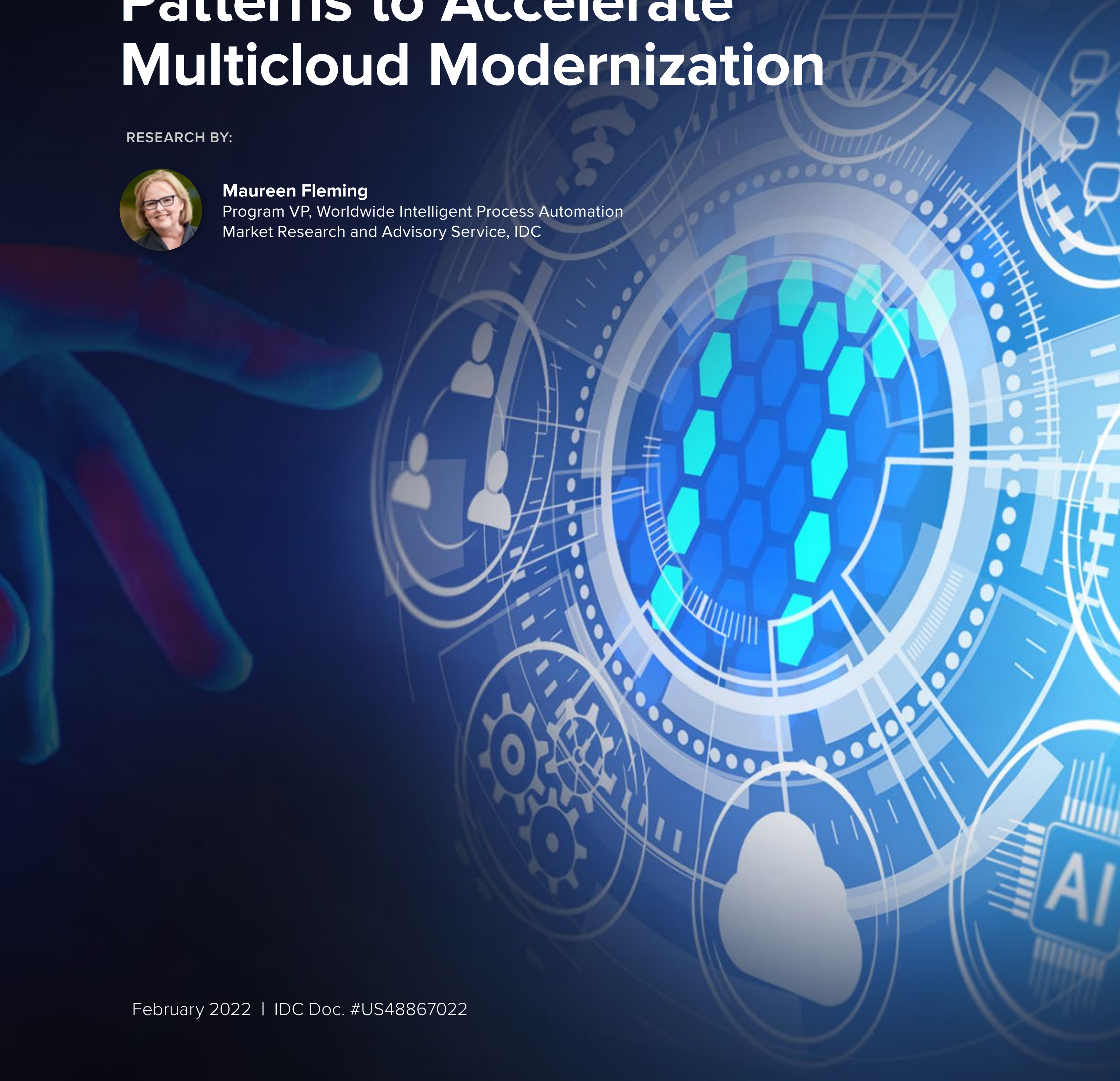
Using Four Integration Patterns to Accelerate Multicloud Modernization

RESEARCH BY:

Maureen Fleming
Program VP, Worldwide Intelligent Process Automation
Market Research and Advisory Service, IDC



February 2022 | IDC Doc. #US48867022

Navigating this White Paper

Click on titles or page numbers to navigate to each section.

Introduction

Investments in modernization and transformation resulted in organizations that are different today than they were three years ago. Ongoing investments mean that differences will be sharper over the next five years. The consequences of these business changes and advances in technology mean that there are equally impactful changes in how applications, processes, and systems are built, extended, connected, observed, and managed:

- A multicloud strategy is required because building, automating and connecting applications and data mean the endpoints and data sources are often distributed across clouds, datacenters, regions, and value chains.
- Automation is a major priority that requires a broader portfolio of technologies to support the full set of requirements.
- Faster cycle times are creating architectural changes to system design along with new lighter-weight transformational overlays interoperating with application back ends.
- The volumes of data that need to be collected continue to be an enormous challenge even as requirements diverge between the teams that need data in near real time, data scientists who need training data, and analysts who need data for reporting purposes and for analytics.

The complete technology stack differs across these topics, but the underlying enablement is the interlocking of both data and application integration, application programming interface (API) life-cycle management, automation, and events and insights to support the needed interoperability, speed, and resilience that form the foundational capabilities necessary to support change.

Organizations that recognize the value of building a modern technology-based foundation for the future will find it far easier to succeed in responding to ongoing change and new areas of opportunity. Three core layers to this foundation are:

- Modern foundational integration and automation patterns to speed up the enablement of change.
- Multimodal automation platforms to broaden the use cases for automation and operational improvements.
- Platform development environments that support new developer roles and styles of development.

Automation is a major priority that requires a broader portfolio of technologies to support the full set of requirements.

Modern Foundational Integration and Automation Patterns

Today, coping with operating at faster cycle times in a highly complex, distributed fashion that is also resilient means there is an intense dependency on being very good at many facets of integration. That means API management, secure API gateways, data movement, and modern ways of broadcasting data as events. But that isn't enough.

When integration involves connecting across distributed systems – including diverse clouds and cloud regions -- many organizations package common methods of interactions and connectivity into integration design patterns. An integration architecture team will typically maintain these patterns, making changes as dictated by new business and technology requirements. When the new design patterns are mapped to existing integration platforms, gaps in functionality are identified and may require the integration teams to implement new types of integration software.

The two most common integration design patterns in use today, arguably, are request-response and extract, transform, and load (ETL). With request-response, application logic sends a request to the API of a web service or to an adapter that connects to a database and waits for the response. The endpoint executes the request and returns the results back to the requester. With ETL, jobs are created to schedule the extraction of new data from a source database; the data is transformed to make it compatible with the target data store and loads the newly transformed data into the target.

While these integration design patterns continue to be heavily used, integration patterns that are not as commonly adopted are becoming more important and new patterns are emerging. New needs and technologies also advance new design patterns. This is particularly true when the new technologies improve ease of enablement and ongoing management of the pattern. In addition, new architectures aligned with business requirements call for new design patterns. In essence, rethinking integration design patterns are much more important for several intersecting reasons including:

- The need to deliver capabilities quickly, even when those capabilities can be complex.
- Highly distributed, multicloud nature of systems design.
- Delivery of capabilities via composition requiring improvements in scalability, application elasticity, and resilience.
- Opportunity to harness newer, more transient, and higher-volume types of data.
- New event-driven styles of application design that need low-latency access to data.
- Requirements for automated proactive communications with customers and ecosystems.

To illustrate how integration patterns are changing, IDC has identified several modern foundational integration-intensive design patterns along with facts and use cases about why these are relevant. We call the patterns modern because they represent key requirements needed to support today's business strategies. And they are foundational because they are already or will be adopted ubiquitously across enterprises.

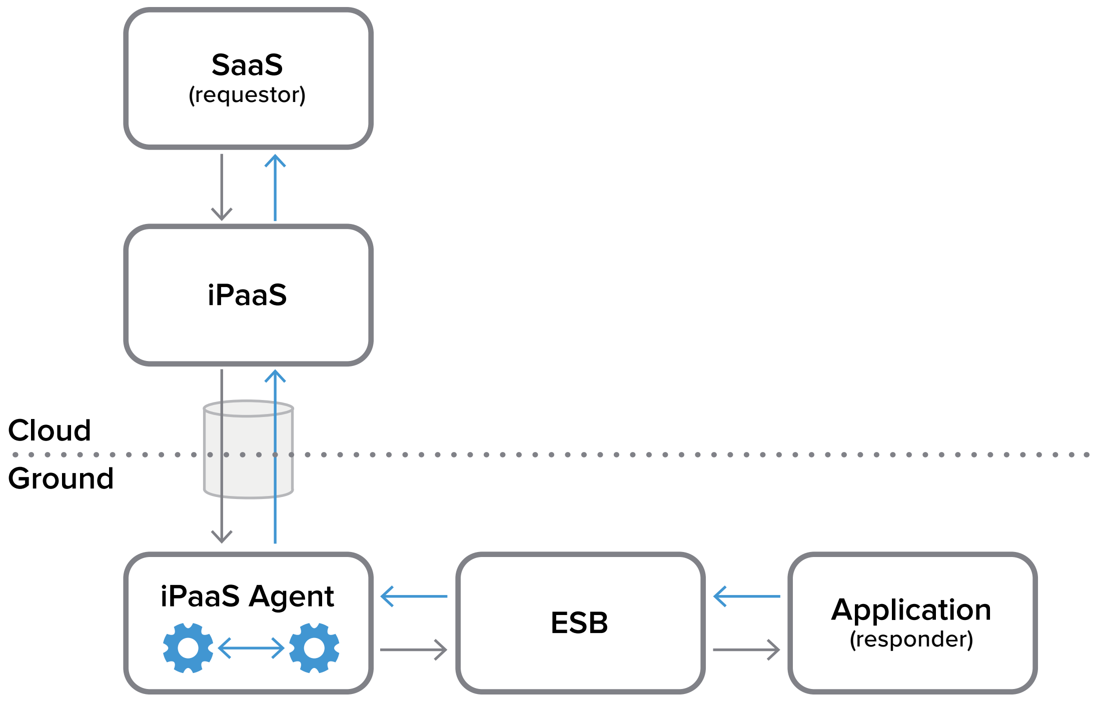
Agent-Based Connectivity Between Cloud and Ground

This design pattern uses local agent software to connect an integration platform as a service (iPaaS) or a cloud integration service with a legacy enterprise service bus (ESB) to exchange information between one or more
on-premises applications connected to the ESB, as shown in **Figure 1**. The pattern requires the on-premises-installed agent to communicate with the iPaaS via VPN and to also connect to the ESB that manages the web services of the required legacy applications.

The cloud SaaS request is executed in the iPaaS in a representational state transfer (REST) format and is forwarded to the agent, which translates the request and communicates with the ESB. The ESB calls an existing web service and returns the results to the agent. The agent then translates the Simple Object Access Protocol (SOAP) response to the required REST format.

FIGURE 1

Agent-Based Connectivity Between Cloud and Legacy Application Ground



Source: IDC, 2022

Background

SOAP was introduced in the early 2000s as a way to exchange structured information across web services using standard internet protocols. The early 2000s also saw the introduction of REST as another internet standard for interacting with web services to exchange structured information.

Enterprise integration vendors opted to support SOAP and introduced the ESB as a way to exchange information across applications. In the case of REST, vendors like eBay and Amazon adopted REST-based web services to open up their services to partners. Partners could use these web services to offer value-added services to their customers. By the early 2010s, REST became more dominant than SOAP, which effectively became a legacy way to handle integration.
Today, ESBs continue to be in production to support active web services connecting to on-premises client/server applications.

The importance of applications running in enterprise datacenters connected to an ESB is considerable, and access to the web services is needed to support interoperability between legacy and modern integration platforms, especially those running in public clouds. In 2020, on-premises software spending accounted for 69% of total software investments, with cloud representing 31%. Over the next five years, the percentages will flip, with cloud accounting for 58% and on premises for 42% of spending. But even at 42%, on-premises spending still represents a large investment for systems that are in production in enterprise datacenters. A substantial percentage of those systems will be characterized as legacy software. They also may be characterized as mission-critical systems.

Meanwhile, 60% of spending on application integration in 2020 was used for on-premises integration, while 40% of spending was based on cloud adoption of integration software for intracloud information exchange as well as cloud-to-cloud and ground-to-cloud use cases.
Both SOAP and REST continue to be used on premises, with SOAP especially associated with applications built on a client/server architecture. REST is currently considered the de facto standard for cross-application information exchange for modern applications.

60% of spending on application integration in 2020 was used for on-premises integration, while **40% of spending** was based on cloud adoption of integration software for intracloud information exchange as well as multicloud and ground-to-cloud use cases.

Use Cases

- One of the earliest use cases of REST-to-SOAP connectivity came from the development of mobile applications, where REST APIs passed requests to ESB. As mobile back ends began to run in the cloud, the need for connectivity to legacy systems continued.
- An ERP that continues to run on premises may need to exchange data with new cloud-based applications, requiring cloud-to-ground mechanisms between cloud integration and ESBs.
- While modernizing its financial applications, an organization may maintain connectivity between core financial systems running in the datacenter and the modernized, cloud-based services until the modernization effort is complete.

Importance of Agent-Based Connectivity Design Pattern

This design pattern serves an enterprise's modernization and transformation journey for different reasons by:

- Providing a mechanism to safely move to a public cloud SaaS option because the pattern can keep the legacy application in production during the changeover.
- Supporting satellite applications that extend the core application as they continue to be used.
- Eliminating the need to invest in legacy changeover when there may be higher priorities for other transformation or modernization efforts.

Given the role of automation in cost reduction and process improvement, simplifying end-to-end connectivity between a legacy protocol and a modern protocol is an important design pattern, especially where the data can be exchanged in the background, without high wait times for customers or business users.

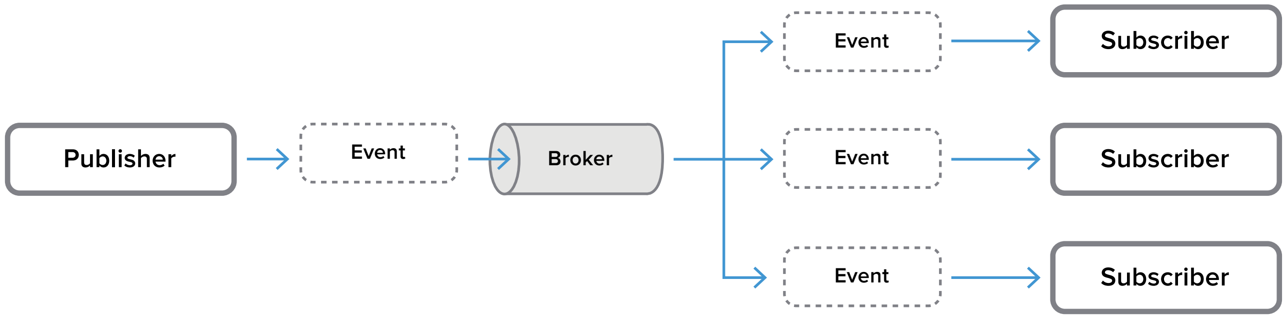
Publish and Subscribe

Publish and subscribe (PubSub) is the backbone design pattern for event-driven systems. Event-driven design is used to build systems that can detect, consume, and react to events. This type of architecture is core to the development of high-speed, highly scalable systems. Event-driven architecture is high speed because the design is associated with rapid delivery of events and scalable because the publisher role is able to broadcast to any number of subscribers, with every part of the interaction operating independently of each other.

PubSub is a type of asynchronous broadcast pattern supporting distributed computing where a publisher broadcasts events and any authorized consumer may subscribe to them. An event is a data state change or status change. Events can be stored temporarily in queues or caches or permanently in topics. **Figure 2** (next page) shows the basic design pattern of PubSub.

FIGURE 2

Publish and Subscribe Integration Design Pattern



Source: IDC, 2022

PubSub consists of the following elements:

- When an event producer creates a new event, it is published.
- Event brokers and event managers filter, process, and add metadata and an ID to each published event and, depending on the technology used to implement PubSub, add the event to a cache or queue or store the event permanently to its dedicated topic.
- Consumers subscribe to events using a designated subscription plan that identifies the channel, subject, or topic to which they want to subscribe and supports filtering policies. The event may be pushed to the subscriber via an AsyncAPI or other mechanism. The event may also be collected from the topic via a request-response pattern using a REST API.
- Consumers interact with event brokers to acknowledge the receipt of the event. Policies supporting delivery guarantees, such as guaranteed delivery and once and only once delivery, are also commonly implemented.

Background

The PubSub pattern originated as messaging technology involving the publishing of events into a message bus or message queue. Today, a variation of the PubSub pattern is extended to also support data replication, where multiple data systems and applications subscribe to or collect the event data via APIs. Apache Kafka and Apache Pulsar are two common open source technologies that support PubSub by capturing log event data associated with databases and delivering to a distributed log file or topic. The source event log data can be collected continuously — or streamed — to support high-speed data flows. The movement of log data from source to topic is called event streaming regardless of whether the log data is delivered continuously or collected in batches.

Traditionally, event-driven systems were specialized and used primarily for high-value, low-latency systems. The value had to be high value because supporting low latency was expensive. Because event-driven architectures typically involve distributed computing, the design complexity and skills required can also be significant. Early use cases were in financial services, where there was value in collecting trading information and aggregating stock price changes into stock quotation services and eventually converging the stock quotations with high-end analytics as well as becoming part of integrated trade execution systems.

From trading, event-driven architecture (EDA) evolved to become a major architecture for innovation that realized value by the high-speed aggregation of events across distributed systems. Innovations that turned into businesses or business differentiators include live turn-by-turn driving instruction, ride sharing, food delivery services, and status change alerts across delivery scenarios, especially for digital purchasing.

Today, heavily competitive businesses that by nature are volatile are shifting to event-driven digital services. Meanwhile, public cloud service providers seeking better scalability and resilience are reengineering their services to support event-driven design. And key SaaS offerings, at the least, are able to emit events that can be utilized by third-party systems.

With the emergence of OpenAPI's specification for AsyncAPI, the ability to subscribe to events using commercial integration and API management offerings promises to significantly increase ease of consumption. While collection and publication of events, especially with guarantee-related semantics, continue to be the most challenging part of the PubSub design pattern, executing this integration design well is increasingly critical in digital transformation projects.

Use Cases

Aside from the digital services that may be tied to a transformation initiative, enterprise use of PubSub is growing, especially where competitive advantage is based on speed and where it can solve customer experience problems. Examples include:

- **Inventory available to promise.** It used to be common to refresh inventory levels across an organization using ETL. Inventory shortages caused by slow update speeds mean that inventory available to promise is inaccurate, creating higher costs, customer dissatisfaction, and slow lead times. Converting to near-real-time situational awareness across all inventory systems may not solve inventory problems when an item is out of stock, but the costs of cancellations and customer service problems associated with false inventory readings are minimized.
- **Status notifications.** Once a digital order is completed, key milestone notifications are delivered to customers to notify them about each status change. Many of us have seen this over the past couple of years as we receive text messages acknowledging our digital purchases, and subsequent notifications that the product has shipped or food is on its way, and then when the product is delivered. This messaging is now ingrained in the consumer market to the point that notifications are now part of customer service.
- **Business to business (B2B) fulfillment.** In B2B, trading partners using electronic data interchange (EDI), AS2, and other approaches have relied on acknowledgements for decades, although documents and acknowledgements tend to be processed and distributed in batch. The promise of AsyncAPI as well as subscribing to topics managed by event brokers means that milestone notifications and delivery of trading documents can readily be delivered outside the domain of EDI.
- **Internet of Things (IoT).** Sensors emitted from IP-enabled equipment and machines use the MQTT protocol for internal communications across networks and are delivered out via gateway software or event streaming to subscribers continuously or in batch.

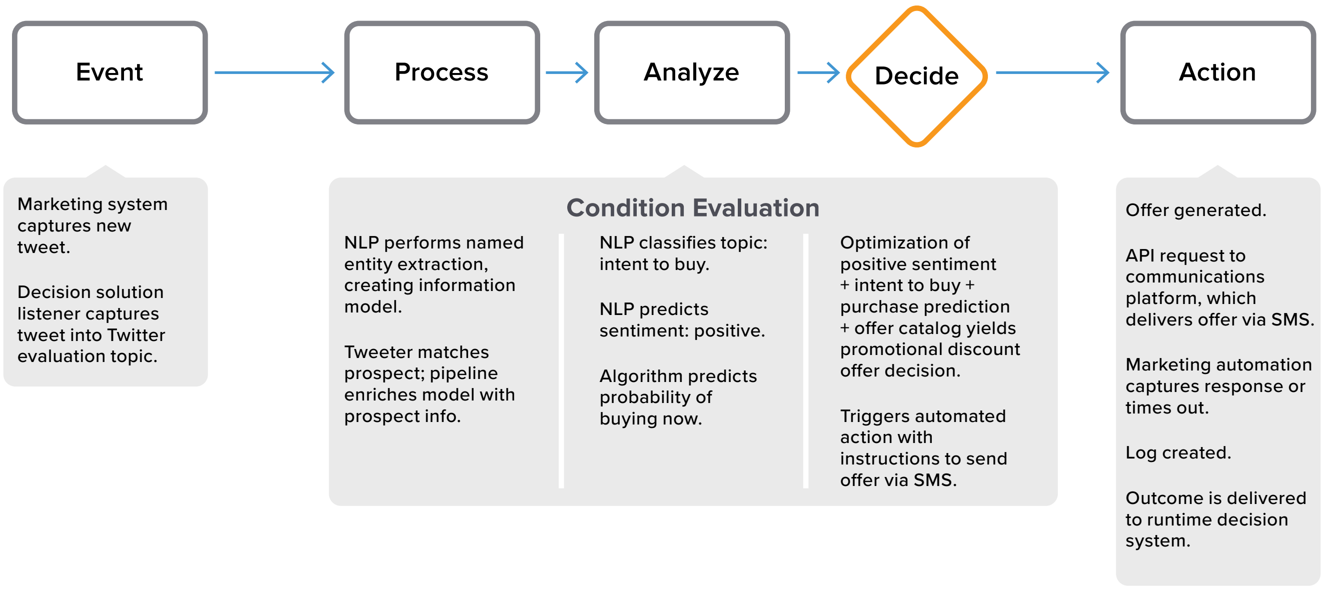
- **IT observability and predictive analytics.** As systems are increasingly instrumented, the log data is collected and streamed to an observability layer that maintains current awareness of how the systems are performing. The goal of observability is to identify pending failures at root cause to fix them as quickly as possible. Observability is moving up stack to data operations and business operations. These types of detect-and-respond systems drive efficiency and performance in each of their respective layers and are rapidly incorporating machine learning models aimed at using predictive analysis for problem detection. The goal is to narrow the window between detection and response to prevent or mitigate problems that damage customer experience or create unnecessary operating costs.

Data Pipelines and Event Stream Processing

While messaging middleware was the traditional mechanism to transform data to be compatible with subscribing systems, PubSub also is used to deliver data to a data pipeline for sequential processing and delivery to a target, whether the target is an application or a data lakehouse. There are variations of this delivery model. For example, a combination of PubSub and a data pipeline can support the combination of artificial intelligence (AI) with machine learning (ML)–based predictive analytics to make automated high-speed decisions as shown in **Figure 3**. When a data pipeline is used with AsyncAPI in combination with a collection of microservices to process the new data and apply it to AI and ML models, the ability to collect and process events at high speed narrows the time to insights when used in combination with the PubSub design pattern.

FIGURE 3

PubSub + Pipeline Design Pattern


Event
Process
Analyze
Decide
Action
Marketing system captures new tweet.
Decision solution listener captures tweet into Twitter evaluation topic.
Condition Evaluation
NLP performs named entity extraction, creating information model.
Tweeter matches prospect; pipeline enriches model with prospect info.
NLP classifies topic: intent to buy.
NLP predicts sentiment: positive.
Algorithm predicts probability of buying now.
Optimization of positive sentiment + intent to buy + purchase prediction + offer catalog yields promotional discount offer decision.
Triggers automated action with instructions to send offer via SMS.
Offer generated.
API request to communications platform, which delivers offer via SMS.
Marketing automation captures response or times out.
Log created.
Outcome is delivered to runtime decision system.

Source: IDC, 2022

Event Streaming with ELT Option

Extract, load, and transform (ELT) collects data from a source database or data store, loads the data into a destination target, and transforms the data upon arrival. This differs from the much more broadly adopted extract, transform, and load, where the transformation is handled in flight before the data is loaded. The ELT pattern is becoming more common as enterprises adopt event streaming, although ELT vendors also connect to Kafka using connectors.

The types of data collected by event streaming often broaden the volume of data that can be added to data stores. For example, enterprises increasingly deliver IoT data via Kafka; they can also collect screen event data from user interactions and can collect other sources of data that are typically classified as transient data compared with the transactional data stored in databases. This data is often used to build models of insight, including the development of prediction models and for lakehouse analytics initiatives.

Use Cases

ELT use cases include:

- **Unifying collection from data sources.** Business units, applications, and processes inside enterprises operate at different speeds, and the speed varies based on many factors, including competition, what frequency is needed to support requirements, or how frequency impacts customer service. The most challenging collection frequency is real time. Collecting once and replicating by frequency requirements simplifies the shift into continuous data collection and minimizes the effort to leverage the same data across multiple systems. Rather than individual teams going to source data to collect individually, deferring to Kafka or competitive approaches is more efficient and less expensive.
- **Business unit data sharing.** As business units make independent decisions about technology adoption, they are providing data access rights to agreed-upon data managed in Kafka or Kafka-equivalent topics. Using these access rights, adjacent business units and central IT then replicate the data into their own data stores.
- **Collection of transient data.** Transient data is data generated within an application session. The data is discarded or reset back to the default state at the completion of the session. Capturing transient data with time stamps is an increasingly valuable practice used in combination with AI, ML, and mathematical models. Capturing and collecting this data into a data store enables new types of high-value insights and algorithms.

Importance of PubSub Design Pattern

As PubSub technology evolves and as the secondary use of data explodes, collecting data once and leveraging it as needed is critical due to the explosion of data being collected as well as data refresh variances. As data refresh cycles shift from weekly to daily, and to multiple times per day to continuous, standardizing on technology that can rapidly adapt to each change is faster and simpler.

PubSub should be considered a de facto design pattern for both modernization and transformation efforts, particularly in processes that have direct impact on customers or where there is advantage in harnessing speed as a competitive advantage or to respond to competitive threats.

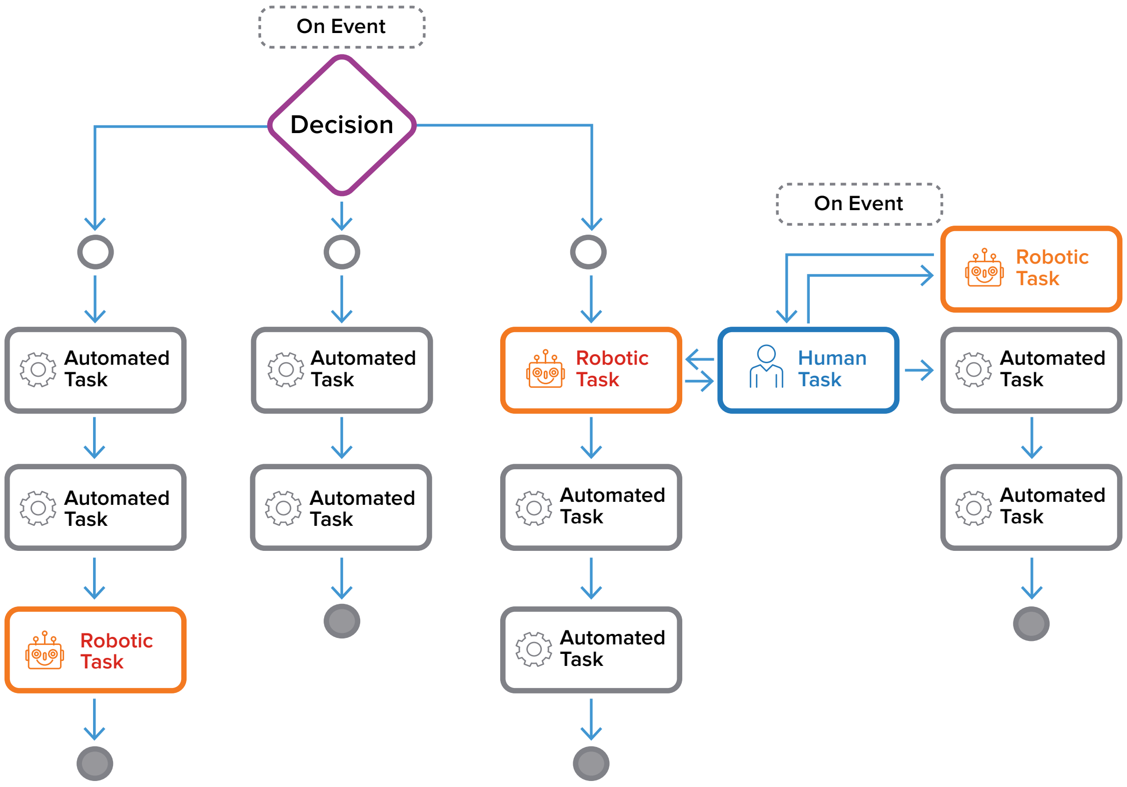
Multimodal Orchestration

An orchestration is the automated configuration, coordination, and management of executing work across a process. The work may involve receiving an order, processing the order, and updating all systems involved with the order. Orchestration is traditionally considered an entirely automated process, executed traditionally through the use of messaging, APIs, and integration.

A newer pattern called "multimodal orchestration" incorporates robotic tasks as a peer to integration and messaging. While minimizing manual tasks, multimodal also incorporates human tasks where needed. While not fully straight through, human tasks serve to advance the orchestration without going outside the orchestration core; **Figure 4** illustrates a multimodal orchestration design pattern. It is considered multimodal because it combines different technologies working together to execute the orchestration.

FIGURE 4

Multimodal Orchestration Design Pattern



Source: IDC, 2022

Orchestration is stateful to ensure each step in the orchestration is executed properly. In **Figure 3** (page 10), execution of the orchestration is managed and governed by orchestration management to ensure each element is completed successfully. There are different approaches to orchestration management, depending on the tools used.

The pattern supports the use of decisions to route the orchestration based on conditions that are established either through a third-party decisions service that can be called as part of an automated task or through simple rules logic that directs the route of the orchestration. The automated tasks of orchestration increasingly use APIs, replacing the use of messaging. Not included in **Figure 4** (previous page) is the inclusion of compensation logic to reverse out all or part of the processing where orchestration involves a change or cancellation.

Background

Orchestration was originally implemented using messaging middleware, particularly for straight-through processing of transactions. Today, messaging-based orchestration continues to be used for high-volume, high-speed transaction processing. Integration platforms also commonly manage the development and management of orchestrations. Business process modeling notation (BPMN)-based process execution engines also support orchestration. As enterprises increase their efforts to achieve end-to-end automation — or straight-through automation — a variety of technologies are now involved with an orchestration. These may include robotic process automation (RPA), intelligent document processing, API management, and AI and ML models, and there may be some human-in-the-loop tasks. Because orchestrations increasingly utilize a variety of technologies for execution, we call this design pattern multimodal orchestration.

An orchestration may be scheduled or may be triggered. End users may also be involved with triggering an orchestration from an on-screen action. In this case, there may be a simple multistep orchestration executed in the back end to deliver a result to the user interface (UI) or the action may result in the execution of a complex orchestration with no results returned to the UI. A customer, for example, may go to the self-service website of a service provider and enter in a change of address, which will trigger an orchestration to perform automated updates to all systems impacted by the change. In fact, the self-service site may have a variety of services that can be initiated by a customer or employee that will trigger separate orchestrations.

Use Cases

Multimodal orchestration use cases include:

- **Process automation.** Orchestration is used for end-to-end process automation, including customer onboarding, employee onboarding, and order processing, particularly where there is an effort to standardize, speed up cycle times, and reduce manual data entry. There are also efforts to identify subprocesses that can be automated via orchestration.
- **Multicloud integration.** Orchestration is used to automate the process between two or more applications or sources of data regardless of where the applications are deployed.
- **Transaction processing.** Orchestration is used to process a transaction by receiving the data and performing sequential and parallel updates to all applications impacted by the transaction. Transaction processing must also create compensation logic to reverse changes when all or a part of a transaction is canceled.

Smart Contracts and Blockchains

One variation of orchestration is the use of smart contracts with a blockchain. Smart contracts consist of code stored on a blockchain that runs when predetermined conditions are met, such as filling out an order form and pressing the submit button. Smart contracts can also automate a workflow, triggering the next action when conditions are met. The code may be embedded as part of an action that, when invoked from a user interface, will trigger a transaction. When the transaction is completed, the blockchain is updated. The blockchain itself is a peer-to-peer protocol for trustless execution and recording of transactions secured by cryptography in a consistent, immutable chain of blocks.

For business-to-business transactions, shared ledgers are updated only after the transaction is validated by the relevant participants involved. Once approved, a block is automatically created across the ledgers of all participants. Every network partner sees the same unalterable set of transactions, including compensating transactions.

The value of this approach is the ability to execute transactions across an ecosystem; each individual member is able to have the same view of what happened as everyone else.

Digital Assistants

Orchestration is also used in combination with digital assistants to replace support emails and provide self-service options for employees and customers. With a digital assistant, a user types in a question in a conversational style or talks to the assistant. Conversational AI is used to categorize the question and extract meaning from the conversation to invoke some type of automated response. This may be a simple interaction or more complicated information lookup. Each leg of a question-and-answer session may involve an orchestration to collect and deliver each answer. When the session is completed, any commitment made by the digital assistant may trigger an orchestration to execute what was agreed to in the conversation. We expect digital assistants to increasingly be able to make commitments, and orchestration will become an important way to meet the commitment through automation.

Importance of Multimodal Orchestration Pattern

If automation is important to an enterprise, then orchestration is foundational. Multimodal orchestration is an end-to-end way to implement automation using a variety of technologies working together to execute steps in the larger orchestration design.

Multimodal Automation Platforms

At IDC, we help clients understand what their automation capabilities should consist of using a framework called intelligent process automation (IPA), which combines the various automation technologies with the use of AI and ML. Today, enterprises and vendors all generally recognize that automation consists of many technologies used to automate, improve processes, and operate efficiently and effectively. Because of these benefits, we are seeing a push by vendors to combine technologies of IPA under a common platform.

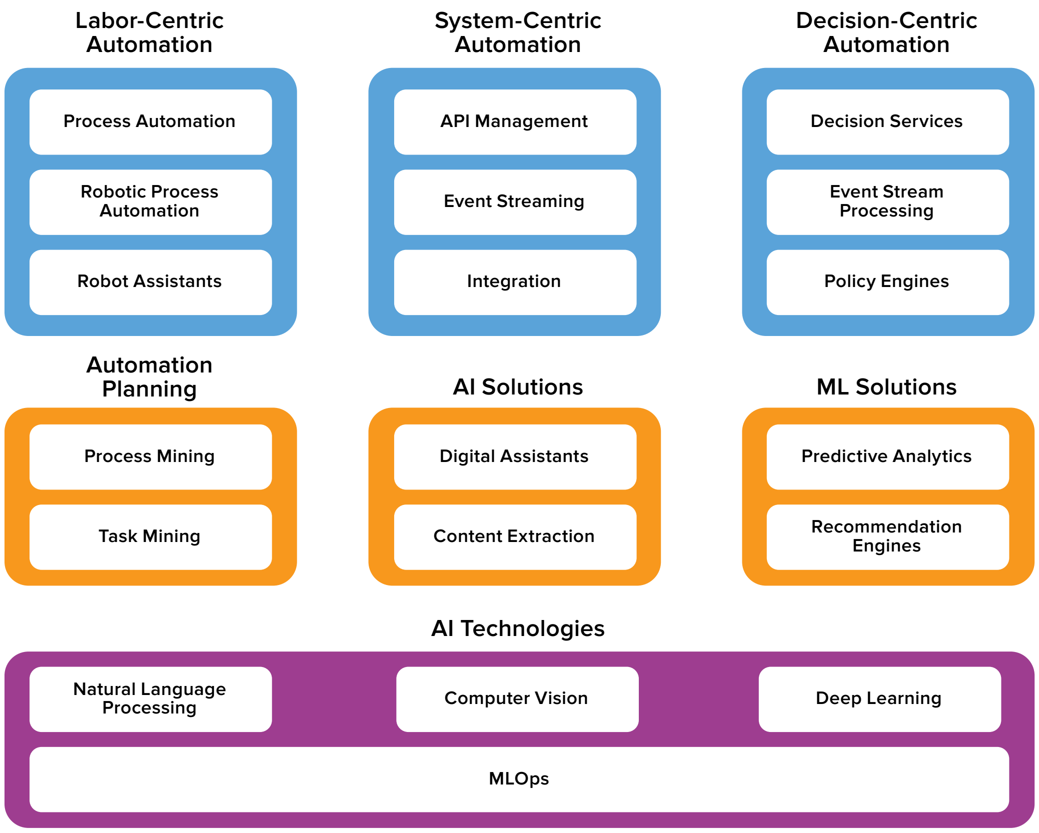
Figure 5 (next page) provides a view of IDC's IPA framework by type of automation and AI-centric offerings. No one vendor incorporates all of these capabilities in a single platform, and, clearly, there is no one vendor that has best-in-class capabilities in each area at this point. The IPA framework is organized into different classes of automation and process improvement technologies:

- **System-centric automation.** Technologies that support straight-through automation, connectivity, event-driven design, and API-based connectivity.
- **Labor-centric automation.** Technologies that improve labor productivity by replacing manual tasks with automation or improve the efficiency of completing work (RPA and process automation are the two largest categories.)
- **Decision-centric automation.** Automates decision making and improves worker decision making and includes decision services, rules and policy engines, and event stream processing.
- **Automation planning.** Use of process and task mining to collect data about how processes work to speed up automation development.
- **AI/ML solutions.** Includes digital assistants, content extraction, recommendation engines, and predictive analytics (These are used in combination with automation technologies to broaden out and improve the quality of the overall automation portfolio.)
- **AI technologies.** Collection of AI technologies for data scientists.

Vendors are increasingly consolidating technologies across these six major categories to build out comprehensive portfolios that can be used broadly for automation. Over time, this consolidates onto a unified platform, which we called multimodal automation platforms. The goal of this effort to consolidate IPA technologies is aimed at simplifying development, change management, and overall management of solutions in production using the platform.

FIGURE 5

Intelligent Process Automation Technologies



Source: IDC, 2022

New Roles and Styles of Development

Not only do multimodal automation platforms mix technologies for digital enablement, but the development environments are increasingly layered to support traditional and new types of development. **Figure 6** (page 18) shows examples of capabilities to illustrate how platforms are evolving to support the different developer personas. These changes to who develops and how development is evolving are accelerating for several reasons, including:

- An increasing number of college graduates who supplemented majors with computer science classes is now large enough to provide a meaningful developer-ready workforce, where workers are able to supplement their jobs with part-time digital enablement as a new area of responsibility.
- AI development requires data scientists and AI software engineers, creating demand for a different type of development due to important and disruptive emerging technologies.

- The ongoing demand for digital enablement of all types is greater than the supply of skilled developers, with the scarcity predicted to worsen over the next five years. The need to supplement skilled developers with supplemental development using the newer, simpler studio environments improves the overall capacity to meet demand for digital enablement.
- Time to market is also a critical driver as competitive forces and closing the gap between an idea and execution need to narrow. Rebuilding development capabilities to support high-productivity, high-speed development is changing technology choices. We are now beginning to see a prioritization around the speed and ease of development becoming a higher priority than actual runtime performance.
- With developer scarcity, there is also a greater focus on preventing highly skilled developer turnover by aligning developers with work that best uses their skills. Part of this is an effort to separate tactical development from strategic development. Codeless or minimal coding environments are seen as the development vehicle for tactical development, with integrated development environments the domain for strategic development.

Roughly 55% of spending on multimodal automation platforms offer high-productivity development studios for professional developers who can produce the bulk of the work without using code for development. The developers also typically provide some level of scripting and development language support to build custom capabilities. This is the middle tier of **Figure 6**, (next page) where developers focus on building out sophisticated solutions as well as use code to build custom components that can be packaged and added to the platform libraries available to peers or the business users in the top layer of **Figure 6**.

A studio environment optimized for business users is also part of the top layer of **Figure 6**. The objective is to provide a highly simplified — or consumer-grade — development experience suitable for trained business users to work together to build their own capabilities with minimal support from professional developers.

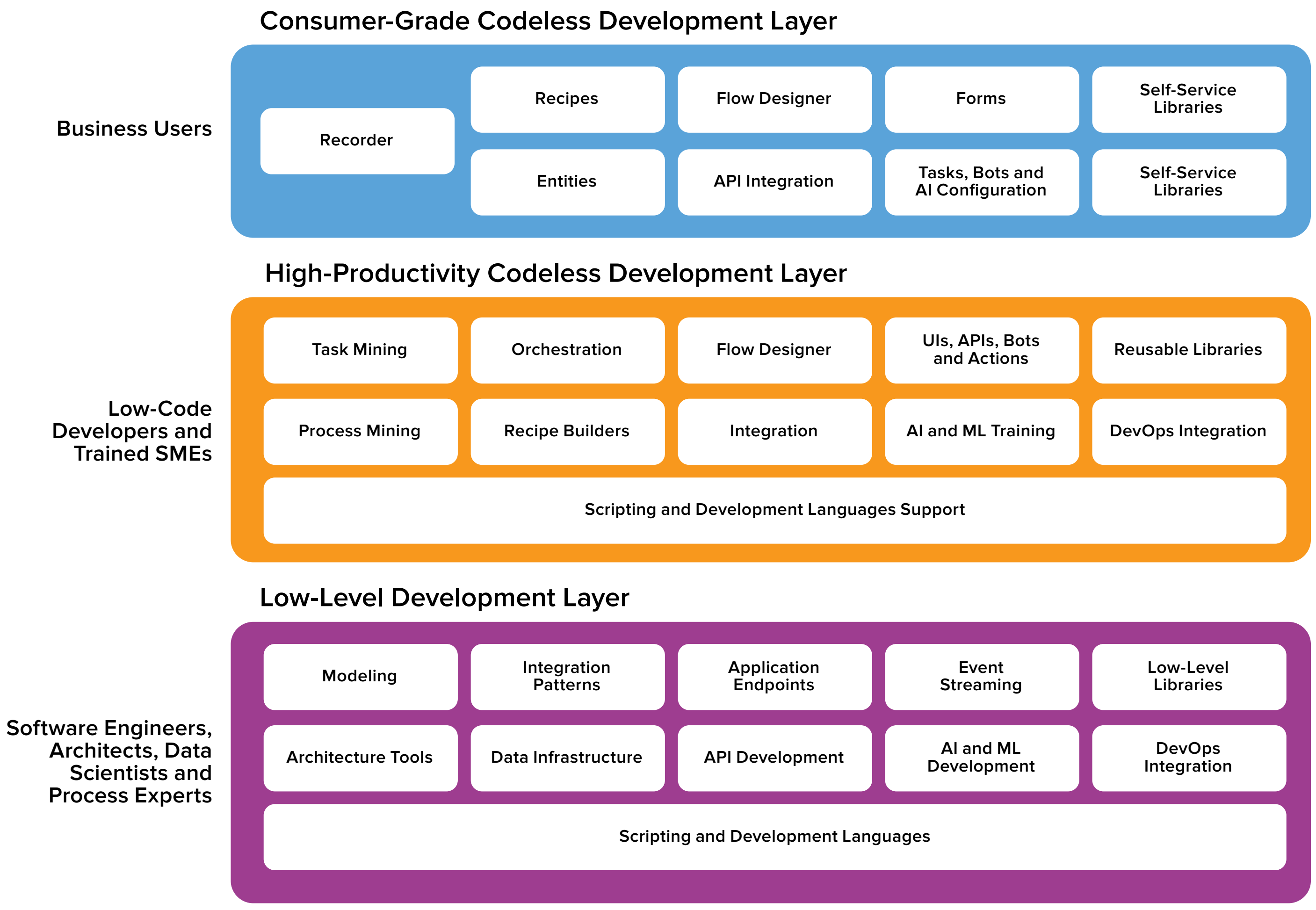
There also continues to be lower-level platforms that require software engineers to use specialized development environments or traditional integrated development environments (IDEs) to build capabilities that run on different types of automation platforms. These are used to build and configure lower-level capabilities, such as event streaming, messaging, orchestration, automation, data infrastructure, and advanced application development. Depending on the technology, some of these platforms offer a layer for collaborative development between business users and professional developers, while some are bypassing and moving directly to support business user development.

Roughly 55% of spending on multimodal automation platforms offer high-productivity development studios for professional developers who can produce the bulk of the work without using code for development.

As vendors build out the different capabilities needed per persona, each lower-layer capability is abstracted and leveraged as services, actions, automations, and so forth in higher layers. Security, access rights, integration with DevOps, and overall governance as well as runtime operations and management are part of the all-inclusive platform.

FIGURE 6

Multimodal Automation Studios: Key Capabilities and Developer Personas



Source: IDC, 2022

Oracle Cloud Integration Services

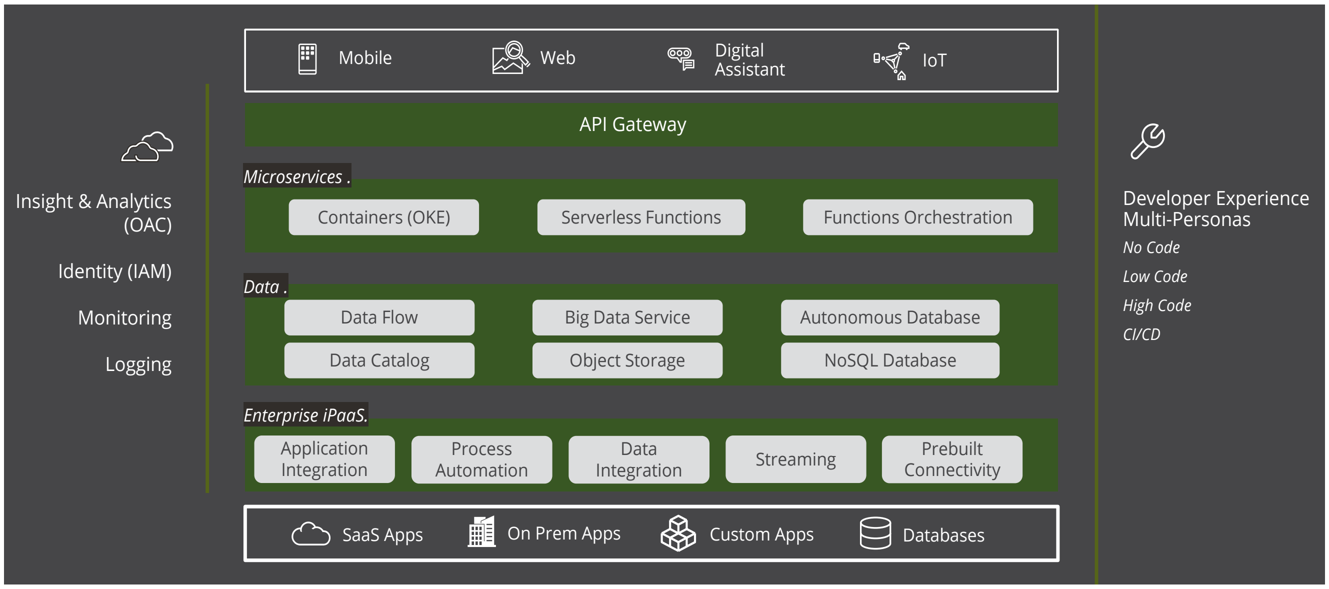
Oracle Cloud Infrastructure integration services connect any application and data source to automate end-to-end processes and centralize management. The broad array of integrations, with prebuilt adapters and low-code customization, simplify your migration to the cloud while streamlining hybrid and multicloud operations. This includes the ability to use prebuilt connectors and a SOAP-to-REST adapter to leverage existing client/server assets running in datacenters. Organizations use Oracle's integration templates, recipes, and core services for more advanced use cases such as combining streaming data with AI and ML modeling to drive high-speed insights from modern lakehouse analytics to optimize human and automated workflows.

Oracle capabilities in **Figure 7**, together with open source services and Oracle's ecosystem of partners, align with IDC's multimodal automation platforms and IPA technologies framework in **Figure 5** and support the modern integration design patterns explored in this white paper.

OCI integration services are part of a broader set of OCI services for multicloud modernization that accelerate the delivery of these and other enterprise integration design patterns.

FIGURE 7

OCI Integration Services



Source: Oracle, 2022

Recommendations

Select Automation Platforms to Fit Strategy, Culture, and Existing Investments

Because of the aggressive buildout of multimodal automation platforms over the past 18 months, enterprises have had to rethink their multicloud integration strategies. Platform selection involves picking one or more core technology-specific platforms and supplementing with secondary capabilities or picking a central multimodal platform. For example, core platforms such as an iPaaS are used for a broad set of integration capabilities. In a real sense, iPaaS is multimodal. Yet adoption of data integration platforms, especially for ELT or extract, transform, and load (ETL), an essential part of multicloud modernization intiatives. Enterprises should expect a full collection of integration technologies to be offered through a single enterprise iPaaS (eiPaaS).

Multimodal automation platforms continue to maintain separate development environments for each type of IPA technology but can mix them together in production to create a more comprehensive solution under a uniform operating environment.

To assess whether to focus on strategic narrow-purpose automation platforms or multimodal capabilities, evaluating the feature sets of both types of platforms is critical. If RPA is strategic and the multimodal platform does not have sophisticated capabilities, the multimodal won't be as effective as best of breed. But a multimodal platform that partners with best of breed while being best of breed itself in other areas makes perfect sense.

It is also important to look at existing automation investment to determine gaps and alignments. Meanwhile, innovation will continue by both well-established and emerging vendors, making it important to evaluate the ease of interoperability of the platform as well as how the vendors under consideration partner and operate within an ecosystem.

Align Platform Choices by Also Factoring in Developer Persona Plans

IDC estimates that roughly 30% of enterprises have begun experimenting or are successful with line-of-business developer programs. Many more are considering moving to this style of development. Working for an organization that is considering this style of development must factor in the cost and speed of training required to develop on any of the platforms under consideration.

This is also true for the high-productivity, low-code developer. If a development team does not have enough developers to meet demand, separating out by the type of development that can be done via codeless studios versus traditional development is becoming much more critical. Both types of development have time-to-market service-level agreements (SLAs) that put pressure on all types of developers. Doing the correct alignment of features suitable for personas is an important factor in platform adoption.

Architecture and Design Patterns Matter

Cycle times are speeding up in business, and executive and domain leadership are increasingly focusing on operating their core business processes in a more holistic end-to-end manner, especially rapidly identifying how the impact of an error or slow operating speed upstream will impact customers downstream. Speed to market and continuous improvement are increasingly core values of operational effectiveness.

Design patterns are an important way to deal with speed and complexity because they can be leveraged and extended into new projects running across different regions, different areas of a business, and different deployment locations. They are able to both span and extend multiple areas of a business, providing a way to speed up digital enablement that aligns with how the enterprise needs to operate.

Meanwhile, development and architecture teams that haven't adopted newer technologies that align with multicloud, distributed computing, event-driven design, and mixtures of technology to support end-to-end automation should begin experimenting with the core technology. Partnering with business teams focusing on process improvement is a good place to look for opportunities, especially to leverage this technology to move into the modern design patterns outlined in this paper.

As design patterns embrace different technologies that must work well together, enterprises are looking for ways to reduce complexity. Multimodal automation platforms outlined in this white paper will increasingly be adopted to reduce complexity.

Look for multimodal automation platforms that support:

- Event-driven design, an enterprise iPaaS, a broad set of automation capabilities, and the ability to effectively manage API life cycles (All of this is needed to build the modern integration patterns outlined.)
- Multiple developer personas, including business users, analysts, and the professional developers as well as support for collaboration across business and technology teams.
- A healthy ecosystem of partnerships, especially as technology rapidly evolves (Vendors that partner and manage a strong ecosystem provide a better way to leverage new technologies or narrower technologies that are more specific to an industry or a particular use case, especially where it is needed to fully execute a design pattern.)

About the Analyst

Maureen Flemming
Program VP, Worldwide Intelligent Process Automation
Market Research and Advisory Service, IDC

Maureen Fleming is Program Vice President for IDC's Intelligent Process Automation research. In this role, she focuses on a portfolio of technologies used by enterprises to speed up, drive cost out of, and support a customer-centric approach to business operations. She especially focuses on the convergence of AI, machine learning, and automation and how that combination changes the economics and benefits of process improvement.

More about Maureen Flemming

IDC Custom Solutions

This publication was produced by IDC Custom Solutions. As a premier global provider of market intelligence, advisory services, and events for the information technology, telecommunications, and consumer technology markets, IDC's Custom Solutions group helps clients plan, market, sell and succeed in the global marketplace. We create actionable market intelligence and influential content marketing programs that yield measurable results.

IDC

@idc @idc idc.com