Evaluating the Importance of Legacy System Platform Modernization

A Research Paper submitted to the Department of Engineering and Society

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Matthew Y. Samuel

Spring 2024

Advisor

Richard D. Jacques, Ph.D., Department of Engineering and Society

Introduction

Organizations and businesses are today using many legacy system platforms nationwide. A legacy application system is a system that is made up of old software and/or hardware, and "researchers estimate that 180-200 billion lines of legacy code are still in use today" (Fanelli et al., 2016). These legacy systems are still being used by organizations today since they fulfill their intended purpose. However, with how fast business needs, consumer needs, and software are evolving now, maintaining legacy system platforms is not viable for multiple reasons. First, these systems are expensive to maintain due to technical debt increasing the maintenance costs for these platforms. Also, these systems are inefficient, which impacts employee productivity and creates a negative experience for the customer in some way. Finally, these legacy system platforms are not compatible with newer systems.

For example, in June 2019, the U.S. Government Accountability Office (GAO) identified 10 legacy systems that are essential to 10 government agencies that needed modernization. These legacy systems ranged from 8 to 51 years old and utilized outdated languages, which makes it more difficult to find someone who has the skills necessary to maintain these systems; unsupported hardware and software; and known security vulnerabilities. According to these agencies, these systems collectively cost about $337 million to run and maintain yearly. GAO found that out of these 10 agencies, only seven had modernization plans for these systems, and out of these seven agencies, only two had complete plans that adopted best practices. GAO conducted this study because each year, the federal government spends over $100 billion on "IT and cyber-related investments", which includes about 80 percent which goes toward the operation and maintenance of existing systems, including legacy systems (U.S. Government Accountability Office, 2023).

Throughout this paper, I highlight the various methods that can be utilized by organizations to modernize their legacy system platforms. Furthermore, I utilize a study conducted by Batlajery (2013) that shows the differing perceptions of legacy system platforms and their modernization between people in academia and people in industry to demonstrate that there should be more education on legacy system modernization. In this paper, I argue that although challenging, costly, and time consuming, organizations must modernize their legacy system platforms to keep up with the rapidly evolving needs of their business and their customers.

Part I: Strategies for Legacy System Platform Modernization

The majority of today's legacy system platform modernization efforts involve shifting from old monolithic applications to newer microservices. In fact, in a survey conducted in 2020 by O'Reilly, an online learning platform for technology and business, 77 percent of participants had adopted microservices, and 92 percent of them were experiencing success with microservices ("O'Reilly's microservices adoption in 2020 report", 2020). In this section, I define what microservices architecture is, including its purpose, its defining characteristics, and its advantages and disadvantages. Furthermore, I discuss several ways in which microservices can be utilized by organizations to effectively modernize their legacy system platforms.

Defining the Microservices Architecture

An application that utilizes the microservices architecture is "a distributed application where all its modules or elements are microservices and can be run independently" (Velepucha & Flores, 2023). Each microservice in an application that takes advantage of this architecture is developed independently of the others and is developed to fulfill a specific business functionality. These microservices are designed to be loosely coupled and highly cohesive, meaning that the microservices should be as independent of each other as possible and that each microservice should have a single responsibility. Since these microservices are independent of each other, they must communicate over a messaging system (Velepucha & Flores, 2023).

The microservices architecture evolved over time in response to the monolithic architecture in which the application can contain one or more modules, with the constraint that the application has a single executable, meaning that these modules cannot be run independently. Therefore, these monolithic applications usually have a single code base that houses all the application's services and functionalities (Velepucha & Flores, 2023). The two figures below provide visual representations of the monolithic and microservices architectures, respectively.

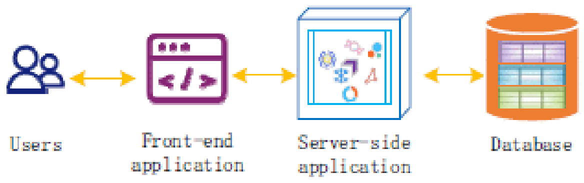

Figure 1: Monolithic architecture. The user interacts with the front-end application, which redirects the user's request to the server-side application that interacts with the database (Liu et al., 2020).

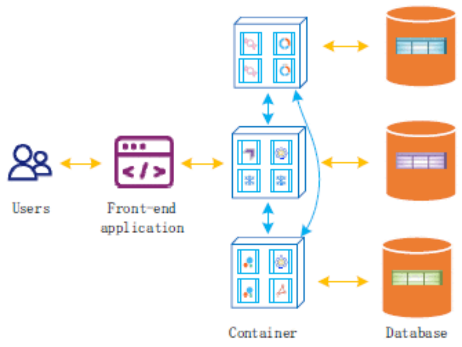

Figure 2: Microservices architecture. The user interacts with the front-end application, which redirects the user's request to the appropriate microservice, each of which has its own database that the other microservices cannot access (Liu et al., 2020).

For smaller applications that do not have many functions, organizations may prefer the monolithic architecture to the microservices architecture. This is because the development, deployment, testing, and maintenance of these smaller applications is much simpler when they are treated as a single block rather than as many independent components. However, as an application becomes larger and more complex, monolithic architecture quickly becomes a bad option for many reasons.

First, if one part of the application fails, the entire application fails. Second, the application becomes hard to maintain. For example, if an error occurs within the monolithic application, it is difficult to identify this error and then evaluate and publish a fix for it. Another drawback of monolithic applications is their scalability. Since these are large applications, scaling them up to improve their ability to manage a larger load, number of users, amount of data, or other resources without a significant decrease in performance or reliability becomes expensive and time-consuming since it requires increasing the capacity of the application's hardware and/or software. Additionally, in the case that only one or a few modules in the application must be scaled, the entire monolith must be scaled (Velepucha & Flores, 2023).

This is where the microservices architecture comes in. Since microservices are small components that run independently, they are easier to maintain. For example, if an error occurs within an application that utilizes the microservices architecture, it is usually easier to find the bug and fix it. Additionally, since microservices are independent, when one of these components in an application fails, the entire application does not fail (Velepucha & Flores, 2023).

Furthermore, a major advantage of the microservices architecture involves the teams within an organization or business that work on these microservices. Since these microservices are independent, teams can take ownership of microservice(s). This allows teams to work independently on these microservices, which leads to quicker development and more frequent deployments of these applications because each team specializes in the microservice(s) that they work on (Velepucha & Flores, 2023). I was able to see this particular benefit of the microservices architecture at play while interning at CarMax last summer. The team that I was on oversaw the development and maintenance of CarMax's online Document Center, which was the small portion of their site that enabled both the customers to upload stipulation documents for approval before being able to purchase a vehicle as well as the employees to review and approve these documents. The team knew the ins and outs of this portion of the site, which allowed them to release new features and fix bugs more rapidly, and their knowledge of this section of the company's site increased during each iteration of development.

Utilizing Microservices to Modernize Legacy Systems

According to a survey of 800 CIOs conducted in 2018 by Dynatrace, a digital performance management company, "76% of organizations think IT complexity could soon make it impossible to manage digital performance efficiently" ("76% of cios", 2018). Their research reveals that a single mobile or web transaction now uses an average of 35 different technology systems or components, compared to 22 five years ago. Furthermore, 88 percent of CIOs stated that they will have the microservices architecture adopted in their technologies within a year ("76% of cios", 2018). Based on this research and the research on the differences between the monolithic and microservices architectures, it is clear that organizations should phase out their older monolithic applications and transition to newer applications that utilize the microservices architecture, especially as the complexity of technologies that organizations and businesses use increases exponentially. The question becomes how they should go about doing this, as there is a plethora of modernization techniques.

Colanzi et al. (2021) present an eight-step roadmap for modernizing legacy systems with microservices. The eight steps are to analyze the driving forces, understand the legacy system, decompose the legacy system, define the microservice architecture, execute the modernization, integrate the microservices and the legacy, verify and validate the microservices, and monitor the microservices and infrastructure. For each step in the process, survey questions were presented to a group of 56 practitioners to gain insight into the following three research questions: “Why do companies migrate monolithic legacy systems to microservices?”, “How do companies perform the migration of monolithic legacy systems to microservices?”, and “What are the aspects of data persistence considered in the modernization of legacy systems with microservices?” (Colanzi et al., 2021).

To answer the first research question, Colanzi et al. (2021) learned that the most common driving forces for legacy system modernization are “easier maintenance and evolution”, “optimized scalability”, and “independent and automated deploy”, with each of these three reasons being mentioned as the main driving force by over 70 percent of participants. Furthermore, to answer the second research question, the results of the survey show that about 50 percent of participants stated that they conducted an incremental migration from monoliths to microservices and decomposed the legacy system by business capabilities. When deciding to define microservices, scalability, requirements, and reusability were the most popular criteria among the candidates participating in the survey. 64.5 percent of participants stated that their company used more than one programming language per project, and 97.1 percent of companies used cloud services, mainly Amazon Web Services (AWS). Lastly, to answer the third research question, it was found that 57.1 percent of companies decided to decentralize their database, 73.1 percent of companies took database transactions into account when designing the new microservices architecture, and 50 percent of companies reengineered their databases to achieve better outcomes in their modernization efforts. Additionally, it was found that the performance of the database was a concern for 57.7 percent of the companies (Colanzi et al., 2021).

In another study, De Lauretis (2019) presents a modernization strategy for organizations looking to migrate from a monolithic legacy system platform to an application with a microservices architecture. This strategy consists of five phases, which are shown below.

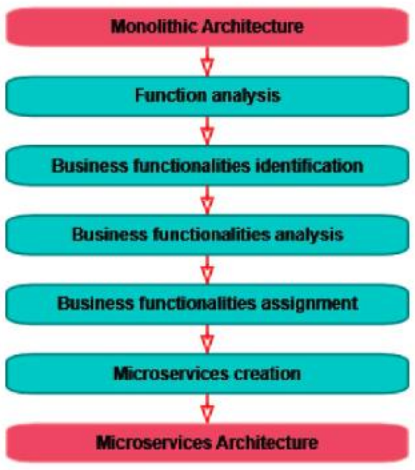
Monolithic Architecture
Function analysis
Business functionalities identification
Business functionalities analysis
Business functionalities assignment
Microservices creation
Microservices Architecture

Figure 3: Strategy for migrating from a monolithic application to a microservices application (De Lauretis, 2019).

The first step consists of a functional analysis of the monolithic application. This consists of evaluating the functions, which are blocks of code that accomplish a specific task, and modifying these functions as needed (e.g., combining functions or splitting a function into subfunctions). The second step consists of business functionalities identification, which is the phase in which the organization examines the current monolithic application and extracts the business functionalities from it. This stage can be challenging since the application is a monolith. The third step is business functionalities analysis, which involves analyzing the business functionalities identified in the previous step. Usually, this analysis involves obtaining data such as the rate of usage and other statistics. The next phase is business functionalities assignment, which involves assigning business functionalities identified in phase two to microservices. The scope and size of these microservices are informed by the statistics that were obtained in the previous phase. The last phase involves developing microservices so that they can be deployed and evaluated (De Lauretis, 2019).

The main limitation of these two studies is that they have little to no specific examples to support the modernization strategies that they present. The study conducted by Colanzi et al. (2021) surveyed different practitioners in the field on questions related to the eight-step roadmap to legacy system modernization that they presented, but they did not go into detail on what it would look like to go through each one of those steps. Further, the study conducted by De Lauretis (2019) describes each of the five phases of the legacy system modernization strategy but does not give practical examples for each phase. However, these two studies are easy to follow and therefore can provide starting points for organizations to begin their modernization efforts. Organizations should look to studies like these and recognize that they need to identify the business functionalities that exist within their monolithic applications, analyze them, and then build microservices that correspond to those business functionalities.

Part II: Differing Perceptions of Legacy Platforms and their Modernization

There are differing perceptions of legacy system platforms and their modernization between people in academia and people in industry. This includes perception of the definition of a legacy system platform, benefits of legacy systems, and reasons to modernize legacy systems. This is important because if the people in academia who are researching legacy system platform modernization techniques do not have the same perception as the practitioners, advancements in legacy system modernization efforts will be hindered. Therefore, in this section, I discuss these differing perceptions and highlight areas in which these perceptions must be reconciled.

Batlajery (2013) conducted 23 interviews with 26 participants who provided information on legacy systems and their modernization from different organizations in the Netherlands. These researchers asked the participants on the definition of a legacy system, perceived benefits of legacy systems, reasons for modernization, and challenges of modernization. These participants were chosen because they have experience with legacy systems and have experience with legacy modernization projects.

The following are the responses to the two research questions, “How do academia perceive legacy systems?” and “How do practitioners perceive legacy systems?,” respectively, asked by Batlajery (2013) based on the findings of the research:

> The research reveals that academic tends to define the legacy system from technical point of view. Business aspect is seen as an implication of the bad technology. On the other words, technology comes first to determine a system for being legacy and business aspect as a result of the bad technical aspect in the system. By this way of thinking, it makes the products from academic arena (e.g. journal, paper, etc.) have a lot of stress in technical side of the system, and only little business aspect in it.
>
> On the other side, professionals in industry always put business value in their definition of the legacy system. They only define legacy system only if their business gets interrupted by the existence of the current system. On the other words, business aspect always come first and determines the system for being legacy. Therefore, as long as their systems satisfy them, there will not be legacy system no matter the system is old and not good anymore from the technology point of view.

These findings show that the general disagreement lies in the fact that people in academia tend to focus on the technical aspects of system platforms when classifying them as legacy or not, whereas the practitioners in the industry tend to focus on their business needs and do not usually classify a system as a legacy system unless it no longer satisfies their business needs. Therefore, if a system is old and has limitations, people in academia will classify it as a legacy system, while practitioners in the field will not. Another differing perception between people in academia and people in the industry is their view of the role that the programming language of an application plays in classifying whether a system is a legacy system. Since practitioners usually only care that their business needs are met by the system, they generally do not consider programming languages to be a factor in deciding whether a system is a legacy system (Batlajery, 2013). This is supported by the findings of an independent market survey on COBOL, a programming language invented in the 1950s, that was conducted by OpenText, an enterprise information management company. OpenText found that there are more than 800 billion lines of COBOL code running on production systems that are in use daily. Additionally, they found that 52 percent of organizations expect for their COBOL applications to still be in use for at least the next decade and that more than 80 percent of participants predict that COBOL will still be in use when they retire (Micro Focus, 2022).

However, people in academia and practitioners both agree that some of the significant issues with legacy system platforms is that there is limited knowledge of these systems, they are difficult and costly to maintain, they are difficult to connect with other systems, and there are limited suppliers to support the hardware and software of these systems. People in academia and people in the industry also agree that organizations should avoid short term solutions for legacy system modernization and that data migration is a challenging aspect of legacy system modernization (Batlajery, 2013).

Based on this research study, it is clear that the differing perceptions of legacy system modernization between people in academia and practitioners in the field need to be reconciled. To do this, people in academia and people in the industry must exchange perspectives. People in the industry must educate people in academia on their business and its strategy and vision. With this knowledge, people in academia must educate practitioners on the cutting-edge technologies and platforms since they are concerned with these advancements through their research. This will benefit both parties and will enable organizations to stay current and agile by being able to keep up with rapidly changing business needs.

Conclusion

Throughout this paper, I argue that although challenging, costly, and time-consuming, organizations must modernize their legacy system platforms to keep up with the rapidly evolving needs of their business and their customers. To support my argument, I use studies focused on the migration from legacy monolithic applications to applications that utilize the microservices architecture. Additionally, I use a study that reveals the differing perception of legacy system platforms and their modernization between people in academia and people in the industry.

The literature by Velepucha & Flores (2023) shows that the biggest advantage of the microservices architecture over the more traditional monolithic architecture is that it consists of independent services that are independently developed, evaluated, and deployed, making it easier to maintain the application. Colanzi et al. (2021) and De Lauretis (2019) present an eight-step and five-step legacy system modernization strategy, respectively, for businesses to use. Both studies emphasize the importance of businesses taking the time to identify and analyze the business functionalities in their existing legacy applications so that they can develop microservices for all these functionalities. The study conducted by Colanzi et al. (2021) suggests that the most common driving forces for legacy system modernization for organizations are “easier maintenance and evolution”, “optimized scalability”, and “independent and automated deploy”. Lastly, the research performed by Batlajery (2013) shows that people in academia tend to focus on the technical aspects of systems when classifying them as legacy or not, while people in the industry tend to focus on the needs of their business and do not usually classify a system as a legacy system unless it no longer satisfies their business needs, even if the technology it uses is outdated. Therefore, based on my research, I conclude that it is important for organizations to take the time to modernize their legacy monolithic systems by replacing them with modern systems that utilize the microservices architecture because of this architecture’s benefits, and it is also important for people in academia and practitioners to reconcile their differing perceptions on legacy systems and their modernization for advancements in this area to proceed unhindered.

References

Batlajery, B. V. (2013). *Revisiting legacy systems and legacy modernization from the industrial perspective* (thesis). *Utrecht University Student Theses Repository*. Retrieved April 14, 2024, from https://studenttheses.uu.nl/handle/20.500.12932/14919.

Colanzi, T., Amaral, A., Assunção, W., Zavadski, A., Tanno, D., Garcia, A., & Lucena, C. (2021). Are we speaking the industry language? The practice and literature of modernizing legacy systems with microservices. *Proceedings of the 15th Brazilian Symposium on Software Components, Architectures, and Reuse*, 61–70. Presented at the Joinville, Brazil. doi:10.1145/3483899.3483904

G. Liu, B. Huang, Z. Liang, M. Qin, H. Zhou and Z. Li, "Microservices: architecture, container, and challenges," *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, Macau, China, 2020, pp. 629-635, doi: 10.1109/QRS-C51114.2020.00107.

L. De Lauretis, "From Monolithic Architecture to Microservices Architecture," *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, Berlin, Germany, 2019, pp. 93-96, doi: 10.1109/ISSREW.2019.00050.

Micro Focus. (2022, February 4). *COBOL market shown to be three times larger than previously estimated in the New Independent Survey*. https://www.microfocus.com/en-us/press-room/press-releases/2022/cobol-market-shown-to-be-three-times-larger-than-previously-estimated-in-new-independent-survey

O'Reilly's microservices adoption in 2020 report finds that 92% of organizations are experiencing success with microservices. O'Reilly Media - Technology and Business Training. (2024, April 14). https://www.oreilly.com/pub/pr/3307

76% of cios say it could become impossible to manage digital performance, as its complexity *soars*. Dynatrace news. (n.d.). https://www.dynatrace.com/news/press-release/76-cios-say-become-impossible-manage-digital-performance-complexity-soars/

T. C. Fanelli, S. C. Simons and S. Banerjee, "A Systematic Framework for Modernizing Legacy Application Systems," *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Osaka, Japan, 2016, pp. 678-682, doi: 10.1109/SANER.2016.40.

U.S. Government Accountability Office. (2023, February 15). *Information technology: Agencies need to develop and implement modernization plans for Critical Legacy Systems*. Information Technology: Agencies Need to Develop and Implement Modernization Plans for Critical Legacy Systems | U.S. GAO. https://www.gao.gov/products/gao-21-524t

V. Velepucha and P. Flores, "A Survey on Microservices Architecture: Principles, Patterns and Migration Challenges," in IEEE Access, vol. 11, pp. 88339-88358, 2023, doi: 10.1109/ACCESS.2023.3305687.