eBook

Digital Transformation: The Case for **Modernizing Legacy Applications**

Fiorano®

DISCLAIMER

No AI was used in the production of this work

TABLE OF CONTENTS

Audience:

- CIO, CTO, CDO
- IT Leaders
- Business Transformation Leaders
- Digital Business Leaders
- Digital Product Leaders

Objectives:

- Understand why applications need to be modernized
- Learn the risks of failing to modernize applications
- Discover how to modernize applications in a phased manner

Estimated read time:

6 mins

INTRODUCTION

Together with the rapid evolution of the digital landscape from a monolithic, on-premises-based server model to a microservices-based, cloud-native model – is the decision of whether to modernize legacy applications or to rewrite the apps from the ground up to fit the cloud-native model.

A Deloitte study found that organizations with the highest level of digital maturity reported a 45% increase in net revenue and a 43% increase in net profit margins compared with their industry peers. This comparison is based on industry averages and whether the companies invested in digital transformation initiatives, driving these organizations toward digital maturity.

Therefore, let's consider the case for modernizing legacy applications, or why organizations should decide to modernize legacy applications instead of rewriting from scratch by starting with the risks of failing to modernize these apps.

Note:
The concept of modernizing legacy applications does not include what is known as the "lift and shift" model, where monolithic applications are moved to the cloud in their original format.

THE RISKS OF FAILING TO MODERNIZE

Even though modernization is not without its challenge, the fact remains that it is no longer an option to retain enterprise applications in their legacy state.

Why?

Monolithic applications are traditionally built as single, large codebases, making them difficult, if not impossible, to scale, maintain, and adapt to evolving business needs. Failure to modernize these applications poses multiple risks, threats, and challenges for organizations, particularly in today's dynamic and competitive digital landscape, where cybercrime is rampant.

For instance, statistics quoted in the Cybercrime Magazine report that cybercrime will cost the world **$10.5 trillion** (USD) by 2025.

Let's add weight to this argument by looking at multiple critical risks associated with monolithic legacy applications, such as:

Scalability:
Monolithic applications are not dynamic. They do not scale quickly based on user demands. Thus, without modernization, organizations will struggle to provide a seamless user experience during periods of high traffic.

Resource inefficiency:
Monolithic applications tend to require more resources, resulting in resource utilization inefficiencies such as networking, computing power, memory, and storage, negatively impacting cost-effectiveness and operational efficiency.

Limited flexibility and agility:
Monolithic architectures are less flexible and agile than their modern, modular counterparts. Failure to modernize will result in a slower response to rapidly evolving business requirements.

Innovation:
Modern applications leverage microservices, cloud-native architectures, and DevOps practices to enable faster innovation cycles. Without modernization, organizations risk falling behind competitors who deliver new features and updates faster.

Maintenance burden:

As monolithic applications increase in size, complexity, and age, maintaining existing code bases is challenging, resulting in longer development cycles, higher costs, and an increased risk of errors.

Integration challenges:

Many legacy applications don't integrate with current technology or third-party services, making integrating with the latest innovative technology complicated.

Security risks:

Monolithic applications often lack security features and best practices to protect against modern cyber threats. Therefore, failure to modernize will expose the organization to a multitude of sophisticated security threats.

User experience and expectations:

Modern users expect responsive, feature-rich, easily accessible applications. Monolithic applications are none of these adjectives. Therefore, failure to modernize will result in a subpar user experience, driving users away.

A company's ability to innovate, respond quickly to changes in the market, and maintain a competitive edge will be severely disadvantaged by this endless list of risks. Modernizing efforts, such as transitioning to microservices, adopting cloud-native architectures, and implementing DevOps practices, will help mitigate these risks and position organizations for growth and success in the post-modern, digital era.

MODERNIZATION APPROACHES AND STRATEGIES

Gartner reports that 89% of all enterprise leaders surveyed state that digital modernization is embedded in their organizational growth strategies. However, only 35% of these organizations are on track to meet their digital transformation (modernization) goals.

However, modernizing legacy applications without a strategy is extremely risky and, as the adage, ***"fools rush in where angels fear to tread,"*** observes, akin to being a fool who rushes into situations where caution and good judgment are needed. Therefore, drawing up a strategy (or approach) by consulting with domain, product, and tech experts to understand modernization's technical, strategic, and business aspects is vital. And adopting a phased approach rather than diving into the deep end without a plan is just as important.

Let’s consider the use case of a public sector application that pays out disability grants. It is a monolithic legacy application, severely in need of modernization to become a citizen-centric application.

The research paper titled: ***“A Qualitative Study of Legacy Systems Modernization for Citizen-Centric Digital Government”*** notes that even though legacy systems are still valuable assets that have been in use for a long time, supporting service delivery to citizens, and fulfilling critical public administration functions and data, they also delay innovation.

As described above, maintenance is virtually impossible. However, due to their importance, these applications cannot be scrapped. They also cannot be moved (lift and shift) to the cloud in their current state. Finally, rebuilding them from the bottom up is also not feasible. It will take too long, cost too much, and critical service functionalities will be lost during the redevelopment process.

Therefore, the only way forward is to modernize this application in a phased manner as a cloud-native microservices application.

Note:

*This research paper points out that **"legacy systems modernization is a multi-faced task that includes not just technical, but also business, environmental, and organizational aspects."***

Modernizing legacy applications in a phased manner is a strategic approach that allows organizations to update and improve their existing systems gradually, minimizing disruptions and risks. Let’s see how:

- **Assess and analyze:** Evaluate the legacy application to understand its architecture, codebase, functionalities, and dependencies. Identify pain points, bottlenecks, security vulnerabilities, and areas that can be modularized and decoupled into microservices.

- **Define modernization goals:** Set clear objectives for modernization, such as improving scalability, agility, security, and developer productivity.

- **Select modules for microservices:** Divide the legacy application into modules or functionalities that can be encapsulated as individual microservices. Prioritize modules based on business value, technical complexity, and feasibility for microservices adoption.

- **Choose technology stack:** Select a technology stack and tools suitable for building microservices, like Docker containers, Kubernetes (orchestration tool), and service frameworks.

- **Design microservices:** Design the architecture of each microservice, defining their APIs, data models, communication mechanisms, and boundaries.

- **Develop microservices:** Develop and implement each microservice independently, adhering to best practices for microservices design, including loose coupling and API contracts.

- **Testing and Validation:** Test each microservice individually to ensure functionality, performance, and compatibility. Conduct integration testing to validate communication between microservices.

- **Incremental migration:** Gradually migrate legacy functionality to microservices while keeping the monolithic application operational. Implement communication patterns like API gateways to ensure seamless interaction between monolithic and microservices components.

- **Refactor and improve:** As you migrate functionality to microservices, refactor code to remove dependencies from the monolith to the microservices architecture.

- **Data management:** Plan data migration or synchronization strategies to ensure data consistency between the monolith and microservices.

- **Scale and optimize:** Leverage the scalability of microservices to handle increased traffic and demand. Optimize microservices for performance and resource efficiency.

- **Feedback and iteration:** Gather feedback from users and stakeholders to fine-tune the modernized application. Continuously iterate on the microservices to improve their capabilities.

- **Complete migration:** Gradually migrate all relevant functionality from the monolith to microservices. Monitor the overall system performance and ensure smooth operation.

- **Decommission monolithic application:** Once all critical functionality is migrated and tested, decommission the monolithic application.

- **Continuous improvement:** Continuously monitor, analyze, and optimize the microservices architecture for performance, reliability, and security.

Modernizing a legacy application such as our public service application paying out disability allowances from a monolith into a citizen-centric microservices-based architecture requires careful planning, coordination, and execution. Moreover, this phased approach allows you to balance modernization with ongoing operations while gradually reaping the benefits of a modernized, more flexible, and scalable architecture.

IN CONCLUSION

Embracing the post-modern technological evolution is mandatory. Part of this evolution is the cloud-native movement, including the imperative to move applications from on-premises data centers to the cloud.

Moreover, it is clear from this discussion that the decision is not whether to move these monoliths to the cloud but whether to modernize them into microservices architectures that fit the cloud-native model or replace them with new applications.

ABOUT FIORANO

Enabling Change at the Speed of Thought

Fiorano is a cloud-native event-driven microservices platform that combines integration, low-code, and eiPaaS capabilities to build and deploy global hybrid multi-cloud applications.

By making business processes event driven, Fiorano helps enterprises achieve massive scalability, responsiveness, and increased productivity.

With Fiorano, companies can respond better in volatile markets and deliver exceptional customer and employee experiences.

Drop us a line:

info@fiorano.com

www.fiorano.com | Fiorano Software | @FioranoGlobal