

The Application Factory

A comprehensive approach to building, modernizing, and integrating purpose-built applications

Five Key Reasons to Implement the Application Factory

Organizations have struggled for decades with effective software development and delivery. Today, the challenges of unbridled application proliferation are exacerbated by the need to consolidate and streamline while continuing to develop new capabilities to meet ever-changing business demands. The average large organization reported 976 individual software applications in the 2022 Connectivity Benchmark Report published by MuleSoft Research.¹ A consistent and automated approach to application development is necessary to reduce cost and complexity, while increasing operational agility and application quality.

Current disconnects among business needs, operational requirements, and technical constraints stem from a plethora of root causes related to underlying architecture that predates many of today's technologies and requirements. The resulting challenges include issues with security, quality, reliability, usability, scalability, interoperability, maintainability, and portability. Additional contributors to these challenges include variations in software development processes, tools, and languages; the lack of a single source of truth for underlying data; the absence of automated integration between software applications; and the underlying complexity of software applications we all use every day.

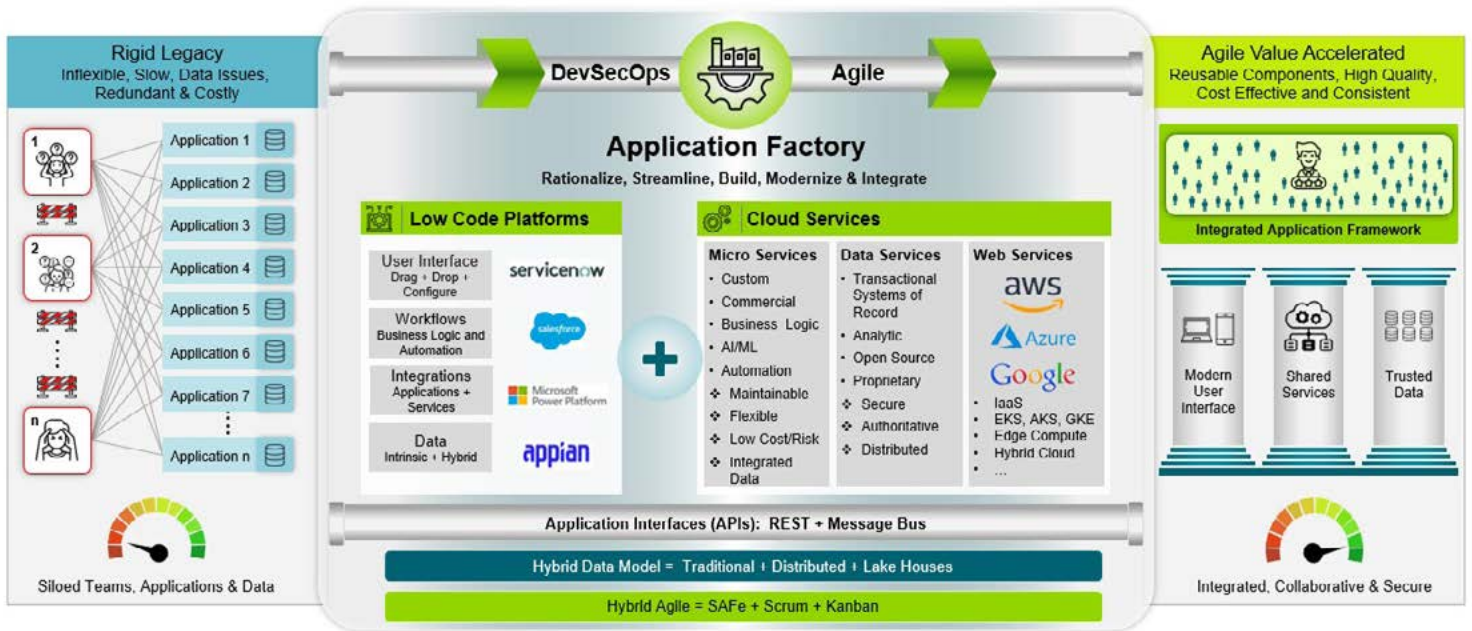


The symptoms of these issues include:

- An inability to respond quickly to changing business demands.
- A lack of visibility into numerous attack vectors that compromise security.
- High costs just 'keeping the lights on' for operations and maintenance.
- Inefficient user experiences overly dependent on swivel-chair manual processes.
- Inconsistent data and duplicate databases, creating silos and blame.

To meet these challenges, many enterprises are turning to the concept of an application factory. Application factories deliver significant benefits to large organizations in terms of agility, productivity, scalability, and cost-effectiveness. Guidehouse's innovative approach combines best-in-class modern low-code platforms and web services. This proven methodology has helped multiple federal agencies, deliver highly successful, award-winning programs that exceeded expectations throughout the software life cycle.

The Application Factory



The auto industry learned more than a century ago that independent function-built components have many advantages over components that serve multiple purposes. A loosely coupled architecture of small reusable components, capable of interacting via standardized interfaces, made it possible for the same component to be used on multiple cars, simplified final assembly, allowed for the outsourcing of work to more cost-competitive suppliers, and created an entire replacement parts industry that resulted in higher quality for end users and higher profits for car manufacturers. The same benefits come from applying a true component-based approach to software application architectures as described so eloquently 20 years ago in Clemens Szyperski's book, *Component Software: Beyond Object-Oriented Programming*², which was foundational in shaping the modern concept of web services.

Modular components, and the development process that creates them, allow teams or individuals to work independently on smaller, manageable components—and for those components to be updated independently in production with a minimal impact footprint. This decoupling of services facilitates faster development cycles, shorter time to market, and the ability to iterate and adapt applications quickly. Matthew Skelton and Manuel Pais call this model X-as-a-service (XaaS) in their book, *Team Topologies*³. Of the three models they consider, it is the best at supporting disconnected development teams that are part of different departments or even different organizations, as is the case when consuming commercial web/data services that provide little to no interaction between service development teams.

Service design requires an intimate understanding of multiple use cases, clear boundaries (which modern representational state transfer (REST) APIs provide) and automated tests to ensure that changes to the service do not adversely impact end users. One drawback of a 100% component-based (or, in modern terms, web-service-based) architecture is the added burden of system integration. Guidehouse's application factory model addresses this by using standards-based integration patterns and leveraging low-code application platforms.

Low-Code

Low-code platforms can significantly reduce the need for custom integration because all the components and modules provided with the platform are designed to work together. This platform-as-a-service (PaaS) approach provides visual development interfaces, prebuilt templates, and drag-and-drop functionality, enabling citizen developers and IT professionals to create new applications and extend existing ones with minimal coding effort. In fact, the move from custom coding to a ‘select-and-configure’ approach is widespread through all sections of software development. It started with shared libraries back in the early 1970s and continues today with the emphasis on user interface configuration to shape behavior as an alternative to custom code.

The democratization of development provided by low-code platforms allows organizations to optimize their resources by involving business users in application development, reducing the burden on IT teams, and accelerating time to market. Two of the leading platforms ServiceNow and Salesforce both started as purpose-built applications but quickly evolved into application platforms, while others, like Appian and Microsoft Power Platform, were created to be general-purpose application-building platforms.

Low-code platforms do have their drawbacks, including vendor lock-in and a monolithic nature. Guidehouse’s application factory model addresses this by combining the flexibility and customizability of web service components with the agility of low code, providing a ‘right tool for the job’ best fit.

Component-Based Services

Web service-oriented component architectures play a vital role in enabling scalability and resilience within an application factory. Large organizations can scale individual services based on demand, allowing them to handle high volumes of users and data without compromising performance. Additionally, these services offer fault isolation, preventing failures in one service from affecting the entire system.



This modular approach to development enhances the robustness and resilience of applications, ensuring continuous availability and reducing the risk of widespread failures. Applying the application factory to deliver architectures that allow web services to meet specific business needs, while integrating those services with low-code application platforms, achieves agility and reuse without sacrificing flexibility.

AI/ML Web Services

There are numerous web services available from Amazon Web Services (AWS), Microsoft Azure, and others that can be used as key components of your applications. These include many prebuilt artificial intelligence services (AIaaS) and machine learning services (MLaaS). One example of the hundreds of AI services available is Amazon Polly, which uses deep learning capabilities to generate speech that sounds like a human voice, providing your applications with the same technology used in Amazon’s Alexa. SageMaker is a powerful MLaaS for building, training, and deploying machine learning models, and Azure OpenAI Service includes large language generative AI services with enterprise-class controls and security.

These represent a very small sample of the thousands of web services available. When an existing web service cannot meet a specific and/or proprietary need, microservices can fill the gap.

Microservices

Microservices are reusable self-contained software components that handle unique business capabilities. They can be built, and function, totally independently of each other, which makes them ideal building blocks for custom development within our application factories.

The advantages of microservices include fault tolerance through isolation and redundancy. When built using containers (Docker/CRI-O) and Kubernetes (OpenShift, Rancher, Amazon Elastic Kubernetes Service, Azure Kubernetes Service, etc.) microservices provide fault tolerance, security, and high availability, while overcoming many of the challenges of managing and orchestrating containerized services that can scale to hundreds or even thousands.

One goal of microservices is to encapsulate data along with business logic and algorithms. The containerization of databases is a powerful tool to address the need for a separate data store for each service. Guidehouse pioneered the containerization of both Relational Database Management Systems (RDBMS) and NoSQL document stores, with multiple federal agencies.

Unlike shared libraries (Java, C#, C++, and others), microservices are fully contained, loosely coupled, and easily integrated, upgraded, and replaced without requiring any rebuilding (compiling/linking) during the integration phase. This agility increases interoperability without having to deal with any special dependencies related to the connecting service or application. Language-specific shared libraries and modules are by no means eliminated using microservices. They are just contained within the microservice itself.

As the name 'container' implies, these services are isolated from other services and processes. A containerized architecture improves security, portability, testability, upgradability, understandability, scalability, and interchangeability, while reducing redundancy.

Data Services

There is a plethora of data available on the internet, including public data sets on AWS, Google Public Data, Snowflake Marketplace, Databricks Marketplace, and Data.gov.

In addition, proprietary data services can be developed and secured, providing data that serves multiple applications and even other services. While static datasets can be downloaded and consumed, the real power of data comes from using live datasets consumed through REST APIs. Webhooks provide notification of change events in a data set, triggering client services to perform the necessary actions in a prescriptive fashion, as opposed to a complete ETL (extract, transform, load), on a timebound schedule.

All data services offer the key advantages of providing up-to-date data and facilitating changes to that data for all who consume it. This single source of truth is vital for any data set that needs to be appended or updated. When required historical lookback can be provided on demand to provide detailed trend analysis.

As with any third-party tool or technology, make sure to read and review the licensing agreements and restrictions on the usability of these datasets, including the ones promoted as free/public, to avoid any legal issues in the future.

Unlocking Efficiency and Innovation

Foundational to a successful application factory is the right combination of people/culture, processes, and technology. Each is of paramount importance at a different phase, but this three-legged stool cannot stand without all three pieces in good working order.

Culture

A culture of respect, innovation, integrity, mentorship and excellence is vital in supporting rapid change and the consolidation of efforts required by an application factory. Blame is the enemy. As Dale Carnegie highlighted in the beginning of his 1936 book, *How to Win Friends and Influence People*⁴, when you instigate a cycle of blame you are causing more damage to yourself. The tendency for software teams to create enemies of other departments, unfamiliar technologies, and commercial software/service vendors is all too common. When problems arise, the appropriate individuals must first be held accountable for clearly

clearly defining the issue (ideally in an Agile tracking system). Then leadership must ensure the correct people are brought together to collectively resolve the issue.

Formal retrospectives are vital and must include all relevant stakeholders to be effective. Once again, these processes should not seek to assign blame but to identify successes, issues and corrective actions that should be taken in the future to prevent the issue from occurring again. Follow-up is required to guarantee that the issues have been addressed.

The division of responsibility between low-code developers and web service developers minimizes the need for excessive communication between teams. This is effective only if the customer and product owner for each component are well-defined. Frequent reminders of two of the laws of software development are vital in maintaining focus: First, know your customer and second, you are not the customer.

Agile methodologies and specifically DevOps processes provide codification and standardization of cultural best practices, shaping the guardrails for behaviors while providing automation that improves delivery velocity, security, and quality.

Process

By utilizing DevOps principles, such as continuous integration and continuous deployment (CI/CD), the application factory streamlines development, testing, and deployment workflows. This automation reduces manual effort, minimizes errors, and fosters collaboration among development, operations, and quality assurance teams. Consequently, developers can focus more on innovation, leading to accelerated delivery of high-quality applications.

DevOps practices bring developers, operations teams, and quality assurance specialists together, enabling seamless collaboration and shared responsibility throughout the application life cycle. This collaboration promotes knowledge-sharing, improves communication, and allows for rapid feedback loops, leading to faster problem-solving and innovation.

The application factory environment also encourages experimentation and the adoption of emerging technologies, driving continuous improvement, and enhancing the organization's competitive edge. Having isolated lab, dev, test, and production environments, at a minimum, is vital for reaching the level of quality, availability, and security that all modern software consumers demand.

As with any change to your organization, it is vital to start small, and gain trust and buy-in at all levels by delivering working software, or at least a demo, biweekly, or preferably weekly where possible. This approach offers great opportunities to solicit feedback from all stakeholders and continuously improve, building upon early wins by increasing velocity and value, along with the number of scrum teams.



Technology

Application programming interfaces (APIs) are the principal mode of communication between applications and component-based services. The primary method for consuming these APIs is representational state transfer. REST has many advantages, including its widespread popularity and the fact that it uses the most popular internet protocol, HTTPS. Any service or application that can access the internet can publish and consume REST APIs.

Message buses like Amazon Simple Notification Service, Azure Service Bus, and Kafka, to name just a few, provide an integration pattern that works much like an FM radio. Applications and services can tune into various channels and listen for broadcasts based on their interests. This is a great way to have one-to-many communications. One common use case is advanced notification of a shutdown of a service, so dependent applications are aware of what is coming and can respond accordingly by implementing backup/failsafe processes.

There are other integration methods—including direct database connections (for example, JDBC/ or ODBC); robotic process automation (RPA), which is a user-interface-based integration pattern; and legacy Simple Object Access Protocol (SOAP), and Remote Procedure Calls (RPC), etc. But REST and modern message buses have emerged as the preferred methods for long-term, persistent, mission-critical architectures.

REST APIs are best suited for request/response (GET/POST) interactions, and message buses are designed for notifications. Both have their place, but the most common integration patterns will primarily use REST when connecting

distributed applications and services, while using message buses between a shared framework or platform. In fact, all low-code platforms use message buses internally, although only the most advanced platform developers are aware of this. REST APIs remain the most frequently used integration method within the application factory.

Conclusion

The agility, productivity, scalability, cost-effectiveness, collaboration, and innovation fostered by the application factory provides a competitive advantage in today's digital age. By embracing this model, enterprises can meet the demands of a rapidly evolving market and deliver high-quality applications more quickly, maximizing time-to-value. The rapid delivery cycles built upon the foundations of Agile and DevOps methodologies provide improved alignment between customer needs and software delivery through a high-velocity feedback loop that takes the results from retrospectives conducted at the end of each sprint and feeds them into the very next sprint to support continuous innovation and improvement.

As the digital landscape continues to evolve at an exponential pace, the application factory is vital in helping large organizations rapidly deliver award-winning applications that exceed customer expectations.

¹ "70% of Organizations Do Not Provide Completely Connected User Experiences, 2022 Connectivity Benchmark Report", Mulesoft, 2022, mulesoft.com.

² Szyperski, Clemons, 1997, "Component Software: Beyond Object-Oriented Programming", Addison-Wesley Professional; 2nd edition (November 13, 2002).

³ Skelton, M. and Pais, M., 2019 "Team Topologies, Organizing Business and Technology Teams for Fast Flow", IT Revolution.


⁴ Carnegie, D., 2012 "How to Win Friends and Influence People", Arrow (Random).


Contacts

Jerry Eshbaugh, Director
Low-Code
jerry.eshbaugh@guidehouse.com

About Guidehouse

Guidehouse is a leading global provider of consulting services to the public sector and commercial markets, with broad capabilities in management, technology, and risk consulting. By combining our public and private sector expertise, we help clients address their most complex challenges and navigate significant regulatory pressures focusing on transformational change, business resiliency, and technology-driven innovation. Across a range of advisory, consulting, outsourcing, and digital services, we create scalable, innovative solutions that help our clients outwit complexity and position them for future growth and success. The company has more than 17,000 professionals in over 55 locations globally. Guidehouse is led by seasoned professionals with proven and diverse expertise in traditional and emerging technologies, markets, and agenda-setting issues driving national and global economies. For more information, please visit guidehouse.com.

 guidehouse.com/services/digital-technology

 linkedin.com/showcase/guidehouse-technology-solutions/

 [@GHtechsolutions](https://twitter.com/GHtechsolutions)