICF alleviates any integration challenges by:

- Providing a reference model to match desired integration capabilities to evolving business needs.
- Reducing unnecessary duplication, diversity and expenditure in the integration portfolio.
- Accelerating time to value for integration-intensive business initiatives and putting an organization on the path to decentralized, self-service integration delivery.

Drivers

Organizations generally have an array of integration tools and scattered platforms which are not governed or managed as a logical whole. These tools and platforms include on-premises and cloud-delivered integration platforms, API management platforms, event brokers, metadata management tools, open-source integration frameworks, SaaS-embedded integration capabilities and other use-case-specific components — often from different providers. The HICF provides a way to organize and manage these multiple products and use cases in order to:

- Enable a range of diverse integration personas to perform integration work in a self-service fashion. These personas include integration specialists (professional integration developers), "ad hoc" integrators (SaaS administrators and business technologists who occasionally have to perform integration work) and citizen integrators (business users who want to automate personal or workgroup processes).

- Integrate a wide variety of endpoints residing in cloud environments, on-premises data centers, ecosystem partners, and mobile and Internet of Things (IoT) devices by using APIs, events and batch mechanisms.

- Support a differentiated set of use cases, including but not limited to application, data, B2B, process, IoT, API and event integration, robotic process automation, and digital integration hubs.

- Deploy integration platform capabilities in a hybrid, multicloud scenario — that is, one featuring a combination of public and private clouds and on-premises data centers — and embed them in applications and edge systems.

- Augment the organization's integration landscape by supporting delivery of SaaS-embedded integrations as well as integration as code delivered via software engineers through programming languages, open-source integration frameworks and serverless functions.

Obstacles

- Organizations fail to consider dimensions like endpoints, personas and use cases, and solely focus on deployment models to select a single integration platform as they confuse the “hybrid” in “HICF” with hybrid deployment models.
- A plethora of technology providers have released integrated technology stacks mirroring the HICF; however, a hybrid integration implementation often requires the aggregation of multiple products from different providers. Such a technology aggregation poses governance and operational challenges.
- The use of a wide range of tools leads to suboptimal outcomes and skills duplication. However, implementing a single, cross-product “control plane” requires notable investments in technologies and skills.
- HICF is a logical framework which is a precursor to enable self-service integration by a variety of organizational units which can lead to chaotic duplication of efforts and high costs in the absence of well-defined governance policies.

User Recommendations

- Map your organization’s integration capabilities to the HICF by aggregating multiple tools and approaches instead of finding one tool, then unify them under a common strategy and governance approach.
- Federate different vendors’ products instead of buying an out-of-the-box HICF-inspired platform to build target capabilities. This approach works best for large organizations as it makes it easier to maintain backward compatibility with in-place integration platforms and mitigate the risk of single-vendor lock-in.
- Adopt an iPaaS, whenever possible, to reduce the complexity of effort in building target capabilities using HICF. This approach works best for midsize organizations as an iPaaS provides a subset of the HICF capabilities that is generally sufficient for such organizations.
- Build your target HICF-inspired platform by adopting a stepwise, business initiative driven strategy, which is much easier to justify than a “big-bang” approach and reduces complexity and risk.

Sample Vendors

Boomi; IBM; Jitterbit; Microsoft; MuleSoft; Oracle; SAP; SnapLogic; Software AG; TIBCO Software

Gartner Recommended Reading

How to Select the Right Mix of Integration Technologies

Integration Maturity Model

Data Hub iPaaS

Analysis By: Keith Guttridge

Benefit Rating: High

Market Penetration: 1% to 5% of target audience

Maturity: Adolescent

Definition:

Data hub integration platform as a service (iPaaS) supports integration between applications and system endpoints via a centralized intermediary store that persists the (often-normalized) data before delivery to the destination. This is different from the pass-through architecture that most established iPaaS offerings utilize today and provides additional information management capabilities at the data store level.

Why This Is Important

The data landscape for most organizations is fragmenting across on-premises applications and data stores, and an ever-increasing number of SaaS applications and cloud data stores. This is causing complex integration and governance challenges. Some iPaaS vendors include data stores to create standardized data models that are not tightly coupled to vendor data models and expose APIs to accelerate development of newer services.

Business Impact

Data hub iPaaS provides these benefits:

- Simplification of integrating applications and data sources.
- Improved data management and data governance.

- Improved resilience of the production environment with record/replay capabilities for integration errors and system availability.
- Simplified access to the centralized data model via APIs, instead of connecting directly to application APIs.
- Real time analytics to gain business insight, and potentially enable real-time and contextual decision support.

Drivers

- Organizations looking to improve data management across on-premises and cloud-based applications and data sources.
- Organizations looking to build a customer engagement hub.
- Organizations looking to build a digital integration hub.
- Organizations looking to reduce the number of vendors providing integration and data management technologies.
- iPaaS vendors converging various integration technologies.

Obstacles

- Regional compliance policies for data stores.
- Industry compliance policies for data stores.
- Organizations' compliance policies for data stores.
- Preference for best-of-breed integration and data management technology.
- Organizational structure impeding unified approach to integration and data management.
- Vendor landscape made up mostly of small startups with only a handful of large vendors providing this service.

User Recommendations

- Acknowledge that this is a relatively new market. The few vendors that provide this capability often do so for relatively niche use cases. It may take several years before data hub iPaaS becomes general-purpose enough for most clients. Since the data is stored within the data hub iPaaS, this brings with it extra challenges, such as security, resilience and compliance, that regular iPaaS vendors do not have to worry about.
- Combine offerings from several technology categories and vendors (such as iPaaS plus data store plus analytics), if the current offerings in the data hub iPaaS market are not suitable for your needs. Once established, however, the combination of iPaaS, data management, real-time analytics and machine learning has the potential to significantly disrupt how organizations integrate their application and data portfolios, as well as their B2B partners.

Sample Vendors

Cinchy; Domo; IBM, Informatica; K2view; OVHcloud (ForePaaS); SAP; Sesam

Gartner Recommended Reading

Magic Quadrant for Integration Platform as a Service, Worldwide

Magic Quadrant for Data Integration Tools

Service Mesh

Analysis By: Anne Thomas

Benefit Rating: Low

Market Penetration: 1% to 5% of target audience

Maturity: Adolescent

Definition:

A service mesh is a distributed computing middleware that manages communications between application services — typically within managed container systems. It provides lightweight mediation for service-to-service communications and supports functions such as authentication, authorization, encryption, service discovery, request routing, load balancing, self-healing recovery and service instrumentation.

Why This Is Important

A service mesh is lightweight middleware for managing and monitoring service-to-service (east-west) communications — especially among microservices running in container management systems, such as Kubernetes. It provides visibility into service interactions, enabling proactive operations and faster diagnostics. It automates complex communication concerns, thereby improving developer productivity and ensuring that certain standards and policies are enforced consistently across applications.

Business Impact

Service mesh is one of many management technologies that provide software infrastructure for distributed applications. Service meshes are most often used with services deployed in container management systems, such as Kubernetes. This type of middleware, along with other management and security middleware, helps provide a stable environment that supports “Day 2” operations of containerized workloads. However, the technology is complex and often unnecessary for smaller deployments.

Drivers

- Microservices and containers: Service mesh adoption is closely aligned with microservices architectures and container management systems like Kubernetes. Service mesh supports useful functionality in ephemeral environments, such as dynamic service discovery and mutual Transport Layer Security (mTLS) between services.
- Observability: As microservice deployments scale and grow more complex, DevOps teams need better ways to track operations, anticipate problems and trace errors. Service mesh automatically instruments the services and feeds logs to visualization dashboards.
- Resilience: A service mesh implements the various communication stability patterns (including retries, circuit breakers and bulkheads) that enable applications to be more self-healing.
- Bundled feature: Many container management systems now include a service mesh, inspiring DevOps teams to use it. The hyperscale cloud vendors provide a service mesh that is also integrated with their other cloud-native services.
- Federation: Independent vendors such as Buoyant, greymatter.io, HashiCorp, Kong and Solo provide service meshes that support multiple environments.

Obstacles

- Not necessary: Service mesh technology can be useful when deploying microservices in Kubernetes, but it's never required.
- Complexity: It's complex to use and administer, and there are increasing discussions on why not to use a service mesh in technology discussion groups and social media.
- Redundant functionality: Users are confused by the overlap in functionality among service meshes, ingress controllers, API gateways and other API proxies. Management and interoperability among these technologies is still nascent within the vendor community.
- Overhead: Service mesh technology consumes resources and typically adds overhead to the interactions it manages. Some vendors now support alternate architectures, such as a shared-agent model to reduce overhead, but this solution reduces the observability benefits.
- Competition with "free": Independent service mesh solutions face challenges from the availability of platform-integrated service meshes from the major cloud and platform providers.

User Recommendations

- Determine whether the value you might get from a service mesh in terms of improved security or observability is worth the increase in complexity and administration of the service mesh. A service mesh becomes more valuable as the number of service-to-service (east-west) interactions increases.
- Favor the service meshes that come integrated with your container management system unless you have a requirement to support a federated model.
- Reduce cross-team friction by assigning service mesh ownership to a cross-functional platform engineering team that solicits input and collaborates with networking, security and development teams.
- Accelerate knowledge transfer and consistent application of security policies by collaborating with I&O and security teams that manage existing API gateways and application delivery controllers.

Sample Vendors

Amazon Web Services; Ambient Mesh; Buoyant; Google; HashiCorp; Istio; Kong; Microsoft; Solo.io

Gartner Recommended Reading

How a Service Mesh Fits Into Your API Mediation Strategy

Digital Integrator Technologies

Analysis By: Keith Guttridge

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Digital integrator technologies apply AI techniques to assist integration design and delivery, as well as optimize production performance and availability. These technologies focus on areas such as engagement via chatbots or voice; automation via process selection, next best action and intelligent data mapping to assist in integration development; and insight for processing optimization, intelligent platform operations and competitive analysis.

Why This Is Important

Digital integrator technologies are designed to simplify integration development. By anticipating user needs and making next-best-step recommendations for designing an integration flow, inference algorithms identify suitable prepackaged integration content, help rectify errors in flow and improve performance. Advanced digital integrator technologies dynamically optimize integration processing and platform operations, with capabilities to autoadjust runtime, auditing and self-healing.

Business Impact

The business impacts are:

- AI-enabled integration platforms provide automated guidance for integrating applications and data, simplifying tasks for integration specialists and improving productivity.
- Initiatives to modernize integration platforms using AI can adopt a low-code or no-code paradigm to increase adoption among business technologists.

- AI-assisted insight of the runtime environment can improve availability, identify process improvement and provide competitive market insights.

Drivers

- Delivery of integration is becoming pervasive, rather than just a specialist task. Digital integrator technologies empower a broad range of integration specialists and business technologists. This advances the ideas of democratizing integration and enabling composable business.
- Increased adoption of conversational user experiences is driving demand for connectivity to applications and data sources, producing the ability to create integration processes on demand or query the operational state of the integration platform.
- Increasing adoption of iPaaS has resulted in vendors gaining greater insight into how their tens of thousands of clients use their technologies via metadata. This, in turn, enables them to assist their clients in getting value from their offerings.
- Rapid improvement in generative AI has massively raised awareness and improved capabilities, such as communication in natural language to simplify the builder experience, data mapping between schemas, and dynamic generation of processes and tasks.

Obstacles

- Governance challenges reign when there is little or no availability of comprehensive lineage/metadata management capabilities that track the activities and outcome of data mapping or process selection. It may be difficult to ensure the traceability of integration flows or avoid potentially substantial, consequential damages created by flawed next best steps that are guided by flawed data.
- Experiences learned from AI come from the metadata generated by the huge variety of integrators building integrations, as well as the vast number of integrations run on the platform. This could misdirect the recommendation engine by highlighting poor design practices that become popular through overuse by nonspecialists.

User Recommendations

- Evaluate the AI capabilities of products against your most common and simplest integration scenarios to assess their accuracy and productivity benefits, before making them available to business technologists.
- Manage expectations by making it clear that digital integrator technology will only help with the most common integration scenarios for the leading applications and data sources. Complex integrations and data structures will see little benefit and still require integration specialists with the current generation of digital integrator technologies.
- Ensure integrations are fully managed by planning for security, monitoring, auditing, reporting and life cycle management. Digital Integrator technology only helps with the building and testing of integrations.

Sample Vendors

Boomi; IBM; Informatica; Microsoft; Oracle; SAP; SnapLogic; TIBCO Software; Tray.io; Workato

Packaged Integrating Processes

Analysis By: Andrew Comes

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Packaged integrating processes (PIPs) are predefined integration solutions to connect, automate and standardize business processes across multiple endpoints. Examples include PIPs for revenue operations (RevOps), hire-to-retire and procure-to-pay. Multiple integration platforms, SaaS providers and service providers supply a portfolio of PIPs to support ERP, human capital management (HCM), customer experience (CX) and other integration requirements for both cloud and on-premises applications.

Why This Is Important

In common business scenarios, PIPs deliver integrations faster by providing near-to-complete, out-of-the-box solutions to connect systems. However, PIPs are not plug-and-play solutions and may require customization to work. PIPs help reduce time to value for standard "commodity" integration scenarios, such as connecting ERP systems with sales tools, increasing productivity in integration delivery and facilitating distributed self-service integration.

Business Impact

The concept of PIPs is often known in the market as "accelerators" or "recipes" for integration between applications. While they have been in the market for some time, PIPs are now becoming popular because of the prevalence of SaaS offerings. PIPs enable ad hoc and citizen integrators, and integration specialists to deliver integration, thereby creating an empowered, self-service model of distributed integration delivery and helping achieve higher levels of integration maturity.

Drivers

- Recently, the total number of vendors providing PIPs has steadily increased, driven by customers' adoption of SaaS offerings. Examples of PIPs include SAP S/4HANA integration with Salesforce, NetSuite integration with Shopify and many more.
- A key factor driving the PIPs hype — and, therefore, promoting its use — is the growing customer demand for very fast integration, which is shifting the integration development toward business technologists and citizen integrators.
- PIPs offerings are especially attractive to midsize and large organizations with limited IT skills and that cannot handle overly complex integration requirements. Many of these organizations see PIPs as a way to rapidly deliver integrations without investing in new skills.

Obstacles

- Some of the PIPs provided by vendors pose a risk of vendor lock-in, which can negatively affect the integration strategy by limiting the ability to extend or optimize the solution.
- Lack of flexibility of PIPs leads to rigid integrations. This may be perfectly acceptable for nondifferentiating, commodity use cases if the intention is to integrate the same solution as everyone else.
- Some of the PIPs provided, especially those not provided by integration platforms, have limited capabilities for monitoring, making it difficult to diagnose and fix anomalies, thus adding technical debt.

User Recommendations

When trying to automate the integration of common, non-differentiating business processes, use PIPs to reduce implementation costs and accelerate time to value. Therefore, software engineering leaders responsible for integration should:

- Identify business application integration processes in the backlog that are nondifferentiating and can be quickly and inexpensively implemented via PIPs.
- Empower new integration personas, such as business technologists and citizen integrators, to take on responsibility for integration by providing them with approved PIPs they can customize and deploy.
- Use existing technical personas to shift workload away from PIPs to further improve uptake.
- Test your ability to implement PIPs efficiently and effectively by performing a proof of concept (POC) for each PIP identified for implementation.

Sample Vendors

Boomi; Celigo; Jitterbit; MuleSoft; Oracle; SAP; SnapLogic; Tray.io; Workato; Zapier

Gartner Recommended Reading

Accelerate Your Integration Delivery by Using Packaged Integration Processes

Choosing Application Integration Platform Technology

Magic Quadrant for Integration Platform as a Service, Worldwide 2023

Climbing the Slope

Microservices

Analysis By: Anne Thomas

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

A microservice is a tightly scoped, highly cohesive, strongly encapsulated, loosely coupled, and independently deployable and scalable software component. Microservices architecture (MSA) applies the principles of service-oriented architecture (SOA), DevOps and domain-driven design to the delivery of distributed applications. MSA has four core objectives: increasing agility, enabling autonomous teams, improving deployment flexibility and supporting precise scalability.

Why This Is Important

Microservices architecture enables autonomous teams and improves agility, flexibility, resilience, and precise scalability. It is a way to build cloud-native applications, and it facilitates continuous delivery practices. However, the architecture is complex, with disruptive cultural and technical impacts. Software engineering teams have been observed to use MSA indiscriminately, leading to overly complex architectures that fail to deliver anticipated benefits and often make things worse.

Business Impact

MSA increases business agility by enabling teams to isolate new feature deployments and regression impacts in their software products, in response to changing business requirements. It improves the scalability of the software engineering organization by enabling small teams to work autonomously to deliver different parts of an application. MSA allows teams to deploy a single feature without the delay, cost and risk of deploying the entire application.

Drivers

- **Continuous delivery:** Software engineering teams adopt MSA to facilitate a continuous delivery practice. The architecture must be combined with robust agile methods and strong DevOps practices to enable teams to safely develop and deploy small, independent features to production systems at the frequency at which they are delivered.
- **Autonomous teams:** When applied properly, MSA increases the independence of different parts of a large application. This enables multiple development teams to work autonomously and on their own schedules.
- **Cloud-native architecture:** MSA facilitates the building of cloud-native applications that support robust scalability and resiliency requirements.
- **Containers:** Microservices are frequently deployed in container management systems, which enable improved automation of delivery pipelines. Container management systems can also dynamically scale service instances in response to load requirements, and automatically recover services that have failed.
- **Resiliency:** When combined with resiliency patterns and practices, MSA enables the creation of self-healing systems that can continue to operate through partial outages.

Obstacles

- MSA and its benefits are often misunderstood, and teams may struggle to deliver outcomes that meet senior management expectations. Microservices should not be shared and they will not save you money.
- MSA delivers the most value when combined with continuous delivery practices. Management may be disappointed with the MSA cost-benefit equation, if teams are not doing continuous delivery.
- MSA is complex. Developers must acquire new skills, and adopt new design patterns and practices to achieve its benefits. Additional complexity also impedes incident recovery.
- MSA disrupts traditional data management models. Data consistency must be managed in service and interface design, rather than in data stores.
- MSA requires new infrastructure that increases the cognitive load on engineering teams.

- Many software engineering leaders underestimate the cultural prerequisites, such as mature agile and DevOps practices, and team structures aligned with service domains.

User Recommendations

- Set clear expectations by defining business goals for MSA adoption, based on realistic cost-benefit analysis.
- Improve outcomes by creating guidelines for where and when software engineering teams should and should not use MSA.
- Create platform teams to manage the development and runtime platforms that support microservices. This would reduce the cognitive load on developers and automate delivery of new microservices.
- Ensure that application architecture is as simple as possible to achieve your goals.
- Use MSA where appropriate to attain those goals. View microservices as a tool, not as a target architecture.
- Avoid MSA for small, simple, low-volume or static applications, where the cost of the increased complexity exceeds the benefits of improved agility, flexibility, or scalability.
- 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

Solution Path for Applying Microservices Architecture Principles

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

Backend for Frontend

Analysis By: Anne Thomas

Benefit Rating: Moderate

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

The backend for frontend (BFF) pattern prescribes creating a dedicated back-end API for each front-end experience. Each BFF provides an optimized API that supports the specific needs of its associated front-end experience, and it encapsulates any orchestration or navigation of back-end services required to support the front-end's workflow. It may also provide app-specific data or logic.

Why This Is Important

The BFF pattern is a popular pattern for enabling multiexperience, which provides optimized experiences for different personas interfacing through different types of devices and modalities. The BFF pattern is often combined with the micro frontend (MFE) pattern to more easily enable reuse and increase team autonomy. These patterns provide more flexibility, higher optimization and a better developer experience than using a single API to support multiple experiences.

Business Impact

The BFF pattern gives front end teams control over their APIs and reduces risks associated with shared APIs, such as dependency delays and unintended side effects from backend changes. It simplifies front-end development and enables faster iterations, thereby accelerating delivery of new features. It can improve app performance by optimizing network traffic between front ends and back ends, and it can improve user satisfaction by simplifying the application's workflow.

Drivers

- **Multiexperience:** Product teams frequently need to implement multiple front-end experiences for an application to support different types of devices and modalities, as well as different user personas. Each user experience often requires unique functionality, which places a burden on the back-end services to support multiple data models and workflows. The BFF pattern creates a dedicated API for each front end.
- **Simplicity:** Product teams often implement back-end functionality as a suite of services, which places a burden on front-end developers to implement complex workflow and navigation logic to invoke the appropriate services in the correct sequence to accomplish a task. The BFF pattern shifts the complex logic and workflow to a BFF service, thereby simplifying the front end.
- **Performance:** Product teams may have requirements to reduce network traffic for specific front ends, such as smartphones or watches. The BFF pattern enables product teams to design optimized APIs that can reduce network traffic and latency and improve overall app performance. Also, by transferring the orchestration logic and workflow to a BFF service, the front end can accomplish its work with fewer API calls.
- **Team autonomy:** Front-end components are dependent on back-end components, and when those components are developed by different product teams, diverging team schedules can cause frustrating delays. When using the BFF pattern, front-end product teams typically are also responsible for building their BFF APIs and associated services, which alleviates many, but not all, scheduling challenges.

Obstacles

Every design pattern involves trade-offs that discourage developers from adopting it. The BFF trade-offs include:

- **API proliferation:** Building a separate API for every front end will result in a huge number of APIs and services, each of which must be secured, tracked and managed.
- **Duplication:** BFF services for a particular application often contain similar and redundant logic and back-end service aggregation functionality.
- **Overloading:** In an effort to avoid duplication, product teams may try to pack more functionality than appropriate into a BFF service, or they may shift some of the functionality to the front end.

User Recommendations

- Build separate BFF APIs to support front ends that have significantly different requirements, although also consider alternative patterns and technologies, such as GraphQL, which can efficiently support multiple front ends using a single back-end service.
- Combine the BFF pattern with the MFE pattern for complex front-end apps to enable more agility via independent deployments of new functionality, although beware of the increase in complexity resulting from these patterns.
- Avoid creating a separate BFF service for every front end to mitigate API proliferation. When two or more BFF APIs expose essentially the same functionality, those front ends should be able to share a single API.

Gartner Recommended Reading

How to Make the Right Technology and Architecture Choices for Front-End Development

Adopt a Mesh App and Service Architecture to Power Your Digital Business

Cloud-Native Architecture

Analysis By: Anne Thomas

Benefit Rating: Moderate

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

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 modernize their applications and migrate to cloud and cloud-like infrastructures. Cloud-native principles and patterns enable applications to operate efficiently in a dynamic environment and make the most of cloud benefits. Organizations adopting cloud, containers and serverless infrastructures must apply these principles to ensure their applications perform well, consume resources efficiently and handle failures gracefully.

Business Impact

Cloud-native architecture has the following business impacts:

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

Drivers

- **Cloud adoption:** Organizations want to make the most of cloud platforms, including serverless platforms and container-based systems, to support their digital business initiatives, but they can't fully exploit cloud benefits without cloud-native architecture (summarized as LIFESPAR).
- **DevOps automation:** Software engineering teams are adopting cloud-native architecture to support cloud-native DevOps automation. A basic set of rules known as the " twelve-factor app" was first published in 2011. It is still a good summary of basic practices that support cloud-native DevOps, although many teams have moved beyond this foundation and are using cloud-native architecture to support continuous integration/continuous delivery (CI/CD).
- **Modern architecture practices:** Cloud-native architecture complements modern architecture practices such as application decomposition (following the mesh app and service architecture [MASA] structure and microservices architecture), containerization, configuration as code, and stateless services.

Obstacles

- **Application renovation:** Many existing applications require significant revisions to convert them to cloud-native architecture.
- **Complexity:** Cloud-native architecture adds a level of complexity to applications, and development teams require new skills, new frameworks and new technology to be successful.
- **Skills gap:** 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.
- **Applicability:** 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

- Follow LIFESPAR architecture principles when building cloud-native applications. Adhere to the twelve-factor-app rules to ensure that the application conforms to basic DevOps practices.
- Incorporate basic 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.
- Apply cloud-native design principles as you modernize legacy applications that will be deployed on a cloud platform to ensure that they can tolerate ephemeral or unreliable infrastructure.
- Select an application platform that matches your cloud-native architecture's maturity and priorities. Low-code platforms enable rapid development of cloud-ready applications, but they don't fully apply LIFESPAR and twelve-factor principles. Container platforms require cloud-friendly architecture. Serverless platforms require a stricter adherence to the LIFESPAR principles.

Sample Vendors

Alibaba Cloud; Amazon Web Services (AWS); Google; Microsoft; Red Hat; Salesforce; VMware

Gartner Recommended Reading

A CTO's Guide to Cloud-Native: Answering the Top 10 FAQs

8 Architectural Principles for Going Cloud-Native

How to Modernize Your Application to Use Cloud-Native Architecture

Essential Skills for Cloud-Native Application Architects

9 Principles for Improving Cloud Resilience

Event Stream Processing

Analysis By: W. Roy Schulte, Pieter den Hamer

Benefit Rating: Transformational

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

Event stream processing (ESP) is computing that is performed on streaming data (sequences of event objects) for the purpose of stream analytics or stream data integration. ESP is typically applied to data as it arrives (data "in motion"). It enables situation awareness and near-real-time responses to threats and opportunities as they emerge, or it stores data streams for use in subsequent applications.

Why This Is Important

ESP enables continuous intelligence and real-time aspects of digital business. ESP's data-in-motion architecture is a radical alternative to the conventional data-at-rest approaches that have historically dominated computing. ESP platforms have progressed from niche innovation to proven technology, and now reach into the early majority of users. ESP will reach the Plateau of Productivity in less than two years and eventually be adopted by multiple departments within every large company.

Business Impact

ESP transformed financial markets and became essential to telecommunications networks, smart electrical grids, and some IoT, supply chain, fleet management and other transportation operations. However, most of the growth in ESP during the next 10 years will come from areas where it is already established, especially IoT and customer engagement. Stream analytics from ESP platforms provide situation awareness through dashboards and alerts, and detect anomalies and other significant patterns.

Drivers

Six factors are driving ESP growth:

- Organizations have access to ever-increasing amounts of low-cost streaming data from sensors, machines, smartphones, corporate websites, transactional applications, social computing platforms, news and weather feeds, and other data brokers. Many new AI and other analytical applications need this streaming data to satisfy business requirements for situation awareness and faster, more-accurate decisions.
- The wide use of Apache Kafka and similar streaming messaging systems is reducing the cost and complexity of ingesting, storing and using streaming data.
- Conventional data engineering pipelines take hours or days to prepare data for use in BI and analytics, causing delays that are unacceptable for some purposes. Therefore, an increasing number of data engineering pipelines are being reimplemented as real-time data flows (continuous ETL) in ESP platform products or stream data integration tools with embedded ESP. These real-time data flows filter, aggregate, enrich, and perform pattern detection and other transformations on streaming data as it arrives.
- ESP products have become widely available, in part because open-source ESP technology has made it less expensive for more vendors to offer ESP. More than 30 ESP platforms or cloud ESP services are available. All software megavendors offer at least one ESP product, and numerous small-to-midsize specialists also compete in this market. Cloud ESP platforms have lowered the cost of entry.
- Vendors are embedding ESP platforms into a wide variety of other software products, including industrial IoT platforms, stream data integration tools, unified real-time platforms (aka continuous intelligence platforms), insider threat detection tools and AI operations platforms.

- Vendors are adding highly productive development tools that enable faster ESP application development. Power users can build some kinds of ESP applications via low-code techniques and off-the-shelf templates.

Obstacles

- ESP platforms are overkill for many applications that process low volumes of streaming data (i.e., under 1,000 events per second), or that do not require fast response times (i.e., less than a minute). Conventional BI and analytics tools with data-at-rest architectures are appropriate for most stream analytics with these less-demanding requirements.
- Many architects and software engineers are still unfamiliar with the design techniques that enable ESP on data in motion. They are more familiar with processing data at rest in databases and other data stores, so they use those techniques by default unless business requirements force them to use ESP.
- Some streaming applications are better-implemented on unified real-time platforms that process both data in motion and data at rest. Some unified platforms use embedded open-source ESP platform products, while others get their ESP capabilities from custom internal code.

User Recommendations

- Use ESP platforms when conventional data-at-rest architectures cannot process high-volume streams fast enough to meet business requirements.
- Acquire ESP functionality through a SaaS offering, an IoT platform or an off-the-shelf application that has embedded ESP logic if a product that targets specific business requirements is available.
- Use vendor-supported closed-source platforms or open-core ESP products that mix open-source with closed-source extensions for applications that need enterprise-level support. Use free, community-supported, open-source ESP products if developers are familiar with open-source software, and license fees are more important than staff costs.
- Use ESP platforms or stream data integration tools to ingest, filter, enrich, transform and store event streams in a file or database for later use.
- Choose a unified real-time platform with embedded ESP capabilities over a plain ESP platform if the application uses both data at rest and data in motion.

Sample Vendors

Confluent; EsperTech; Google; Hazelcast; IBM; Microsoft; Oracle; SAS; Software AG; TIBCO Software

Gartner Recommended Reading

Market Guide for Event Stream Processing

5 Essential Practices for Real-Time Analytics

Create an Optimal IoT Architecture Using 5 Common Design Patterns

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

Entering the Plateau

Full Life Cycle API Management

Analysis By: Shameen Pillai

Benefit Rating: High

Market Penetration: More than 50% of target audience

Maturity: Early mainstream

Definition:

Full life cycle API management involves the planning, design, implementation, testing, publication, operation, consumption, versioning and retirement of APIs. API management tools help creating API ecosystems, publishing and operating APIs, and collecting analytics for monitoring and business value reporting. These capabilities are typically packaged as a combination of a developer portal, API gateway, API design, development and testing tools as well as policy management and analytics.

Why This Is Important

APIs are widely used as the primary choice to connect systems, applications and devices; integrate business partners, and build modular and composable software architectures. The use of APIs as digital products — monetized directly or indirectly — is also on the rise. Digital transformation initiatives have accelerated the need for creation, management, operations and security of APIs. They have also made full life cycle API management an essential foundational capability for every organization.

Business Impact

Full life cycle API management provides the framework and tools necessary to manage and govern APIs that are foundational elements of multiexperience applications, composable architectures and key enablers of digital transformations. It enables the creation of API products, which may be directly or indirectly monetized, while its security features serve to protect organizations from the business risk related to API breaches.

Drivers

- Organizations are facing an explosion of APIs, stemming from the need to connect systems, applications, devices and other businesses. The use of APIs in internal, external, B2B, private and public sharing of data is driving up the need to manage and govern APIs using full life cycle API management.

- APIs that package data, services and insights are increasingly being treated as products that are monetized and enable platform business models. Full life cycle API management provides the tooling to treat APIs as products.

- Digital transformation drives an increased use of APIs, which in turn increases the demand for full life cycle API management.

- Organizations looking for growth acceleration and business resilience are adopting API-based architectures to eliminate or modernize their legacy and monolithic applications.

- Developer mind share for APIs is growing. Newer approaches to event-based APIs, design innovations and modeling approaches — such as GraphQL — are driving interest in full life cycle API management, leading to more experimentation and growth.

- Cloud adoption and cloud-native architectural approaches to computing (including serverless computing) are increasing the use of APIs in software engineering architectures, especially in the context of microservices, service mesh and serverless.

- Greater awareness of and increasing significance to API security are driving organizations to take charge of discovering and managing APIs. This often starts with operational management of existing APIs.

- Regulated, industry-specific initiatives in the financial, insurance and healthcare industries continue to provide a steady demand. However, use of APIs is spreading across most industries (especially retail, technology and telecom, manufacturing), increasing the demand.

- Commoditization and widespread availability of API gateways as part of cloud services, security solutions and other bundled software applications are increasing the need for distributed API management that involves multiple gateways, including those from multiple vendors.

- Rising influence of generative AI and large language models in software engineering is likely to increase and reshape the need for APIs.

Obstacles

- A lack of commitment to adequate organizational governance processes hinders the adoption of full life cycle API management. This can be due to a lack of skills or know-how, or due to putting too much focus on bureaucratic approaches rather than federated and automated governance approaches.

- A lack of strategic focus on business value (quantifiable business growth or operational efficiency) and too much focus on technical use cases can disengage business users and sponsors. This is particularly apparent in cases where API programs fail to deliver the promised return on investment.
- Traditional, single-gateway approaches to API management no longer fit well with modern, distributed API management approaches.
- A partial or full set of embedded API management capabilities provided by vendors in other markets — such as application development, integration platforms, security solutions and B2B offerings — can partially meet the use cases for API management and shrink the market opportunities.

User Recommendations

- Use full life cycle API management to power your API strategy and address both technical and business requirements for APIs. Select offerings that can address both well beyond the first year.
- Treat APIs as products managed by API product managers in a federated API platform team. Adopt business metrics for API products.
- Choose a functionally broad API management solution that supports modern API trends, including microservices, multigateway and multicloud architectures. Ensure that the chosen solution covers the entire API life cycle.
- Use full life cycle API management to enable governance of all APIs, including third-party (private or public) APIs that you consume.
- Ask full life cycle API management vendors about their support for automation, especially for API validation, testing and DevOps integration. Also ask about support for low-footprint and third-party API gateways.
- Full life cycle API management solutions are suitable for implementing a single-vendor strategy as opposed to a best-fit tooling approach for different API life stages. Ensure the choice aligns with your organization's goals.

Sample Vendors

Axway; Google; IBM; Kong; Microsoft; Salesforce (MuleSoft); Software AG

Gartner Recommended Reading

Magic Quadrant for Full Life Cycle API Management

The Evolving Role of the API Product Manager in Digital Product Management

How to Use KPIs to Measure the Business Value of APIs

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

Reference Model for API Management Solutions

IoT Integration

Analysis By: Andrew Comes

Benefit Rating: High

Market Penetration: More than 50% of target audience

Maturity: Mature mainstream

Definition:

Internet of Things (IoT) integration comprises the strategies and technologies to assemble end-to-end, IoT-enabled business solutions. Specific IoT integration challenges include constraints on devices, limited network connections and dealing with edge infrastructure. Traditional IoT integration challenges include integrating IoT applications with enterprise applications, events and data, business processes, SaaS applications, B2B partners and mobile apps.

Why This Is Important

All IoT projects require substantial integration work to enable IoT devices, data and applications, and various existing business applications to deliver business value. The unique requirements around integration demand major investments in strategies to support IoT integration projects.

Business Impact

IoT integration is essential to the success of IoT projects. Software engineering leaders and application leaders whose teams are supporting IoT projects must address IoT integration. To successfully deliver IoT products, they must train or hire software engineers with unique IoT integration skills.

Drivers

- Proliferation of IoT projects — for which IoT integration is always required.
- A growing desire to ingest and analyze IoT data to support data-driven business decisions.
- Extraordinary IoT project technology heterogeneity — for example, multiple types and OEMs offering IoT devices, brand-new and decades-old products and equipment, diverse IoT-device data heterogeneity and diverse application systems to be integrated.
- To fully realize the benefits of IoT, companies will eventually need to integrate new IoT technologies with legacy business applications and software — that is, pre-IoT — using new, enhanced workflows.
- Complex, distributed IoT projects often involve a mix of IoT devices, IoT platforms, business applications, mobile apps, cloud services and, often, external business partners. Such complex projects are needed to enable new IoT-driven outcomes — for example, self-diagnosing and self-repairing assets and equipment, lights-out-manufacturing or product as a service.
- A growing need to align time-series data generated by various IoT-connected assets and equipment with traditional EAM master data — for example, bill of materials (BOM) — for the same assets and equipment.
- Performance and scalability — that is, potentially large numbers of IoT devices, products and equipment with high API throughputs and large volumes of time-series data must be integrated.

Obstacles

- Software engineering leaders tend to focus on building software engineering teams for IoT projects with skills in IoT data, applications and analytics — rather than skills in IoT integration.
- IoT integration requires a specific set of skills that combines integration and IoT skills — this combination is not often common. Vendors investing in IoT products — for example, IoT platforms — tend to focus more on IoT data, applications and analytics rather than on integration, which creates integration functionality gaps.
- A function gap exists in general-purpose integration tools for IoT-specific integration needs of IoT projects. While many integration tools support modern IoT device protocols, most cannot connect to older “brownfield” operational technology (OT) equipment.
- IoT integration products focused on OT integration may be needed and must be licensed separately.
- Perceived high cost of IoT-specific integration tools or services.

User Recommendations

- Avoid simplistic approaches to IoT integration, such as believing that APIs are the same as integration. That cannot alone address all your needs and does not address functionality, such as IoT data translation and OT integration.
- Identify what IoT integration functionality is needed for IoT projects.
- Hire or train software engineers with IoT integration skills.
- Adopt event-driven approaches to integrate IoT.
- Confirm the availability of required IoT integration capabilities for any IoT product or service.
- Make your IoT integration strategy part of your overall hybrid integration platform capability if you have specific IoT requirements.

Sample Vendors

Alleantia; Amazon; PTC; Reekoh; Sky Republic; Software AG; Youredi

Gartner Recommended Reading

What Should I Do To Ensure Digital Twin Success?

Use APIs to Modernize EDI for B2B Ecosystem Integration

How to Deliver a Truly Hybrid Integration Platform in Steps

Select the Right Event Broker Technologies for Your Event-Driven Architecture Use Cases

MASA

Analysis By: Anne Thomas

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Mature mainstream

Definition:

Mesh app and service architecture (MASA) is a composition-based application architecture pattern that enables application delivery teams to respond rapidly to changing business demands and support multiple experiences. A MASA application is implemented as a mesh of distributed, loosely coupled, autonomous and shareable components, including one or more fit-for-purpose apps and composable multigrained back-end services. Apps and services communicate via mediated APIs.

Why This Is Important

MASA is a foundational pattern for modern business application architecture. It is the technical architecture that enables agility, composability, multiexperience applications and rapid delivery of new features. MASA facilitates incremental modernization of legacy applications while providing mechanisms that ensure security and robust operations.

Business Impact

MASA is both an architecture pattern for individual applications and a strategy for modernizing application portfolios. It enables organizations to respond rapidly to opportunities, disruptions and changing business priorities by extending existing applications or composing new ones.

Drivers

Engineering teams initially adopted MASA to improve agility by decoupling front-end and back-end capabilities. This driver was reinforced as teams were asked to add support for mobile experiences. MASA enables many other critical application capabilities, such as:

- Multiple experiences for different types of devices and modalities, such as voice, touch, wearables and immersive technologies.
- Distinct, optimized experiences for the different personas that use an application.
- Rapid response to disruptive events and changing business priorities via composition of existing services and creation of new experiences.
- Greater flexibility through loose coupling of components.
- Improved application performance, scalability, security and resilience through intelligent mediation.

Obstacles

- The biggest obstacle to MASA is the extensive technical debt embedded in existing application portfolios.
- MASA requires applications to be decomposed into distributed components, thereby increasing application complexity.
- MASA requires application functionality in legacy applications to be encapsulated and exposed via APIs. To gain the most value from MASA, those legacy applications should be modernized and refactored to convert the embedded business logic into composable services.
- The architecture enables iterative modernization, but it will take years (perhaps decades) to modernize the entire application portfolio.
- MASA requires an investment in API mediation and multiexperience technologies.
- MASA requires culture change as architects and development teams need to embrace distributed models and adopt new ways of thinking about the architecture of systems.

User Recommendations

- Analyze your business's digital transformation roadmap and identify and prioritize applications to modernize.
- Take a pragmatic approach to creating services: Encapsulate, extend or refactor existing applications or build new services.
- Determine appropriate service granularity based on your objectives. Don't presume that all services must be microservices.
- Ensure that development teams have competence in user-experience design, service-oriented architecture, API design and domain-driven design.
- Update existing technical architectures, governance mechanisms and success metrics to align them with MASA.
- Mediate API traffic to apply governance, performance and security policies and to enable API transformation and orchestration.

Gartner Recommended Reading

Adopt a Mesh App and Service Architecture to Power Your Digital Business

How to Create Shared API Services to Enable Composability

MASA: Create Agile Application Architecture With Apps, APIs and Services

Accelerate Digital Transformation With an API-Centric Architecture for Enterprise Applications

Mediated APIs: An Essential Application Architecture for Digital Business

Leading Teams to Success With Microservices Architecture

Data Integration Tools

Analysis By: Ehtisham Zaidi, Robert Thanaraj

Benefit Rating: High

Market Penetration: More than 50% of target audience

Maturity: Mature mainstream

Definition:

Data integration tools allow independently designed data structures to work together by enabling the design and implementation of data access, transformation, enrichment and delivery capabilities. Data integration tools have matured from supporting batch integration to supporting a combination of delivery styles (such as data virtualization, data replication, messaging and streaming). Data integration tools must support hybrid and multicloud integration scenarios for diverse data persona.

Why This Is Important

Organizations need data integration tools to access, integrate and deliver data at all latencies across a range of data sources and targets in support of various data and analytics use cases. These include analytics, data science, data fabric enablement, MDM, application integration and self-service data preparation. Growing requirements to migrate and integrate data with cloud data stores also require data integration tools as a foundational infrastructure investment.

Business Impact

Organizations adopting mature data integration tools increasingly exploit comprehensive data access and delivery capabilities. They get immediate benefits in the form of:

- Flexibility (to access new data sources)
- Reduced time to integrated data delivery
- Cost savings (by reduced integration technical debt)
- Quality enhancements (for analytics/data science products)

Integration tools that support data fabric designs will increase the productivity of data engineering and data science teams.

Drivers

- Ability to execute data integration in a hyperconnected infrastructure (irrespective of structure and origins) and to automate transformations through embedded ML capabilities are the most important drivers for organizations investing in modern data integration tools.

- Traditional data integration architectures and tools, which focus solely on replicating data, are slow in delivering semantically enriched and integrated datasets that are ready for analytics. This pushes data integration tools to provision a mix of variable latency, granularity, physical and virtualized data delivery. This is another major reason to invest in these tools.

- Self-service data access and data preparation may be conducted by citizen integrators, non-IT roles or by skilled data engineers. These varied personas spur additional requirements for data integration tools.

- While traditional data integration tools have become mature in technical metadata ingestion and analysis to support data integration activities, there is still room for maturity to be able to harness and leverage "active" metadata. Organizations must investigate data integration tools that can not only work with all forms of metadata, but also share it bidirectionally with other data management tools to support data fabric architectures that enable automation.

- Data integration tools will be foundational in supporting this strategy to enable a multicloud/hybrid data management architecture.

- Most organizations are in the midst of their cloud data migration strategy.

- Organizations are struggling to make their limited data engineering teams more productive. Data integration tools that are able to access and utilize metadata across data management infrastructure to provide recommendations on data integration design and delivery will drive integration technology adoption.

Obstacles

- Tightly integrated data integration tool suites in which all components share metadata, and support a common design and administration environment, remain an area for improvement in the data integration tools market.
- The popularity of data preparation (and other self-service integration tools), with the sole focus on analytics use cases, will create some confusion in the market, slowing the advance of more complete integration tools that support both data and analytics use cases.
- The demand for a data integration platform that combines multiple data delivery styles (e.g., batch with data virtualization), multiple deployment options (hybrid and multicloud) and multiple personas, exceeds the capabilities of most offerings.
- Many new data integration tools are competent at data ingestion, but don’t support complex data transformations, data pipeline orchestration, data lineage, etc. This requires organizations to invest in several other tools to enable end-to-end integration scenarios.

User Recommendations

- Assess your data integration program to identify gaps in critical skill sets, tools, techniques and architecture needed to position data integration as a strategic discipline at the core of your data management strategy.
- Review current data integration tools to determine whether you are leveraging the capabilities they offer. These may include the ability to deploy core elements (including connectivity, transformation and movement) in a range of different data delivery styles driven by common metadata, modeling, design and administration environments.
- Identify and implement a portfolio-based approach to your integration strategy that extends beyond consolidating data via ETL/ELT to include stream data integration, event recognition and data virtualization.
- Make automation of data integration, ingestion and orchestration activities your primary goal for the year, and focus on those data integration tools that can support data fabric designs.

Sample Vendors

Denodo; IBM; Informatica; Matillion; Oracle; Precisely; Qlik; Talend; TIBCO Software

Gartner Recommended Reading

Magic Quadrant for Data Integration Tools

Critical Capabilities for Data Integration Tools

iPaaS

Analysis By: Keith Guttridge

Benefit Rating: High

Market Penetration: More than 50% of target audience

Maturity: Mature mainstream

Definition:

Gartner defines integration platform as a service (iPaaS) as a vendor-managed cloud service that enables end users to implement integrations between a variety of applications, services and data sources. iPaaS platforms support at least one of the three main uses of integration technology — data consistency, multistep processes and composite services. iPaaS provides adapters to simplify configuration connectivity. It provides a low-code workflow environment and supports hybrid deployment models.

Why This Is Important

The shift to the cloud along with digital and automation initiatives is boosting the iPaaS market, making it the biggest segment of the integration platform technology market (valued at $6.5B in 2022). Its functional breadth makes it the natural alternative to classic integration software such as enterprise service bus (ESB), and extract, transform and load (ETL) software for large organizations.

Business Impact

By addressing integration needs in a rapid and cost-effective way, iPaaS enables organizations to improve efficiency, increase business agility and introduce innovation faster. iPaaS adoption helps organizations achieve these goals cost-effectively, efficiently and with less specialized personas than are needed for classic integration software. Also, iPaaS makes these benefits accessible to midsize and small organizations that cannot afford the costs of a classic platform.

Drivers

- iPaaS is frequently adopted by organizations that seek to modernize their integration capabilities. This way, they can support digital transformation and cloud adoption (especially SaaS), and enable new roles and personas to be part of the integration delivery.

- There has been relative success for iPaaS targeting particular industries, SaaS ecosystems, business processes or geographies. This is due to its appeal to time-, skill- and resource-constrained organizations.

- The main goal of iPaaS providers now is to maximize opportunities to upsell and cross-sell to their vast installed base. Therefore, they are evolving their offerings into enterprise-class suites that address a wide range of hybrid, multicloud scenarios. Hence, large and global organizations now position iPaaS as a strategic option to complement — but increasingly also to replace — classic integration platform software. This drives deeper commitment among enterprises.

- A growing number of SaaS providers "embed" in their applications their own iPaaS, or one from a third party, which they typically extend with a rich portfolio of packaged integration processes (PIPs). This makes embedded iPaaS offerings attractive to organizations that need to quickly integrate a SaaS application.

- Providers will keep investing to improve developers' productivity, reduce time to value and shorten the learning curve. The goal is to further expand their potential audience to include business users. Hence, providers' R&D efforts might focus on using AI, machine learning and natural language processing. These capabilities can assist development and operation, enrich PIP portfolios, and enable continuous integration or delivery (CI/CD) and DevOps to entice professional developers.

Obstacles

- The market's extreme fragmentation (more than 175 providers) and diversity of capabilities makes it hard for user organizations to select the best-fit iPaaS for their needs. This could generate a proliferation of diverse, stand-alone and embedded iPaaS offerings. It could also risk fragmenting service providers' investments in skills building.
- The top five iPaaS providers command about 56% of the market, and only 10 providers have more than a 2% share. The vast majority of providers are smaller vendors with limited brand awareness. This may discourage risk-averse organizations from making strategic investments in smaller vendors or less familiar brands.
- Key obstacles are the API rhetoric of seamless "plug-and-play" integration, and confusion among less technically savvy users about the differences between iPaaS and API management platforms. There is also a growing trend for code-based integration encouraged by the cost of integration platforms and serverless PaaS functions.

User Recommendations

Adopt iPaaS when looking for:

- An integration platform for midsize organizations moving to the cloud and for "greenfield" integration initiatives.
- A strategic complement to traditional integration platforms — increasingly in the context of integration strategies based around hybrid integration capability framework— in order to empower a collaborative, democratized approach to integration.
- An enabler of self-service integration for business technologists, such as SaaS administrators or citizen integrators.
- A platform to support well-defined, tactical integration projects with low budgets, severe time constraints, and informally defined and incrementally formulated requirements.
- A potential replacement for classic integration platforms that are obsolete or cannot support changing requirements.

Sample Vendors

Boomi; Jitterbit; Microsoft; Oracle; Salesforce (MuleSoft); SAP; SnapLogic; TIBCO Software; Tray.io; Workato

Gartner Recommended Reading

Magic Quadrant for Integration Platform as a Service

Critical Capabilities for Integration Platform as a Service

Choose the Best Integration Tool for Your Needs Based on the Three Basic Patterns of Integration

How to Select the Right Mix of Integration Technologies

LCAP

Analysis By: Paul Vincent, Yefim Natis, Oleksandr Matvitskyy

Benefit Rating: High

Market Penetration: More than 50% of target audience

Maturity: Mature mainstream

Definition:

A low-code application platform (LCAP) supports visual programming abstractions, such as model-driven and metadata-based application development. Developers use LCAP to create user interfaces, design data schemas, and implement business logic with simplified tooling and their catalog of packaged capabilities.

Why This Is Important

LCAPs are one of the most popular types of low-code tools. They are used as a general purpose platform for web and mobile application development with high productivity while requiring fewer specialized developer skills. They can facilitate access to new, rich or advanced technology sets for developer personas ranging from enterprise software engineering experts to citizen developers. Over 300 vendors support a wide variety of business use cases and industry specializations.

Business Impact

Accelerating application delivery while reducing developer effort and increasing self-service capabilities are high priorities for business IT. Businesses adopt LCAP to deliver more automation and reduce their application backlogs as well as enable democratized application development. Mostly cloud-based, LCAP vendors are investing in the development of new capabilities to increase their use-case coverage and justify their platform subscription costs.

Drivers

- The demand to automate and digitize business through applications continues to outstrip conventional software development capacity. This is despite the rise of SaaS usage for standard business services — indeed, the latter has resulted in higher demands for application development for differentiating SaaS extensions. This means a large part of the LCAP market is for SaaS vendors' LCAP.
- LCAPs have evolved from rapid application development, business process technologies, mobile application development and SaaS platforms in order to increase productivity for common application types. They have evolved common capabilities around user interface, database, business logic definition, process orchestration and integration of ubiquitous REST API services.
- LCAPs continue to evolve to meet customer needs. Their low-code paradigms have been extended to include ever more functionality such as integration, artificial intelligence/machine learning (AI/ML) services and multiexperience support. LCAPs increasingly overlap with the business process automation market for workflow use cases, and the multiexperience development platform (MXDP) market for user-interface-driven use cases.
- Vendors are focusing more on cloud-native scalability to support larger B2C deployments, and deeper governance and collaboration tooling to support citizen/business and IT development fusion team structures. Support for composing applications from multiple API and service types enables LCAPs to cover an increasingly large set of enterprise application requirements, with some enterprises starting to choose them as a strategic application platform.
- Professional developers have started to realize the opportunities of low-code development, creating new markets for LCAP beyond business developers and citizen developers. This is driving the evolution of new low-code development accelerators for pro-code developers, further widening the LCAP market.

Obstacles

- Current LCAP market share is heavily biased toward some very large SaaS providers and a few successful independent vendors. However, enterprises often have multiple LCAP offerings to support different developers and use cases, often at increased total cost of ownership.
- LCAPs that have been implemented by the main SaaS platform vendors could diminish the opportunities for many small LCAP vendors. Vendors have been known to deprecate features or change low-code offerings, requiring additional maintenance by developers.
- LCAP trades productivity for vendor lock-in (of both applications and developer skills). Lock-in reduces customer flexibility and can increase costs.
- The wider developer audiences encouraged by LCAP increase the requirements for governance and guardrails, which are at best still only partially delivered by vendors.
- Licensing models vary across vendors and often change, and may not scale for new use cases. This can lead to vendor disillusionment.

User Recommendations

- Formulate LCAP usage strategies to address vendor selection against different use cases, developer persona, licensing costs, and technology requirements. Most organizations will support several LCAPs in their portfolios across SaaS, multiexperience and business process automation platforms.
- Review applications for the long-term effects of vendor lock-in, the lack of portability or standards, and ease of technical debt accumulation. Vendor relationships (and contracts) need to be considered strategic and long-term.
- Analyze LCAP subscriptions against their productivity and time-to-market benefits: minimize end-user pricing if desiring B2C rollouts.
- Ensure developers are governed according to their needs. Different developers with different skill sets will vary in their successful adoption of different LCAPs.
- The large number of LCAP vendors implies possible future market instability, although to date, there have been only a few LCAP retirements.

Sample Vendors

Alibaba Cloud; Huawei Technologies; Kintone; Mendix; Microsoft; Newgen Software; OutSystems; Retool; ServiceNow; Unqork

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

How to Navigate the Application Platforms Market Including Cloud-Native, Low-Code and SaaS

Drive High-Value LCAP Outcomes Through Strategic Partnerships With Service Providers

Appendixes

See the previous Hype Cycle: Hype Cycle for Application Architecture and Integration, 2022

Hype Cycle Phases, Benefit Ratings and Maturity Levels

Table 2: Hype Cycle Phases

(Enlarged table in Appendix)

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

Source: Gartner (August 2023)

Table 3: Benefit Ratings

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

Source: Gartner (August 2023)

Table 4: Maturity Levels

(Enlarged table in Appendix)

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

Source: Gartner (August 2023)

Document Revision History

Hype Cycle for Application Architecture and Integration, 2022 - 21 July 2022

Hype Cycle for Application Architecture and Integration, 2021 - 15 July 2021

Hype Cycle for Application and Integration Infrastructure, 2020 - 30 July 2020

Hype Cycle for Application and Integration Infrastructure, 2019 - 1 August 2019

Hype Cycle for Application and Integration Infrastructure, 2018 - 26 July 2018

Hype Cycle for Application Infrastructure and Integration, 2017 - 7 August 2017

Hype Cycle for Application Infrastructure, 2016 - 7 July 2016

Hype Cycle for Application Infrastructure, 2015 - 30 July 2015

Recommended by the Authors

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

Understanding Gartner's Hype Cycles

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

Hype Cycle for Application Architecture and Integration, 2022

Hype Cycle for Software Engineering, 2023

Table 1: Priority Matrix for Application Architecture and Integration, 2023

Benefit	*Years to Mainstream Adoption*			
↓	***Less Than 2 Years ↓***	***2 - 5 Years ↓***	***5 - 10 Years ↓***	***More Than 10 Years ↓***
Transformational	Event Stream Processing		Data Fabric Self-Integrating Applications	
High	Data Integration Tools Full Life Cycle API Management IoT Integration iPaaS LCAP MASA Packaged Integrating Processes	Cloud Event Brokers Data Hub iPaaS Digital Integration Hub Digital Integrator Technologies Event-Driven Architecture Hybrid Integration Capability Framework Integration Strategy Empowerment Team Microservices	Superapps	
Moderate	Backend for Frontend Cloud-Native Architecture	GraphQL APIs Micro Frontend	Miniapps	
Low		Service Mesh		

Source: Gartner (August 2023)

Table 2: Hype Cycle Phases

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

Phase ↓	*Definition* ↓

Source: Gartner (August 2023)

Table 3: Benefit Ratings

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

Source: Gartner (August 2023)

Table 4: Maturity Levels

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

Source: Gartner (August 2023)