Using TOGAF for Migration of Monolith Systems to Microservices

Abstract
The transition from monolithic systems to microservices is one of the essential transformations in the developing technological environment. Monolithic architecture, though relatively easier to implement in formative stages, invariably comes across scalability, flexibility, and failure contingency issues as architectures evolve. With this approach of broken structures and small services, microservices provide great flexibility, redundancy, and scalability. However, moving away from monolithic systems involves certain risks, such as disruptions in operations and technical challenges. This paper examines how the structure of TOGAF, especially the ADM, helps to facilitate this migration. TOGAF contains a logical division based upon phases so the work can be carried out sequentially. It checks conformance to organizational goals, manages risks well, and supports staggered transformations. By incorporating Agile with TOGAF, the improvements of architectures are made on each increment to allow the flexibility required in migrating projects. Leveraging both solutions at scale reduces risks and technical debt and increases the delivery pace. The researchers also respond to organizational resistance, a way to deal with complexity and the trade-off between speed and quality. Techniques like hybrid migration approaches and constant reassessment are among the most effective. Using TOGAF's governance and risk management capabilities, an organization can attain the improvement requirement while sustaining business operation smoothness and adherence to the risk management regime. This paper explores the notion of strategic stability and control over a complex ecosystem of technological advancement vis-a-vis the need to create architectures for agility and organizational value creation presently complemented by TOGAF-led migration architectures. Mitigating concerns regarding value capture warrants attention to detail while illustrating a blueprint for firms to modernize while maintaining sustainable competitive advantage.

Keywords;
Microservices, Monolithic Systems, TOGAF, Migration, Agility, Scalability, Architecture Development Method (ADM), Governance.

1. Introduction

Current software architectures are shaping dynamically due to increased flexibility, speed, and cost-effectiveness demands in today's complex environment. Monolithic systems and microservices are two architectural styles commonly used to design the structure of software applications. It is important for organizations to know the differences between these stages and how transitioning from one stage to the other is vital for those who wish to avoid obsolescence. A monolithic system is a basic architectural model for designing an application in which all the elements are developed as one whole. Such a configuration implies that the user interface, business logic, and data management are entwined within the same application code. Although this approach makes a system's first setup and introduction much easier, it proves rather complicated as the system expands. To augment or scale up some parts, it is necessary to redevelop the whole application, which will not be effective and will waste time.

Microservices architecture breaks down an application into several modules of finer-grained, autonomously deployable functions. Each service has a certain level of functionality, and interaction is done through public interfaces only. These attributes make the system easily scalable, fault-tolerant, and flexible because change can be implemented at a micro level while keeping the rest of the system independent of the change. Microservices break the mold of traditional business structures well to conform to current needs and create new solutions. This migration is increasingly crucial for organizations that want to tackle the problems resulting from the monolithic structures of old systems. What many business people and developers do not realize is that as a business evolves or customer expectations change, monolithic systems do not adapt well. Such models are challenging to scale, maintain resilience, and leverage agility, which impedes creativity.

There are numerous advantages to developing an application with the use of microservices. They provide just-in-time scaling where businesses can scale individual services instead of the whole application. It is cheaper and more efficient in terms of resource use to undertake the approach in question. Errors are also contained; therefore, in the case of microservices, failure in one service does not necessarily mean failure of the whole application and increases system availability. The other strategic success factor is technology responsiveness. Organizations can adapt to twenty-first-century tools, methodologies, and approaches like DevOps. Microservices also accelerate delivery cycles, allowing the team to constantly deliver incremental changes to solve complex problems and adapt to changes in market demand. Microservices fit well in terms of globalization because they accommodate a distributed development model, allowing teams to be located in different geographical areas and work on different services simultaneously.

Failure to move away from monolithic systems implies missing the technological ship and getting stuck. Companies associated with legacy architectures suffer from losing positions to rivals adopting microservices for increased flexibility and creativity. Additionally, current software developers are more comfortable dealing with modern technologies; holding on to outdated systems leads to high talent turnover and higher recruiting and training costs. TOGAF is an enterprise architecture method widely recognized for its use in building and planning a company's IT strategies. It provides a framework for linking IT efforts to business objectives, therefore becoming useful when spearheading difficult transformation projects such as shifting from monolithic to microservices.

TOGAF is valuable for migration purposes because of the ADM Architecture Development Method. The ADM divides the migration process into distinct phases and sub-phases to guide and maintain the structure of the whole process and its components. TOGAF offers the procedure and tools for implementing the architectural vision, along with a detailed plan of the migration process. Regarding microservices migration, TOGAF has the advantage of providing an alignment strategy, handling risks, and a phased

approach. It confirms that migration is aligned with an organization's strategic vision in tackling operational, technological, and organizational problems. Through this framework, businesses can successfully transfer from legacy architecture to the new, desired one by avoiding hazards and effects that would otherwise cause more problems than necessary.

This article intends to identify how TOGAF can support transitioning from monoliths to microservices. It also explains the tendential risks organizations experience during such changes and how the TOGAF framework can minimize them. The article also continues to integrate TOGAF with Agile to develop an efficient and scalable migration plan that provides value without interrupting operations. At the end of this discussion, readers will understand how TOGAF can be useful for creating a transition from legacy frameworks to more appropriate sets for today's tough competitive environment for any organization.

2. Why Migration is Important

It is imperative for organizations to transition from monolithic systems to microservices so as to deal with the Technological landscape of the current generation. Over time, as businesses scale, the disadvantages of monolithic architectures, such as their inherent design for simple and fast implementation during system development, arise.

2.1. Challenges of Monolithic Systems

One of the biggest misconceptions people have about monolithic architectures is that one will be easier to develop initially because all of the functionality can sit in one application. While these systems may be beneficial, the constant increase in the complexity and size of the systems prove disadvantageous in the long run. One of the key issues that arise when using monolithic systems is scalability. Though the system has these features, one disadvantage of using the layered approach is the failure to scale individual components of the application, resulting in higher performance constraints on some parts than others. For instance, one application component may face high traffic. As a result, the entire infrastructure must be updated, which may cost many resources and be inefficient (Jones and Henderson, 2017).

Another obstacle is the nature of the so-called monolithic systems, that is, the combined and strict framework. Such systems are pigeonholed into providing support for a specific set of requirements or organizational conditions that have been identified internally or externally. The individual components are interconnected, and if one component requires an update, the whole application may have to be reinstalled, which brings about the aspect of time and interferences with organizational operations (May et al., 2024; Lee, 2019). Moreover, the problems associated with maintenance appear due to handling a large code base as a single unified unit. While developing the system, technical debt is incurred, and the overall software complexity increases; therefore, subsequent modifications become more challenging and expensive (Soni & Agrawal, 2018). Monolithic systems also have resilience problems. A breakdown of any link in the chain can crash the whole application, and this has the unfortunate effect of making it hard to determine where exactly the problem is without inevitably disrupting the business (Gill, 2018). This lack of fault tolerance means that the business is far more susceptible to system crashes, which can inevitably cause massive disruption and a loss of customer trust.

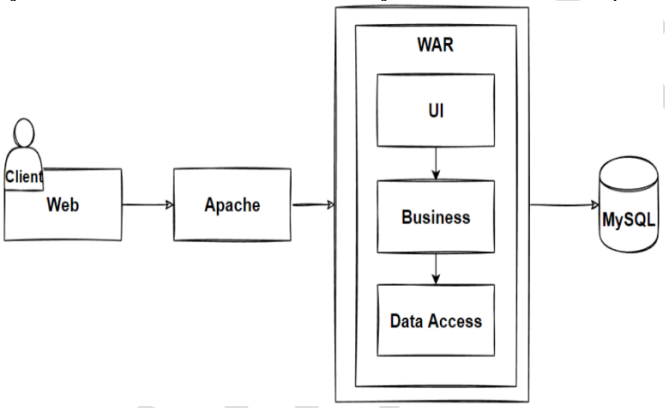

Figure 1: An Overview of Monolithic Architecture

2.2. Benefits of Microservices

Microservices provide a more flexible and elastic architecture for application development. There are countless benefits organizations aim to retain their agility and innovation in the transition to the microservices architecture.

- **Scalability and Flexibility:** Microservices are built to function as single parts, and businesses can graze only the individual service that demands additional resources than the other parts. Such a scale-out or scale-in approach is particularly viable in cloud computing since it assists in managing demand fluctuation and determines the application's capability to address the loads proficiently (Mishra & Agrawal, 2020). For instance, if an e-commerce platform engages in many customer transactions, then only the service that deals with order processing will be demanded to expand to accommodate the load compared to the entire platform.
- **Enhanced Resilience and Risk Management:** By nature, microservices are more easily recoverable because one service does not depend on another. If one service becomes impaired, it does not influence the rest of the system's services, reducing risks. This makes it possible to compartmentalize issues, remedy them, and resume activities without disruption (Hernandez & McDonald, 2020). The distributed reality of microservices also has advantages in more effective fault isolation and high availability, which are crucial in retaining customers' trust and constant business operations.

- **Technological Agility:** Microservices enable an organization to implement new technologies and frameworks independently of the rest of the system (Auer et al., 2021). Unlike traditional architectures in which a change in technology means that large proportions of code have to be rewritten throughout the application, microservices allow for gradual updates and the implementation of different technology solutions for various services. Such technological flexibility helps businesses remain competitive and implement advanced technologies such as artificial intelligence (AI), machine learning (ML), or the Internet of Things (IoT) within a single environment that would otherwise be challenging in monolithic architecture (Mishra & Agrawal, 2020).

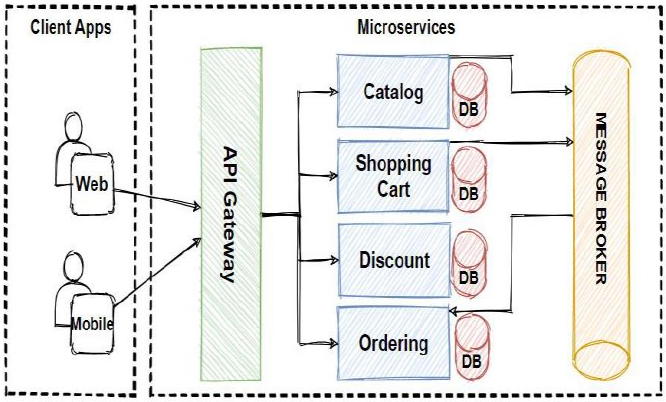

Figure 2: An Example of Microservices Architecture

- **Support for Innovation:** Microservices shorten the deployment cycle while enabling organizations to implement periodic enhancements and new additions. This ability to do rapid deployments helps cultivate a new venture mindset, where you can experiment, iterate more frequently, and adapt quickly to what customers want. In contrast to monolithic systems, where development and deployment are infrequent to allow the testing of novel approaches, microservices let teams implement enhancements more frequently and handle changes more effectively (Soni & Agrawal, 2018).
- **Globalization and Distributed Teams:** As organizations go global, microservices allow different teams to develop different services simultaneously. Every team can work on its microservice independently, addressing certain functionality without worrying about the complexity of the code implemented in a monolith. This approach is greatly useful for organizations operating in multiple regions because it allows the members to work synchronously while keeping the development processes going at all times (Hernandez & McDonald, 2020).

2.3. Costs of Delaying Migration

Although the actual transformation to adopting microservices has numerous valuable elements, the tendency to postpone the switch can lead to considerable detrimental impacts for organizations. This paper identifies the following as the loss that is achieved in the course of delayed migration. One of the immense costs is the opportunity cost. Organizations maintain monolithic architectures in their applications, making them lose on microservices' opportunities, such as time to deploy, scalability, and technological flexibility. This missed opportunity can disadvantage competitors who employ microservices since they deliver offerings faster and are capable of developing new solutions more quickly (Lee, 2019).

Apart from these lost opportunities, there are more reasons for companies to fail to migrate their applications. The pressure to match up with competitors will reach its peak. Any other business that does not adopt microservices for greater efficiency will be left behind as the other businesses adopt the new systems. This competitive disadvantage might deny companies the flexibility to counteract new trends or regulatory or customer demands (Jones & Henderson, 2017).

2.4. Costs of Maintaining Monolithic Systems

Sustaining monolithic systems proves expensive and resource-consuming because of their increased complexity. A key cost is the technical debt that keeps piling up due to the preference for one technology over the other. As organizations keep adding and repairing architectural scarps, the software code base becomes complex, expanding the cost of maintenance and shortening cycles (Rahman et al., 2022). Such technical debt slows down the assessment of new features or the inclusion of modern technology, which can hinder growth and development.

Further, supporting legacy systems poses certain challenges to the organization, specifically skill and talent demands that are often unique. While adopting new technologies, developers switch to different projects and may be unable to run and update monolithic systems for businesses. This resource intensiveness raises operational costs and thrusts the firm into a skills shortage that further hinders its versatility (Mishra & Agrawal, 2020).

2.5. Risks of Stagnation and Talent Attrition

Another distinct risk the organization bears is stagnation due to reliance on large integrated systems. Since organizations continue to employ traditional technologies within their marketing functions, they are not adaptable enough to changes in circumstances within the market. This lack of flexibility can slow innovation, making it much more difficult for a company to offer new products or services consumers may want (Soni & Agrawal, 2018). In addition, this erodes the market share, as failure to innovate and modernize results in competitors applying microservices to advance the customers' experiences (Nyati, 2018).

The loss of employees is also a major risk that significantly impacts organizations; this explains why talent attrition is considered a major risk. Younger generations, especially developers and engineers, may want to engage with emerging technologies. Corporations that remain loyal to conventional single-vendor systems will likely lose the best employees to companies with more attractive arrangements. This talent drain can mean the cost of hiring and training rises, compounding the problems that monolithic systems bring regarding resource usage (Hernandez & McDonald, 2020).

Monolithic Architecture	Microservices Architecture
Consists of a singe codebase with multiple modules within according to the business functionalities.	Consists of individual services with each service being responsible for exactly one functionality.
Do not need expert domain knowledge for development.	Risky to implement without domain expertise and container knowledge.
Easier deployment.	Relatively complex deployment.
Updating the system is a tedious process which would need the entire system to be redeployed.	Only the service which is updated needs to be redeployed.
Reusing the modules from one software into other software systems is difficult.	Microservices can be easily used in development of other software.

Figure 3: Some of the features of Monolithic and Microservices Architecture

3. How TOGAF Assists in Migration

It is quite a complex process to transfer from monolithic systems to microservices. Among all the frameworks that could help manage this transition, the Open Group Architecture Framework (TOGAF) is one of the most useful tools. TOGAF is a good tool for managing transformations in enterprise architecture, as it provides a framework for this.

3.1. Overview of TOGAF and its Architecture Development Method (ADM)

TOGAF is an inclusive frame of reference in the enterprise architecture (EA) sphere, including directives, procedures, and approaches to designing, developing, executing, and managing IT environments. Many organizations use it since it provides efficient and scalable designs for the Organization's goals. TOGAF aims to ensure that IT systems and business processes create value in the least expensive and risky manner.

TOGAF's strength lies in the ADM – Architecture Development Method – providing an approach for managing enterprise architecture. The ADM comprises eight tasks, each focusing on several architecture development sub-processes (Rinaldi et al., 2024). These phases only act in harmony to help organizations realize an integrated and balanced system that supports set business objectives and adapts to constant alterations in business processing. In the migration process from the monolithic structure to the microservices one, TOGAF provides Organizations with detailed step-by-step guidance on efficiently working through the migration process while staying aligned with business goals, meeting technical requirements, and handling all possible operational issues.

3.2. Mapping TOGAF Phases to Migration Activities

It was revealed that each phase of the ADM offers direction and a systematic approach to migration. Through the use of TOGAF, there is a way of ensuring that the transition from the monolith architecture to the microservices architecture is structured more guided and orderly.

Phase A: Setting Architecture Vision and Aligning Goals

In the ADM, Phase A, or Architecture Vision, the first step involves creating the vision for the enterprise architecture and aligning it to the organization's strategic objectives. In this phase, the organization has to set the goals for migration to microservices. One has to ensure that the vision is consistent with business imperatives and strategic goals, as the change to microservices should help accomplish them (Nookala, 2023). This phase assists the stakeholders in realizing what needs to be migrated, what is to be gained from the migration, and the organization's exposure to the migration.

When it comes to migration, the architecture vision helps define the future state of affairs and the objectives of the work moving forward. The design phase lays the groundwork for the overall migration process and guarantees that the change corresponds to the business model, technology framework, and necessities (Lankhorst et al., 2020).

Phase B: Business Architecture and Evolving Processes

In Phase B, enterprise architecture at TOGAF is working on the business part. During this phase, the ongoing business processes must be evaluated to determine the changes needed to support the microservices model. It mainly applies to understanding which business functionalities require realignment or enhancement to align with the new architecture. With this methodology, creating a business architecture that matches the microservices approach guarantees the alignment of technical changes with the operational and organizational requirements for properly implementing a microservice architecture (Söylemez et al., 2024).

Microservices can mean reconsidering organizational processes, activities, and communications. Much of this phase is chiefly strategic as it seeks to implement all these changes without derailing fundamental operations. The migration process can capitalize on this phase to introduce new business processes previously most integrated within the monolithic form.

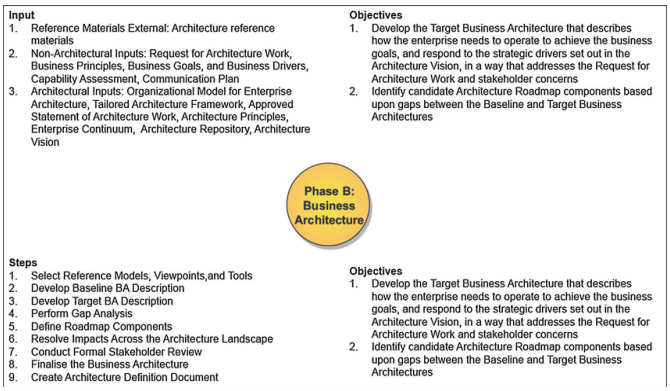

Figure 4: Overview of the Phase B in the TOGAF ADM

Phase C: Information Systems Architecture and Technological Gaps

Phase C concerns information systems architecture, which is the technological and data requirements that must be met to facilitate the transition to microservices. This phase assists in defining the technical requirements and enablers needed for a distributed microservices architecture. During this phase, two matters should be focused on: data integration and communication protocol and the interdependency of systems.

During the migration of organizations away from a monolithic design, issues of data migration, application of service integration patterns, and the interaction between legacy structures and newly created microservices can arise. The goal of Phase C is to devise a set of architectural decisions for the microservices architecture that would meet these challenges, as well as the fundamentals for scalability, dependability, and security (Niemann et al., 2019).

Phase D: Technology Architecture and Scalability Considerations

Phase D addresses technology architecture. This phase helps guarantee that the selected technology stack is appropriate for technology migration and growing a microservices architecture. Microservices are sometimes referred to as technologically demanding, and this is true because they need active containerization, orchestration tools, and cloud infrastructure (Saboor et al., 2022).

In phase D, it is necessary to review several technologies, consider how some may be integrated with existing systems, and determine whether these systems support the increasing needs of the organization's services. This phase is important to accommodate scalability and performance issues that emerge when organizations transition into developing microservices loosely coupled applications (Wang & Ye, 2020). In this phase, other crucial technologies such as Kubernetes, Dockers, and microservices frameworks are reviewed based on the required technical criteria.

Phase E: Developing a Roadmap for Phased Migration

In Phase E, the goal is to create a detailed architecture roadmap that provides a plan for gradually migrating to a new version. This roadmap helps prevent the organization or its infrastructure from feeling the heat when transitioning from monolithic systems to microservices (Baruah et al., 2024). It also offers the necessary indicators to determine which pieces of the monolithic system should be migrated first and to recognize interdependencies between several business capabilities.

A phased migration is possible, minimizes risks, and guarantees that the migration will not negatively impact business processes. As the activity plan will help develop a precise course of action, splitting the migration into segments is easy, ensuring progress in small, although consistent, steps (Vernadat, 2019).

Phase F: Transition Planning and Implementation Governance

Phase F includes transition planning and the plan for completing the migration. The final step of this phase is the establishment of governance procedures that will support the changeover process; they will monitor it and ensure that any complications encountered during the migration process are dealt with quickly. The kind of governance required during the migration process will help eliminate issues such as the expansion of the scope of migration, increase in budget costs, and overall time likely to be taken in completing the process. This phase defines how the resources will be aligned, how the stakeholders' expectations will be controlled, and how operational disruption will be minimized during the migration. It also defines parameters for evaluating the migration process's success, control measures, and risk indicators (Hughes et al., 2018).

Table 1: Mapping TOGAF Phases to Migration Activities

TOGAF Phase	Key Focus	Activities in Migration
Phase A	Architecture Vision	Define migration objectives, align goals with business strategy, and create a roadmap.
Phase B	Business Architecture	Analyze current processes, evolve business capabilities, and align operations with microservices.
Phase C	Information Systems Architecture	Address technological gaps, ensure data integration, and prepare a distributed architecture.
Phase D	Technology Architecture	Select and assess technology stacks, ensure scalability, and evaluate tools like containers and orchestration platforms.

TOGAF Phase	Key Focus	Activities in Migration
Phase E	Roadmap Development	Develop a phased migration plan, prioritize system components, and manage dependencies.
Phase F	Transition Planning	Implement governance mechanisms, coordinate resources, track progress, and address migration challenges.
Phase G	Risk Management	Identify risks, develop mitigation strategies, and engage stakeholders.
Phase H	Continuous Evolution	Monitor architecture, adapt to business needs, and ensure ongoing optimization.

Phase G: Risk Management and Stakeholder Engagement

The fourth phase of migration is risk management, where risk implications are sought at the early stage of migration, and ways of managing those risks are established (Simelton et al., 2021). This phase is critical for most migrations to maintain the stability and structural integrity of the migration process. The risk management provided in TOGAF allows an organization to foresee challenges, including technological dissimilarities, security threats, and shifts in the business agenda.

Besides risk management, Phase G of the project employs stakeholders' engagement. Stakeholder involvement during the change process makes everyone aware of their needs and concerns, so change will produce better results and be less disruptive (Vernadat, 2019).

Phase H: Ensuring Adaptability through Continuous Architecture Evolution

The last one is Phase H, which tackles the ongoing progress and improvement of the architecture. This way, it is possible to maintain the architecture of the microservices, the scale, and the ability to create new plans that meet the needs of the business as it operates after the migration phase. Closely related to this is the continuous monitoring of results, communication, and adaptation process, which are also key phase tasks.

It is necessary to designate a phase of continuous improvements so that migrating to microservices is not an isolated event but a slow and consistent process. When an architecture is continuously assessed and changed, it is more probable that the microservices architecture is optimal for the business and up to date with the technology (Wang & Ye, 2020).

3.3. Governance and Risk Management with TOGAF

Risk management and governance have been part of the TOGAF framework because the migration process must be controlled, aligned with business objectives, and work for different issues. Decision Right TOGAF acknowledges that an organization's decisions should be relevant to its strategic direction (Kornyshova & Deneckère, 2022). However, it should also be able to address new risks as they evolve or new requirements arise.

Regarding risk management in TOGAF, risks must be first identified at every phase in the migration process. Then, their potential impact must be evaluated before any form of mitigation is implemented. Any risk, whether a technological risk concerning a fresh software technology or an operational risk concerning stakeholder management, is done systematically through TOGAF. The key focus when planning for the migration is the aspect of governance to ensure that the change is successful and does not bring new problems that are likely to affect the organization's running (Hughes et al., 2018).

Figure 5: TOGAF Business Scenarios Guide

4. Integrating Migration, Agile, and TOGAF

With organizations integrating modern technologies, transitioning from monolith to microservices architecture has become a high business priority. This migration is not easy and needs to be executed systematically; otherwise, it will cause more problems than it solves. Adopting Agile has been harmonized with the TOGAF framework to considerably aid in transitioning from the traditional business (Schmitz & Wimmer, 2023). Through constant improvement, an integrated approach, utilization of data for decision-making, and the reduction of technological debt, the organizational dynamics of migration can be managed effectively, and the organization's technologies can be aligned with its strategic goals. This section discusses the advantages of integrating TOGAF and Agile methodologies, using tangible metrics to measure migration, and implementing agile governance in the TOGAF migration process.

4.1 Benefits of Combining Agile Methodologies with TOGAF

Incremental Development and Reducing Risks

Agile methodologies are well known for principles such as incremental development. Micro-reach is advantageous when decomposing large systems like monolith applications into microservices. Agile allows an organization to migrate in segments rather than as a large project—each segment consists of sprint tasks (Drutchas & Eppinger, 2022). This approach also minimizes the possible risks when launching many changes simultaneously since smaller changes are easier to perform, evaluate, and approve.

When using Agile, migration efforts can be disaggregated into smaller, deliverable parts, each of which has to correct the previous one incrementally. The nine phases and the overall structure of the TOGAF framework support this iterative process effectively. The ADM within TOGAF will see that any migration phase is in harmony with the organization's goals, aims, and long-term plans. With these methodologies integrated, it risks the new environment a smoother transition and less disruptive, as they can address the new challenges or changes in business priorities during the migration process (Kumar, 2019).

Cross-Functional Team Collaboration

Agile frameworks require cross-functional synergy among developers, business analysts, operations, and other teams. This kind of integration is critical in a migration project to provide cross-functional support to address the key goal of a migration initiative, for instance, user experience. This way, all the sentinel teams are involved in decision-making so that there are coordinated technical and business solutions to migration.

TOGAF enhances this collaboration by highlighting business and IT alignment through the concept of business architecture. It makes organizations think of business requirements, processes, and stakeholders during migration's first steps, which is a strong point of the framework. TOGAF makes it easier to connect technical and business aspects of the migration by keeping the technical focuses and business drivers aligned so that each migration phase includes reasonable and valuable business adjustments (Iacono, 2018).

Incorporating Stakeholder Feedback

Engaging stakeholders as feedback sources is one of agile's significant principles. This feedback loop plays a vital role in confirming that the migration is in the correct direction and, more so, checking the solution against the dynamic needs of the business. It assists teams in responding to the feedback gathered from the company's stakeholders based on the experiences of the end-users and the changing internal and external business environment.

Feedback can be naturally integrated into the migration process since TOGAF has a structured series of phases (Yudhistira & Fajar, 2024). The ADM has various cycles where users or stakeholders can interject, which is an effective way to ensure that the architecture continually adapts according to the firm's requirements. If feedback is beneficial during the migration process, organizations can improve their migration plans, and the resultant transformation from monolithic systems to microservices is achieved as intended (Thomson & Turley, 2020).

Data-Driven Decision-Making

In Agile, using data for decision-making is a central approach to maximizing the value of each process's steps. Real-time data and performance metrics can facilitate the migration process, including decisions about which components to migrate, which technologies to adopt for migration, and how to address performance or scalability issues (He & Buyya, 2023).

Combining TOGAF with Agile enables a vision of how the transition occurs, and statistics provides that backup. All stages of the ADM offer tangible goals, such as evaluating the technology architecture or recognizing the deficiency in the business architecture. These metrics enable teams to make resource-allocation decisions based on objective rather than perceived data, increasing the migration's impact, speed, or quality (Henderson & Venkatraman, 2018).

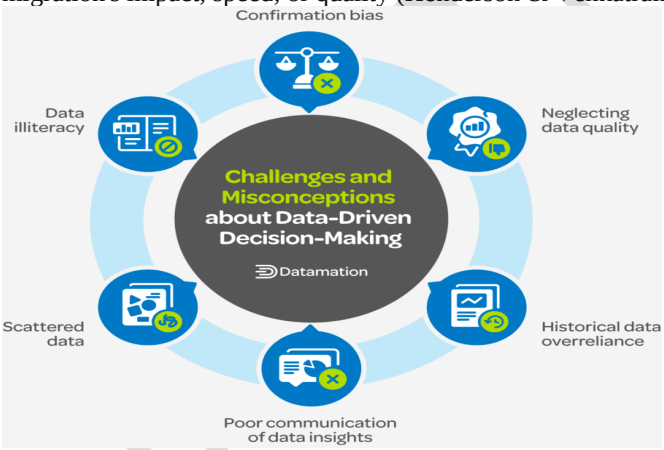

Figure 6: Data-Driven Decision-Making

Minimizing Technical Debt

Technical debt is the expense that comes when opting for the convenient short-term solution instead of the best long-term solution. Technical debt can be built regarding migration if the application transition from monolithic to microservices is not done properly (Solberg, 2022). However, with the help of Agile, the amount is reviewed and optimized regularly, and technical debts are paid as they are accumulated.

This focus on using a process-orientated approach to migration helps ensure that technical debt is not accumulated or is minimized as much as possible during the migration process. Because the framework avoids the build-up of technical debt by clearly defining the architecture vision and accepting that every migration phase must add some value, it can be considered innovative. The

continuous improvement methodology, including agile principles of integrating and testing the migrated systems before deploying, means that the migrated system must be sustainable, extensible, and consonant with future organizational objectives (Gartner, 2019).

4.2 Agile Metrics to Track Migration Progress and Agility

Monitoring a migration project's status is crucial because it has to be aligned with the right course to meet the set goals. Flexibility of the regular work cycle is based on a set of parameters that reflect the progress of the migration processes with the use of agile methodologies. All these help the different teams assess the performance of the particular strategy used and make necessary changes.

One of the work-rate measures most often employed is velocity, which demonstrates how much work can be done during a certain sprint. Velocity gives a conception of how long it will cost to finish the whole migration or where possible constraints may arise. Another important metric is cycle time, which represents the time needed to finish a particular task or a feature (Bebortta et al., 2023). Such a metric is most beneficial when used to locate those portions of the migration process where an increase in performance is possible. Lead time measures the time between requesting a feature and its implementation. The lead time definition allows organizations to understand whether they are adjusting quickly to emerging business requirements or customer needs. Together with the repeatedly outlined structured phases of TOGAF, these metrics give insights into the migration's efficiency and the flexibility of the involved teams (Visweswara, 2023; Hossain et al., 2020).

4.3 Agile Governance in Migration

The concept of Agile is then useful in maintaining governance to ensure that the migration process meets the organization's standards and strategic needs. Agile governance differs from conventional governance models that may slow down the development and decision-making process through heavy bureaucratic structures; it focuses on making quick decisions while monitoring critical factors of migration. Agile governance is another concept that refers to the capacity to make quick decisions based on the information received. Real-time data and feedback can be used to make correct decisions about how to handle migration by the agile teams, which microservices to focus on first, and what to do about some of the technical problems that may arise (Hedenäs Bennet & Jyborn, 2024). This makes the migration easier and faster and minimizes interruptions.

Agile governance is not only an adaptation of the principles of its native concept but also a system of flexible governance that must be combined with strict control. In many ways, these decision-making processes are efficient. However, considerations still need to happen within the governance structures to ensure the migration will meet the organization's goals, security, and compliance standards. TOGAF provides a framework for architecture to keep the migration on track while still engendering the necessary levels of flexibility. For example, through TOGAF's governance framework, risk management practices are implemented, and the organization can reverse unfavorable circumstances (Kuo et al., 2019). When integrated with TOGAF's architecture phases, agile governance provides organizations oversight of the migration process while embracing the speed required for change. It means that the migration process does not have to be slow, so it always complies with long-standing business goals.

5. Governance, Monitoring, and Ensuring Success

Assessment programs and controls implemented during migration to smooth the transition from a monolith approach to the microservices architecture are significant. These processes offer governance, management, and control mechanisms and ensure that migration objectives correspond to business needs and technological requirements. They also guarantee that the migration process stays on schedule and that possible dangers are well controlled.

5.1 Levels of Governance during Migration

Managing resources, business objectives, and threats in the transition toward microservices requires an efficient governance system. Governance frameworks typically operate at three distinct levels: strategic, tactical, and operational. Each level deals with different issues of migration management, providing a full-covering approach to transition.

- **Strategic Governance** determines if the migration should be done with the organization's main objectives, vision, and mission. Other architectural layers like the C-suite or architecture board is charged with the responsibility of overseeing the migration process to enable the achievement of long-term company goals such as scalability, flexibility and innovation (Hussain & Alsaadi, 2020). This level of governance defines the general direction of the migration and offers a vision for the migration as it aligns with the overreaching goals of the company.
- **Tactical Governance:** Tactical governance applies to project decisions and workings at a project level. This level is usually executed by project managers, senior architects, and technical activists who work towards performing specific key migration tasks. At this level, decisions on tools, technologies, and methodologies that would be used in the migration are ma, facilitating the migration successfully executioners to migration its suc. 2018). Tactical governance helps the project remain on the right track regarding scope, time, and costs.
- **Operational Governance:** At the operational level, the issue is centralized around a more detailed approach to implementing the migration plan (Poudel et al., 2023). This involves the management of the delivery, integration, and operation of microservices as these start to be deployed incrementally. Technical and business migration plans state that operational governance guarantees that the migration is run as planned and that all arising challenges are managed. The technical and business needs are met (Beneš et al., 2019). This also entails the constant assessment and guarantee of the functionality of the old and new systems during the migration process.

5.2 Monitoring Strategies

Progress should be monitored to ensure that progress is on track and that each risk factor that may hinder the migration process is detected on time. Several methods are employed to monitor the migration process with a view to ascertaining the quality of the migrated systems and, more importantly, determining whether migration brings in the anticipated business value.

- **Progress Monitoring with Key Metrics:** During migration, the most basic preoccupation often is to confirm that the migration process remains on schedule. It is also important to maintain metrics, including key performance indicators (KPIs) for measuring the level of the monolithic system to be migrated, the number of microservices, and features and deliveries that indicate how far the migration has been done (Sahlfeld et al., 2020). These metrics give a good idea of the state of the migration process and allow you to determine if there are any obstacles before the migration process has even started.

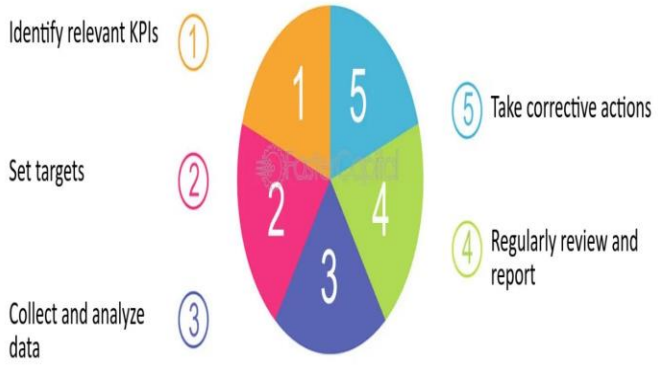

Figure 7: Monitoring And Evaluating Key Performance Indicators

- **Quality Assurance and Defect Monitoring:** While moving these systems, there needs to be a focus on the quality of these microservices to elicit positive results in the long run. The quality of migrated services is continuously checked with the help of automated testing, bug tracking, and code review to maintain the system's integrity (Monostori & Vancza, 2016; Gärtner et al., 2019). This step assists in detecting problems early and correcting them before they affect the entire system. Furthermore, if the quality of the code is measured during migration, the technical debt will not be published and can slow future applications and maintenance.
- **Business Value Realization Metrics:** The migration process that has been considered should be compared to the effects of migration on the business. Evaluations used in business value realization are based on operational efficiencies, customer satisfaction, and time to market for new enhancements. Unfortunately, these metrics assist in guaranteeing that the migration provides the projected business value and profitability, which also provides a competitive edge to the business organization (Aaker & Moorman, 2023; Kilic et al., 2018). To manage this, an organization will benefit from the continuous evaluation of the business value that results from the migration and be in a better position to create, develop, and implement strategies to meet the expected migration objectives.

5.3 Ensuring Compliance and Managing Technical Debt

The process should not overlook compliance with standards for the technological and business outcomes achieved. Management of technical debt is also paramount during the migration. When a migration is being conducted, technical debt may be quickly incurred when standards and proper processes are not adequately adhered to in the future.

- **Ensuring Compliance:** Requirements of maintaining the standard and qualification of the industry in migrating the process cannot be overemphasized. Business entities need to ensure that microservice architecture in their organizations follows the laws, security measures, and compliance with data privacy (Koutsou et al., 2019). Without continuous monitoring for compliance and performing periodic audits, it can be challenging to guarantee that the microservices adhere to the mentioned requirements and other pertinent laws and regulatory guidelines that may affect their legal and operational feasibility.
- **Managing Technical Debt:** Technological debt may emerge in organizations that are in the process of implementing the process of breaking down large monolithic systems and replacing them with multiple microservices since this process may be implemented haphazardly, with an emphasis put on speed, not on the quality of the resultant application. There is always a need to manage technical debt during the migration by performing code cleanup, imposing coding standards, and applying modern development paradigms (Buwalda et al., 2019). Continuous code reviewing, unit testing, and progressive adoption of microservices help keep the architecture lean, maintainable, and scaleable. This is because inadequate technical debt management may affect future development and be difficult to manage in the long run, owing to maintenance costs.

6. Practical Example: A TOGAF-Led Migration Strategy

6.1 A Case Study-Style Walkthrough

Setting a Vision for Migration

Before detailing the goals for microservices, some sound advice is that the vision must always be clear when migrating monolithic systems to an SOA environment. This vision runs throughout the process and helps maintain the necessary focus on business objectives and the chosen technical approaches. For example, a microservices architectural style should be targeted if an organization wants to advance its migration agenda to increase scalability and flexibility. The first step involves always considering the business

goals, such as enhancing customer satisfaction, cutting costs, or time to market, among others. Concerning TOGAF's Architecture Vision in phase A, the business can develop a migration map aligned with the company's short and long-term vision. This vision forms the basis of subsequent planning to support converting from a monolithic style to a microservices architecture while simultaneously satisfying the referred organizational strategic aims (Lange, 2017).

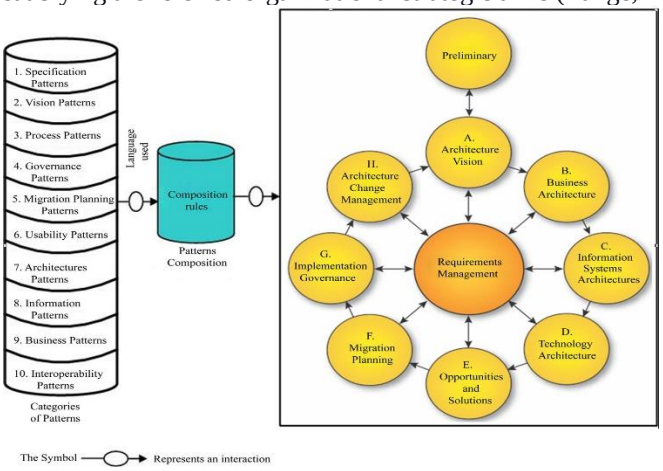

Figure 8: Pattern-Oriented Approach for Enterprise Architecture

Identifying Business Processes for Evolution

After setting the vision, the next step in TOGAF is to assess which business processes need to change to accommodate microservices architecture. Business System Analysis, as described in TOGAF's phase B, also involves a detailed business architecture that, when implemented, aids in comparing the current business processes with the existing monolithic structure and determining areas of weakness or inefficiency. For instance, a retail firm has to update its order management or inventory tracking mechanisms to adapt well to the microservices architecture. It reveals that the microservices approach makes these business units work independently, making the applications more scalable and tolerant to faults. During this phase, the organization has to involve cross-functional teams, such as business sponsors, enterprise architects, and developers, to ensure that the chosen processes will meet the business needs and be flexible to accommodate future changes (Sommerville, 2015).

Phasing Out a Monolith Incrementally

Several objectives must be met while implementing a migration, and studies have shown that one key element of such a strategy is that it must be split into stages. The traditional approach of doing a 'big bang' that requires a massive transfer of the entire system will come with many risks and instabilities. However, employing the framework provided by TOGAF's Phase F (Transition Planning), the organization should map out an incremental model (Visweswara, 2023). This means that functionalities in the monolith system will be split into microservices where each phase can be tested, and then the next component can be split. For example, a banking application may begin the implementation of microservices by shifting its customer management system to microservices but leaving its transaction processing systems in monolithic forms. Using microservices for certain business capabilities will minimize risk and allow the organization to continue operating normally while minimizing disruption to the development teams (Tornaz 2020).

Aligning Technology Stacks with Business Goals

When migrating, it is important to keep in tune with the business's goals and ensure that the technology stack reflects them. Moving from monoliths to microservices is not just a change of perspective regarding the service architecture and design. However, it can also be viewed as a chance to convert new tools and technologies that significantly enhance performance, scalabilities, and development productivity. In the TOGAF Phase C, Information Systems Architecture, the organization must consider existing technology deficits, needs, and emerging infrastructure for microservices. For instance, an enterprise can use an application that places workloads in cloud solutions through a container orchestration system, like Kubernetes, to handle microservices' deployment. In the same way, database choices, message queues, and all other supporting infrastructure components for each team will shape this architecture so that it is both scalable and fault-tolerant. The alignment of the proper technology stack with the business objectives ensures that the migration strengthens operational characteristics without bringing any problems or conflicts into the relationship between the IT and business departments (Winter, 2019).

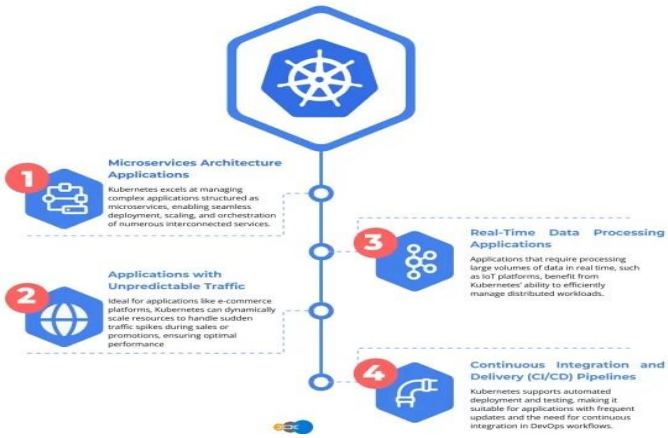

Figure 9: Kubernetes Deployment Strategy For an Organization

Continual Improvement and Changes

One of the major tenets of TOGAF is the evolution of the methodology, which is best seen in Phase H or Architecture Change Management. This means the migration process does not end after moving the systems to the microservices environment. What is more important is to ensure that the new architecture will accommodate future refinements and advancements. Integration and deployment cycles and agility enable microservices to be updated and improved in cycles. This approach is beneficial to developing a technical architecture since, with an iterative approach, changes can happen every time the business expands or develops. Furthermore, feedback should be gathered through stakeholders and users in all the migration phases so that what is produced meets business requirements and expectations and provides business value (Bennett, 2018). For example, after the relocation of the customer management system, one can collect feedback that facilitates subsequent improvements prior to the relocation of other business processes.

6.2 Highlight Key Lessons Learned

The following lessons can be drawn out of this case study as a guide for subsequent migrations employing the procedures embodied in TOGAF. First and foremost, a clear vision is essential in directing migration and ensuring the latter is in tune with organizational objectives in the long run. Without such a vision, the migration process will likely be characterized by low direction and fragmentation, thus exposing a system to more risk and cost. Moreover, dividing the migration into several, more manageable phases is essential. This makes it easier for organizations to check on each phase before going to the next one because all these phases are conducted staggered, which will not interrupt the normal running of the business. It was also highlighted that the choice of technology stacks to support business objectives remains critical. Cognitive misfit harms performance because new technologies must be well-aligned with a business operation's goals, demands, and requirements to deliver high performance. It is also because technology choices have implications for security and technical debt.

The other lesson is establishing feedback and improvement throughout the process. Organizations can adjust the system's architecture based on data analysis using feedback mechanisms within the migration process. In addition, flexibility should also be fostered within the culture of the organization. Integration of agile techniques with the TOGAF allows the organization to sidestep any barriers that can surface along the process without toppling the whole migration. Good and effective governance systems and structures at multiple levels and operation tiers help keep all the stakeholders informed, manage the risks that may come with the migration, and implement the migration systematically (Chuang, 2022). These lessons will avert future migration strategies merely delivering their technical factors but will generate more strategic business outcomes.

7. Challenges and Best Practices

7.1. Common Challenges in Migration

The transition from a monolithic system to microservices exposes various issues that may complicate the project's successful implementation. These challenges include the organizational culture that opposes change, the conveying of change, and finally, the problems associated with the complexity that results from distributed systems.

Resistance to Change

Another typical issue that organizations encounter while migrating is cultural resistance from personnel and other stakeholders. This resistance is due to the ease with which individuals stick to already established systems, the anxiety created by the possibility of interruption when migrating to a new system, and the control an implemented system exerts over different processes (Hidalgo and Ferreiro, 2020). Because monolithic systems have been developed over a long period for some employees, changing the ways of working or introducing new tools and technologies becomes complicated and can thus hamper the migration process. According to Wilson and Abdullah (2019), this resistance has to be tackled by organizations by enhancing a culture of openness and innovation. To these concerns, adequate training and resources to assist staff in transition could help ease these and thus adopt microservices.

Managing Complexity

This migration from monolithic systems to microservices is complex by design because it involves distributed structures. While monolith is characterized by a single code and a single database, microservices are an assembly of different parts, each of which is a small service to be managed and coordinated in different teams (Jadhav & Yadav, 2020). The added scalability is not only a technical

issue, which would be issues such as data transfer between the services and ensuring data integrity, but also an organizational problem for arranging the cross-functional teams with different goals and schedules. Such complexity can only be effectively managed by effective architecture planning and a migration program that is structured in phases, each of which is fully executed before the next is attempted (Adams et al., 2021). If the organization is not very strategic about the migration process, it may lead to performance complications, higher overhead costs, and scattered business logic.

Balancing Speed and Quality

Several important issues can be observed when discussing migration, including the problem of speed at the cost of quality and reliability. The nature of the migration process is that, sometimes, the migration to microservices must occur quickly, and this pressure comes from several business demands, which in turn result in the creation of shortcuts that affect the quality of the developed microservices (Zhao et al., 2020). While striving to complete them on time is not wrong, it has several drawbacks that manifest themselves as security flaws, suboptimal code, and setting up technical debts that make subsequent enhancements more challenging. This balance has to be tightly governed and quality-checked throughout the migration process to avoid the creation of one extreme at the expense of the other. To minimize these risks while making sure that the new architecture of microservices meets the set quality standards, the following measures should be employed (Milić & Makajić-Nikolić, 2022).

7.2. Best Practices for Success

Some of the best practices organizations use as they migrate to microservices eliminate the mentioned challenges. Such best practices include communication and synchronization, partial migration techniques, and frequent testing and validation.

Clear Communication and Alignment

The process of migrating should have a clear and coherent message. For the planning of the migration, various stakeholders on the technical aspect and other business aspects of the organization need to be on the same page concerning the goals and expectations of the migration (Zhang & Niu, 2021). Therefore, there is a clear and mutually understood understanding of why the migration occurs, the end vision, and how the success will be evaluated. Further, communication is also useful in pointing out other problems or challenges to the flow of work and effectively managing them to avoid time wastage or confusion of one form. Scheduled progress sessions, status check-ins, and reporting structures can align everyone and assist in addressing issues. As Bansal and Gupta (2019) outlined, the post of a migration champion or dedicated project manager can contribute greatly to efficient communication and decision-making processes.

Using Hybrid Migration Approaches

A blended migration strategy within which organizations can move away from monoliths incrementally while adopting microservices in a more significant step is a useful model. This approach helps avoid the shocks inherent to full-blown microservices adoption and offers a way to fix problems when they emerge. Rather than moving the entire system, which is usually full of complications and may require extensive time for the system to come up, organizations can migrate one business function or module at a time (Nyati, 2018). This helps teams integrate the microservices in a modular fashion, where they are designed so that it is easier to recognize if there is a problem. The hybrid approach also allows for changing the migration strategy depending on the feedback and issues that were met regarding migration.

Figure 10: Effective Hybrid Cloud Migration Strategy

Continuous Testing and Feedback Loops

Testing and feedback are important so that the migration process will continue effectively and the newly created microservices address the business and technical needs. Due to the continuous evolution of microservices, integration testing is crucial across the development life cycle to ensure continued functionality, performance, and security (Zhao et al., 2020). Integration testing and automated testing frameworks spotted during the early stages of testing can help deal with screw-ups early enough to avoid making the problem bigger and more costly. In addition, user feedback and stakeholders' feedback should be collected frequently to evaluate if the migration will bring the expected business benefits (Hidalgo & Ferreiro, 2020). Asking for feedback means that the migration strategy can be changed as the process progresses while staying true to the organizational objectives. This work-in-progress approach can also help control technical debt as it can fix problems when they arise and not pile up for a long time.

8. Conclusion

It is not a secret that the shift from monolithic structures to using microservices is a push that is growing more critical for companies that wish to stay relevant in a world of constant technological advancement. Monolithic systems are easier to design and implement than microservices but have limited scalability, are challenging to maintain, and are not robust. While these features seem natural in small organizations, they become serious limitations as they evolve. They must meet new customer needs and alter the offering to fit their requirements better. On the other hand, microservices offer a sound architecture that can accommodate these challenges and help organizations become more agile and adopt emerging technologies. Achieving success in this migration process is facilitated by frameworks consisting of TOGAF (The Open Group Architecture Framework). TOGAF has its Framework Administration Apparatus (ADM), an orderly procedure to assist organizations in dealing with complex change management, especially while transitioning from monolithic architecture to microservices architecture. The ADM makes the transition formal, enterprise-focused, and risk-aware simultaneously. TOGAF spans eight phases. Its main goal is not only to align architectural visions, revolve the business processes, and guide on how to integrate technology and manage change but also to provide a gradual transition plan, unlike applying big changes at once, leading to the occurrence of risk.

Using TOGAF, organizations can plan migration in phases that do not affect system functionality, hence allowing for efficiency in migration. In addition, integrating TOGAF with agile frameworks improves the migration process through continuous development methodologies, adaptability, and feedback integration. The principles of agility allow the migration project to be divided into possible sub-tasks so businesses can implement the changes steadily and concurrently with the delivery of other tasks. By taking this iterative approach, it is possible to mitigate the risks inherent in migration and check that each step in the migration process delivers the envisaged value and meets the organization's strategic goals. However, it is also important to note that though the benefits of microservices are quite apparent, the migration journey is not easy. Another challenge that organizations may encounter during this transition is cultural resistance, the challenge arising from the fact that managing distributed systems can be rather complicated, and the final challenge concerns the fine line between moving fast and delivering quality work. All of these must be brought down through effective and efficient interpersonal and group communication, strategic planning, enhanced operational and technology management, and testing with feedback. Companies should embrace trends such as the hybrid migration approach, which gradually makes the transition to microservices, and good governance, which deals with the migration process.

It is a technological and strategic change for a business to choose microservices architecture. Implementing the transition can be smooth, with lower risk, and achieve organizational goals if organizations use a well-developed framework such as TOGAF. The use of both TOGAF and Agile as frameworks to facilitate this process makes leveraging them for such a task a very sensible approach to achieving this complex change in a way that maintains organizational flexibility, robustness, and capability to respond to new technological realities adequately. This paper describes how organizations can successfully transition to a microservices architecture, allowing them to grow and develop in the constantly evolving business world by adopting precise planning and implementation and a continuous assessment of the project.

References;

1. Aaker, D. A., & Moorman, C. (2023). *Strategic market management*. John Wiley & Sons.
2. Adams, R., Gupta, S., & Patel, M. (2021). Managing the complexity of microservices migrations: A case study. *Journal of Software Engineering and Development, 8*(2), 103-118.
3. Auer, F., Lenarduzzi, V., Felderer, M., & Taibi, D. (2021). From monolithic systems to microservices: An assessment framework. *Information and Software Technology*, *137*, 106600.
4. Bansal, S., & Gupta, R. (2019). Communication strategies for enterprise architecture transformation. *Journal of Business Architecture*, *11*(4), 56-64.
5. Baruah, B., Vivekanandha, A., & Ramadoss, K. (2024). Introduction: Why Evolve from Infrastructure to Innovation with SAP on AWS?. In *Evolve from Infrastructure to Innovation with SAP on AWS: Strategize Beyond Infrastructure for Extending your SAP applications, Data Management, IoT & AI/ML integration and IT Operations using AWS Services* (pp. 1-72). Berkeley, CA: Apress.
6. Bebortta, S., Tripathy, S. S., Modibbo, U. M., & Ali, I. (2023). An optimal fog-cloud offloading framework for big data optimization in heterogeneous IoT networks. *Decision Analytics Journal*, *8*, 100295.
7. Beneš, M., Kováč, J., & Farkas, A. (2019). A comparative analysis of architecture governance models for microservices migrations. *Journal of Software Engineering Research and Development*, 7(1), 25-42.
8. Bennett, C. (2018). *Enterprise architecture as strategy: Achieving business agility through TOGAF and agile integration*. Wiley.
9. Buwalda, S., Wits, M., & Rijk, J. (2019). Mitigating technical debt during enterprise migrations: A case study. *International Journal of Software Engineering & Applications*, 10(2), 55-70.
10. Cheng, L., Liu, J., & Zhang, D. (2020). Business architecture in enterprise transformation: A systematic literature review. *Journal of Enterprise Architecture, 16*(2), 1-14.
11. Chuang, J. A. (2022). The International Organization for Migration and New Global Migration Governance. *Harv. Int'l LJ*, *63*, 401.
12. Drutchas, J., & Eppinger, S. (2022). Guidance on application of agile in combined hardware and software development projects. *Proceedings of the Design Society*, *2*, 151-160.
13. Gärtner, J., Müller, M., & Löffler, F. (2019). Quality assurance during the microservices migration process. *Journal of Computer Science and Technology*, 15(3), 115-130.

14. Gartner. (2019). *Tech trends and strategic planning for digital transformation*. Gartner.
15. Gill, A. (2018). *Developing A Real-Time Electronic Funds Transfer System for Credit Unions*. International Journal of Advanced Research in Engineering and Technology, 9(1), 162-184. Retrieved from https://iaeme.com/Home/issue/IJARET?Volume=9&Issue=1
16. He, T., & Buyya, R. (2023). A taxonomy of live migration management in cloud computing. *ACM Computing Surveys*, *56*(3), 1-33.
17. Hedenäs Bennet, O., & Jyborn, A. (2024). The Data Management of a Microservices Migration of Embedded Software-Strategies and Challenges in Migrating Embedded Software to Microservices Architecture.
18. Henderson, J. C., & Venkatraman, N. (2018). *Strategic alignment: A model for the integration of business and IT*. Journal of Business Research, 42(3), 397-406.
19. Hernandez, J., & McDonald, R. (2020). Microservices: Enhancing the Scalability and Flexibility of Software Systems. *Journal of Software Engineering Research and Development*, 4(2), 22-36.
20. Hidalgo, R., & Ferreiro, P. (2020). Overcoming resistance to change in microservices adoption. *International Journal of Business and Technology Management*, *25*(3), 230-242.
21. Hossain, M. M., Ahmed, M., & Khan, S. (2020). *Agile methodologies and their use in system architecture transitions*. International Journal of Software Engineering, 28(4), 227-245.
22. Hughes, T., Baumeister, H., & Morandi, A. (2018). Governance and risk management in enterprise architecture: A critical analysis. *Journal of Risk and Governance*, *9*(1), 45-57.
23. Hussain, M., & Alsaadi, F. (2020). Strategic governance in IT infrastructure transformation: A case of migration to microservices. *International Journal of Computer Applications*, 41(6), 22-37.
24. Iacono, A. (2018). *Business architecture frameworks in enterprise architecture.* Journal of Enterprise Architecture, 15(1), 16-26.
25. Jadhav, S., & Yadav, N. (2020). Managing the migration of monolithic systems to microservices: A strategic approach. *Journal of Software Engineering*, *13*(1), 77-93.
26. Jones, D., & Henderson, P. (2017). *Agile Enterprise Architecture: The Role of Microservices*. *Enterprise Architecture Journal*, 12(4), 91-106.
27. Kilic, N., Bayram, A., & Kantarcioglu, M. (2018). Measuring the business value of IT transformations: The role of microservices architecture. *Journal of Strategic Information Systems*, 27(2), 89-100.
28. Kornyshova, E., & Deneckère, R. (2022). A proposal of a situational approach for enterprise architecture frameworks: application to TOGAF. *Procedia Computer Science*, *207*, 3499-3506.
29. Koutsou, D., Djemame, K., & Amani, M. (2019). Compliance and governance frameworks in enterprise architecture transformations. *International Journal of Digital Enterprise Technology*, 7(3), 56-71.
30. Kumar, A. (2019). The convergence of predictive analytics in driving business intelligence and enhancing DevOps efficiency. *International Journal of Computational Engineering and Management*, 6(6), 118-142. Retrieved from https://ijcem.in/wp-content/uploads/THE-CONVERGENCE-OF-PREDICTIVE-ANALYTICS-IN-DRIVING-BUSINESS-INTELLIGENCE-AND-ENHANCING-DEVOPS-EFFICIENCY.pdf
31. Kuo, Y., Lin, Y., & Yao, H. (2019). *Agile project management in IT transformation*. Journal of Technology Management & Innovation, 14(2), 42-53.
32. Lange, T. (2017). *Architecting with TOGAF: A systematic approach to enterprise architecture*. Springer.
33. Lankhorst, M., Proper, H., & Vliet, H. (2020). A framework for enterprise architecture: The TOGAF approach. *Enterprise Architecture Journal*, *14*(2), 10-21.
34. Lee, S. (2019). *From Monolithic to Microservices: Overcoming the Challenges in Legacy Systems*. *International Journal of Software Engineering*, 28(3), 132-145.
35. Liu, Y., Xu, L., & Yu, Z. (2018). Tactical governance for microservices migration: A project management perspective. *International Journal of Project Management*, 36(4), 678-689.
36. May, M. C., Glatter, D., Arnold, D., Pfeffer, D., & Lanza, G. (2024). IIoT System Canvas—From architecture patterns towards an IIoT development framework. *Journal of Manufacturing Systems*, *72*, 437-459.
37. Milić, M., & Makajić-Nikolić, D. (2022). Development of a quality-based model for software architecture optimization: a case study of monolith and microservice architectures. *Symmetry*, *14*(9), 1824.
38. Mishra, P., & Agrawal, R. (2020). *Adopting Microservices: Benefits and Strategies for Organizational Transformation*. *Software Engineering Review*, 15(1), 45-61.
39. Niemann, F., Pasquier, L., & Maurer, P. (2019). Aligning business and IT through TOGAF and agile methodologies. *International Journal of Information Management*, *45*(1), 94-107.
40. Nookala, G. (2023). Microservices and Data Architecture: Aligning Scalability with Data Flow. *International Journal of Digital Innovation*, *4*(1).
41. Nyati, S. (2018). Revolutionizing LTL carrier operations: A comprehensive analysis of an algorithm-driven pickup and delivery dispatching solution. *International Journal of Science and Research (IJSR)*, *7*(2), 1659-1666. https://www.ijsr.net/getabstract.php?paperid=SR24203183637
42. Nyati, S. (2018). Transforming telematics in fleet management: Innovations in asset tracking, efficiency, and communication. *International Journal of Science and Research (IJSR)*, *7*(10), 1804-1810. https://www.ijsr.net/getabstract.php?paperid=SR24203184230

43. Poudel, L., Elagandula, S., Zhou, W., & Sha, Z. (2023). Decentralized and centralized planning for multi-robot additive manufacturing. *Journal of Mechanical Design*, *145*(1), 012003.
44. Rahman, T., Nwokeji, J., & Manjunath, T. V. (2022). Analysis of Current Trends in Software Aging: A Literature Survey. *Computer and Information Science*, *15*(4), 1-19.
45. Rinaldi, R., Khairul, K., Wijaya, R. F., Nasution, D., & Siahaan, A. P. U. (2024). Enterprise Architecture Using TOGAF ADM to Support Smart Campus at STAI Raudhatul Akmal. *Jurnal Info Sains: Informatika dan Sains*, *14*(04), 619-630.
46. Saboor, A., Hassan, M. F., Akbar, R., Shah, S. N. M., Hassan, F., Magsi, S. A., & Siddiqui, M. A. (2022). Containerized microservices orchestration and provisioning in cloud computing: A conceptual framework and future perspectives. *Applied Sciences*, *12*(12), 5793.
47. Sahlfeld, A., Fehn, K., & Buchegger, D. (2020). Monitoring migration progress in microservices architectures. *Software & Systems Modeling*, 19(2), 535-550.
48. Schmitz, A., & Wimmer, M. A. (2023). Framework for interoperable service architecture development. *Government Information Quarterly*, *40*(4), 101869.
49. Simelton, E., Duong, T. M., & Houzer, E. (2021). When the "strong arms" leave the farms—migration, gender roles and risk reduction in Vietnam. *Sustainability*, *13*(7), 4081.
50. Solberg, E. (2022). *The transition from monolithic architecture to microservice architecture: A case study of a large Scandinavian financial institution* (Master's thesis).
51. Sommerville, I. (2015). *Software engineering: A practitioner's approach* (10th ed.). McGraw-Hill Education.
52. Soni, P., & Agrawal, A. (2018). *Impact of Legacy Systems on Business Innovation and Growth*. *Journal of Technology Management*, 18(2), 89-102.
53. Söylemez, M., Tekinerdogan, B., & Tarhan, A. K. (2024). Microservice reference architecture design: A multi-case study. *Software: Practice and Experience*, *54*(1), 58-84.
54. Thomson, M., & Turley, K. (2020). *Stakeholder feedback and its impact on architecture migration*. Journal of Information Systems, 34(2), 132-149.
55. Tornaz, P. (2020). *Microservices migration strategies for enterprises*. Springer.
56. Vernadat, F. (2019). Modeling enterprise architecture for industrial systems. *Computers in Industry, 112*, 12-22.
57. Visweswara, S. (2023). *An agile enterprise architecture methodology for digital transformation* (Master's thesis, University of Twente).
58. Visweswara, S. (2023). *An agile enterprise architecture methodology for digital transformation* (Master's thesis, University of Twente).
59. Wang, Y., & Ye, J. (2020). Technology architecture for microservices migration: Best practices and challenges. *Journal of Software Engineering*, *32*(3), 159-172.
60. Wilson, D., & Abdullah, M. (2019). Organizational culture and migration to microservices: A survey of enterprises. *Journal of Enterprise Architecture*, *21*(5), 45-61.
61. Winter, A. (2019). *Microservices in practice: A migration guide for enterprises*. Springer.
62. Yudhistira, A., & Fajar, A. N. (2024). Integrating TOGAF and Big Data for Digital Transformation: Case Study on the Lending Industry. *Sinkron: jurnal dan penelitian teknik informatika*, *8*(2), 1215-1225.
63. Zhang, L., & Niu, Z. (2021). Effective communication strategies for cloud migration. *Cloud Computing and IT Management, 7*(2), 112-124.
64. Zhao, Y., Wang, T., & Liu, J. (2020). Balancing speed and quality in microservices migration. *Journal of Software Maintenance and Evolution, 32*(6), 589-603.