# Operational Backbone Work: Modernization Activities in the Migration of Monolith-Oriented IT Architectures

Egil Øvrelid[1], Olav Haaland Moseng[1], og Ludvig Oftedal Vinje[1]

[1] Department of Informatics, University of Oslo, Gaustadalléen 23B, 0373 Oslo

**Abstract.** To cope with new digital markets**,** incumbent financial organizations need to modernize their monolithic system portfolio to a more flexible and efficient form. This has proven to be quite challenging since the existing systems are tightly integrated with historical practices. We frame the portfolio as the operational backbone (OB) and ask *what are the key activities of OB work in monolithic systems migration, and what is the outcome of such a migration process?* We chose a case study approach to grasp a complex issue within a real-life context: a big financial institution embarking on a digital journey to modernize its core systems. Our contribution is a migration model that describes three key modernization activities attributed to OB Work, and the role of these activities in modernizing the IT portfolio from a fragmented to a more coherent and flexible OB

**Keywords: Operational Backbone Work, Modernization activities, Migration process, Monolithic Systems, IT architecture**

## 1 Introduction

All organizations that have operated for a while have an extensive portfolio of IT systems. Some of the core systems are often referred to as monoliths, i.e. three-layered systems that cover distinct requirements present in particular business areas [1]. These monolithic systems, developed and implemented at different times, are configured to respond to different business needs [2]. Examples are patient records developed before there was an expectation of patient access to own data, or financial systems where the transactions are between the bank and the system, not directly between the bank and the customer.

To cope with emerging requirements attributed to new interactions, most organizations have tried to integrate monoliths in a more or less systematic way, with varying degrees of success. The integrations have only solved some of the problems related to the lack of system flow but also created new dependencies and architectural debt. Rolland and Lyytinen (2021) developed the concept of *architectural debt* to denote architecting practices that threaten performance and robustness in IT architectures [3]. These dependencies may take the form of "meshwork", making it difficult to identify core functions during modernization.

Although challenging for incumbent organizations, the new dynamic landscape of fast immediate, and precise transactions, and competition from emergent entrants, requires drastic changes in the IT architecture [4]. Ross et al [5] describe several key challenges for organizations that move from tightly connected system portfolios to more flexible system landscapes. The IT architecture is central to this process. A central architectural component is the *Operational Backbone* (OB) defined as "the technology and business capabilities that ensure the efficiency, scalability, reliability, quality, and predictability of core operations" [2, p. 17]. The purpose of OB work is to achieve a more coherent system landscape, with robust and trustworthy digital services. Central to this process is the splitting of monoliths into smaller independent modules, which interact through modern API (Application Provider Interfaces) [6].

At the same time, through its role as a consolidated data bank that integrates services, and safely delivers them to actors working with innovations, OB is an object where the struggle between inertia and innovation is made evident. It is a venue for a demanding relationship between the monolithic system portfolios, and what is needed to achieve a more dynamic and flexible architecture [7]

Consequently, our key interest is to gain a better understanding of activities associated with OB work in a complex large-scale organization that is modernizing its IT portfolio. We understand this as a migration process; a gradual pragmatic replacement of monolithic systems in favor of a more modular and business-friendly architecture [8]. Our research question is: *What are the key activities of OB work in monolithic systems migration, and what is the outcome of such a migration process?*

We proceed by describing a theoretical discourse aligned with our interests before we describe our case and findings. Our contribution is a closer elucidation of the modernization activities linked to OB Work, and their contribution in the migration process.

## 2 From a monolithic to a modularized operational backbone

### 2.1 Operational Backbone Work

The Operational Backbone (OB) is a central part of the digital organization and includes "the technology and business capabilities that ensure the efficiency, scalability, reliability, quality, and predictability of core operations' [2, p. 17]. Incumbent organizations struggle with multiple non-consolidated IT systems [5], leading to a lack of trust in information due to potential incorrectness. Consequently, in these organizations, the OB often is an obstacle to innovation [5]. To reach Sebastian's' standard an extensive amount of work is required. Earlier literature has framed this work as architecting [9], [10], highlighting the multiple issues architects confront in streamlining IT architecture [9]. Other has framed it from a development perspective [11], focusing on the particular challenges addressed during planning and programming.

We align with these earlier framings but refer to these activities as OB work. The central goal for OB work is to modernize the IT portfolio – making it more responsive to instant changes in the business surroundings. Inspired by [12] we understand OB work as *modernisation activities associated with migrating the IT architecture from a tightly coupled monolithic structure to a more modular state.*

### 2.2 OB Work and the challenge of monolithic systems

Monolithic systems are generally described as systems where the various functional parts (data input and output, data processing, error handling, and user interface) are interwoven in a unified whole [13]. Monolithic systems are often historical systems that have supported the organization over time. Integration of monoliths to enable processes that are more efficient across expert groups in the organization, creates a very complex and entangled system landscape [14]. Thus, monolithic systems or clusters of integrated monolithic systems are hard to maintain because of tight coupling between their various components [15],[16]. Monolithic systems consist of a large code base, with limited maintenance of old code as the system continuously evolves. Often accumulated knowledge and experience is the usual way to understand the systems as they become increasingly complex. Since these systems contain crucial business logic, monolithic systems become the backbone of an organisation, constituting a key part of conducting business [15]

Monolithic systems also introduce an interesting dilemma, as replacing them is a costly undertaking, and despite their incumbent technical foundation, they are often very reliable when performing the tasks they were designed to perform [17]. Sneed, [18] infers that *re-engineering* of monolithic systems[1] has shown to be very challenging and associated with significant risk, and managers and developers are often reluctant to initiate such complex projects. Reluctance has persisted, and monolithic systems remain crucial parts of the systems in many large organisations, including most large banks today [21]. [17], [19] also highlights these risks, as failure in monolithic systems can have a catastrophic impact on an organisations ability to perform core activities.

Since monolithic systems are historically bounded to other contextual surroundings, they may reduce organisations abilities to execute their *digital strategies* and deliver innovative digital customer solutions and offerings [5]. As technology has advanced, it has enabled companies to increasingly use the capabilities of data, connectivity, and processing power to better understand their customers and deliver new value propositions [5], [22]. This has a significant impact on existing products and services, as using new technologies can enhance the customer experience, or make existing products or services obsolete [5]. Thus, we will next outline some core approaches to modernizing monolithic systems

## 2.3 OB Work and approaches to modernize monolithic systems

Bisbal et al. [19] present three main approaches to cope with monolithic systems: (1) redevelopment, (2) wrapping, and (3) migrating.

*Redevelopment*, or *Re-engineering*, is the most discussed technique in early monolithic systems research [18], [23] and describes the process of rewriting monolithic systems from scratch or replacing them with a Commercial off-the-shelf (COTS) system [19]. A redevelopment approach requires careful examination of the monolithic system and the organisation to understand business rules. This can result in an effective retirement of monolithic systems, replacing them with an architecture customised to the organisations need [12], [19]. Redevelopment is considered a "big-bang" strategy, and seen as risky in most cases. It is both time-consuming and expensive to replace system portfolios dominated by monolithic systems, with a completely new

[1] In the new DORA framework, legacy systems are defined as systems that is not longer supplied or maintained and has reached the end of its lifecycle. Both [17], [18] or [19] refer to legacy systems as systems that are still maintained. Since the systems referred to in the paper are still receiving maintenance, we use the term monolithic systems to comply with the DORA framework [20]

system portfolio [24]. Thus, redevelopment is in most cases considered an undesirable strategy [8], [12].

In contrast, *Wrapping*, also referred to as *Encapsulation*, is the simplest solution. This strategy implies creating new interfaces to enable access to existing data, systems, and applications [18], [19]. It is considered a "black-box" technique, as it concentrates the interfaces of the wrapped monolithic systems while hiding its internals. It is a fast, cost-effective, and simple way of exposing the services of monolithic systems, allowing new functionality on top of older systems. Wrapping also leverages the existing systems, which are often well-tested and trusted after many years in use (Almonaies et al., 2010). Wrapping strategies implies retaining the original systems. Reliability in terms of overloading and scaling remains an issue as the monolithic systems still process the requests. The cost of maintenance will also remain, or potentially increase as the wrapping itself also needs to be maintained [12], [19]. *Wrapping* is thus considered a short-term solution [19].

*Migration* strategies aim to move functionality from monolithic systems to a new platform, causing as little disruption in the operational environment as possible [19]. The process involves identifying functions in a monolithic system, extracting, and decoupling them into a new target system. Migration often utilise wrapping and redevelopment techniques as a part of the strategy, intending to move functionality away from the monolithic systems, over to a platform with improved design [12]. As migration aims to meet the requirements of the business already served by the target system, a good understanding of the monolithic systems' interactions, operations, and data is important before a migration effort can begin. The main drawback of migration techniques is that they are time-consuming, requiring many steps and large amounts of testing [12]. The *Strangler fig* pattern [8] is an example of a commonly used migration technique. It is a three-step process, combining elements from both wrapping- and redevelopment strategies.

## 2.4 Modularized IT architectures

*Platform architectures* and a *Service-oriented architecture (SOA)* have emerged as successful examples of modular architectures, emphasising modularisation of systems and services, with standardised mechanisms for communication and transparency [4], [25]. Modularisation enables the re-usability of secure and standardised components and facilitates interoperability, flexibility, and scalability [6], [26]

A modular design aims to decompose complex systems into many "black-boxed" services that communicate with each other, together forming a larger system [25], [26]. Services can range from delivering simple functions to complex processes, monolithic systems, or COTS systems, where their functionality can be viewed, accessed, and reused through standardised interfaces [27]. This "loose coupling" enables services to be called remotely without critically affecting the overall architecture, state, or behaviour of the system [28]. Microservices is an example of a decomposition technique. Microservices are autonomous services deployed independently, with a single and clearly defined purpose [1]. The modules need to be connected, allowing for interoperability. System- and data integration must enable visibility and accessibility, allowing them to be accessed and combined seamlessly [27]. In SOA, this is referred to as the *Service Bus*, enabling services to be accessed uniformly. Web services are the commonly accepted technology for enabling integration, using standardised protocols for service calls through API, messaging, and security tokens and routing. Combining and configuring these standards into reusable components ensures shared rules and policies across all services enabling standardisation of communication, interfaces, and security policies across the whole system [27]. Combining modularised services and standardised integrations through a service bus creates a modular architecture better served for fulfilling the requirements of an operational backbone [5]. We proceed with our Method before we describe our case and findings.

## 3 Case and method

### 3.1 Case: BankCorp

To improve our understanding of challenges attributed to OB work we investigated a case from a large financial institution. BankCorp is currently in the middle of a very important digital journey to modernize core systems. This entails important investments in building in-house competency to reach the strategic goal of transforming mainframe monolithic systems into modular architectures. BankCorp has chosen a gradual migration to cope with the fact that the bank has a huge amount of customers on several platforms, so they cannot shut down the business.

### 3.2 Data collection and data analysis

This is a qualitative case study based on interviews, documents, and conversations [29]. A case study encourages detailed study of a phenomenon, in a particular real-life context and may draw on multiple data collection sources [30]. We interviewed and had informal conversations with 12 architects from different departments in BankCorp: Integration and Flow, New Payment Platform (NPP), and Private Market (PM). Understanding the financial sector implies understanding both financial, organizational, and technical challenges, as well as regulatory requirements. The interviews lasted between 45 to 90 minutes.

We analyzed the data in four steps [31]. First, we created a timeline for the evolution of BankCorps IT portfolio, and the IT architecture. Based on the data, we then discussed the particular transition strategies used by BankCorp to adapt to the shifting contexts. We detected a significant shift around 2017 when BankCorp moved towards a modernization strategy. Central to this strategy was the effort to remove the constraints of the monolithic systems. We found three central activities: encapsulation, decoupling, and modularisation & integration to be decisive for this strategy. We also saw that they chose a migration process with gradual replacement of the older system functionality. The result of our analysis is summarized in Figure 1 - a process model. The model is inspired by the relation between context, mechanisms, and outcome [32].

## 4 Case background: BankCorps’ monolithic systems evolution

BankCorps' current systems have evolved from their early adoption of information technology in the late 1960s when the financial industry was at the forefront of IT adoption in Norway. The historical IT systems were primarily designed to support branch operations, with batch processing of transactions and accounting performed overnight. These systems were built to run on IBM mainframe computers using COBOL programming and DB2 database systems.

As online services became more prevalent and important, BankCorp prioritised modernising the systems. However, many of the critical core systems still ran on the mainframe, dating back to the 1970s, 1980s, and 1990s.

> The main underlying reason for our challenges lies in the disparity between the evolution of user interface-related technologies and the slow pace at which core banking systems are evolving, including those used by BankCorp. Some of these systems date back to the 1970s and 1980s, with some employees having worked on the same system throughout their careers until retirement.

In the 1990s, distributed systems such as Windows and Linux platforms, the Java programming language, and .NET emerged as modern alternatives to mainframe systems. The 1990s saw BankCorp make efforts to modernise systems to meet the demands of more modern and reliable services that required more frequent online access. However, these modernisation efforts did not address BankCorp’s reliance on the mainframe from the 60s for its core operations.

In the early 2000s, the emergence of online banking began with Sbanken leading the race. Before this, online banking was a very small segment. The initial development of BankCorp’s online banking system ran on the *IBM WebSphere* platform, a customised off-the-shelf software. The need for a mobile banking application became apparent in the mid-2010s, as a mobile-optimised website that was introduced in the early 2010s provided similar functionality to the online banking system.

In 2015, work began on a mobile application, which resulted in the development of a mobile-native online bank for Android and iPhone. Since the monolithic systems were not exposed as reusable APIs, it was necessary to start by integrating the core systems to enable the development of services.

## 5 Findings: Modernization activities as OB Work

### 5.1 Drivers for OB Work

**Business drivers: Strategic planning and maintaining customers.**
The integration of new services may necessitate a significant time investment for migrating underlying systems before the development of a new system can start. The head of tech strategy says that:

> We need to look a year or two ahead when deciding which areas to modernise because all new features added to our services might take a couple of years to solve, minimum. Some of them take more than five years to solve if the underlying systems require significant modernisation.

BankCorp recognises that maintaining a loyal customer base is the cornerstone for financial stability and growth, not technological excellence. The development of several shared services in the 2010s addressed this concern. An example of how the business needs drive modernisation activities lies in the mortgage process. BankCorp is currently facing new challenges in the form of competition from rivals that provide more efficient mortgage processes. In response, BankCorp must modernise its offerings to remain competitive and attractive to consumers, as explained by the Head of tech strategy, 2023;

> We are now at risk of losing business because *Bulder Bank* and our competitors have quite good mortgage processes. We look at the business drivers, where there is a threat to our position, and where we need to respond.

Since the majority of people have mortgages, the mortgage process is a core product within the retail business. To maintain its competitive advantage, BankCorp needs to continuously modernise and update its mortgage services.

**Technological drivers – Reduce Architectural debt and create robust services**
The mortgage process also concerns technological drivers as it has historically been challenging to automate. In 2017, significant efforts were made to automate the process through screen scraping, resulting in a fully automated mortgage process. However, this merely automated the existing system, utilising the same COBOL code and mainframe technology, as explained by the Head of tech strategy, 2023;

> It was the first fully automated mortgage process, but it was the same COBOL code running on the same mainframe. It did a good job, but we want to improve it again. So now, we're modernising the process from the ground up.

Adding additional layers to old systems makes it progressively harder to develop them further. Consequently, BankCorp is now looking to modernize the mortgage process, which will be an expensive initiative, but it will allow BankCorp to continue competing with new entrants. An efficient mortgage process requires stable payment systems. When the payment system relied on batch-based integrations of massive files, it became increasingly unstable.

## 5.2 Modernisation Activity 1: Encapsulation

Shortcomings in BankCorps' system portfolio became obvious during the development of the mobile bank in 2015. The old systems could not expose data and services efficiently. Any new application that needed to access the older system's services required system-to-system integration, which was a resource-intensive process. The challenges posed by these systems limited BankCorp's ability to innovate and develop new applications quickly. Modernization of the core systems to enable faster and more flexible interactions with modern applications was required.

Encapsulation was done to support this modernisation, by exposing the core systems to be consumed by applications based on modern protocols. By encapsulation, we refer to the process of bundling together data and processes to ensure controlled accessibility [8]. BankCorp enabled encapsulation by using a tool developed by IBM called *z/OS*

*Connect* which provided a standard way to expose COBOL assets using REST APIs (IBM, 2023). Encapsulation, thus, allows modern applications to easily access information that was previously only available through system-to-system integrations, as explained by the Engineering manager, 2023b;

> We had to encapsulate core functionality to support the modernisation of new applications, like mobile banking. These core systems could not be exposed and consumed by modern applications without being encapsulated concerning the functionality of what these core systems could expose in terms of data themselves.

Modern standardised protocols have been crucial in exposing the data and services from older systems, allowing work on new services across different channels. Making business logic from older systems accessible through APIs is essential for facilitating modernisation activities as it enables developers to use older functionality when developing new services.

## 5.3 Modernisation Activity 2: Decoupling

Encapsulation enabled the building of modern applications on top of older monolithic systems. This enabled the quick development of the mobile banking application. While the encapsulated systems continue to function adequately, the encapsulation strategy can be considered an "anti-pattern". It merely conceals the complexity of the back-end processes, as mentioned by the Head of tech strategy, 2023;

> It was always the intention to use encapsulation as a temporary solution. It was highly strategic when it was put in place and it’s still extremely important now, but over time, it should become less important because you create complexities by introducing an API gateway as a middleman.

This strategy persisted for about three years. Currently, the strategy for BankCorps' modernisation efforts involves extracting functionality from the older systems, ensuring that the modernised components remain entirely separate from these systems. This process is known as decoupling. However, decoupling of back-end systems is a time consuming endeavor that is expected to span for decades.

Customer data is crucial in the modernization of BankCorps IT Architecture. Historically customer data is fetched from several systems. Since the objective is to establish a distinct autonomous customer system that can be accessed directly, the Integration and Flow group is actively working on decoupling it from the older systems.

Payment services are also subject to significant migration efforts led by the New Payment Platform (NPP) project. A platform was created alongside the monolithic systems, and services were gradually migrated from the old systems to the new platform. However, certain aspects of the modernisation efforts at NPP follow a "big bang" approach, where a comprehensive overhaul is implemented. For instance, all European settlements for BankCorp are scheduled to occur over a specific weekend, constituting a significant change over a short time-frame. Modernisation efforts within NPP are deliberately avoiding encapsulation and building new systems on top of existing ones, in favor of new systems built from the ground-up, as explained by the Head of tech strategy, 2023;

> The modernisation strategy involves separating and extracting specific components from the monolithic systems instead of integrating them. This process, known as decoupling, focuses on the back end and is expected to span over a decade. The monolithic systems are broken down into smaller parts, such as payments, which can be delivered independently as part of the modernisation efforts.

BankCorp recognises the importance of using monolithic systems as a crucial part of the value chain for conducting payment transitions. In their long-term migration strategy, BankCorp systematically selects and decouples portions of the portfolio while continuing to leverage the functionality of the old systems. Consequently, the modernisation process unfolds gradually, with incremental updates being introduced over time.

## 5.4 Modernisation Activity 3: Modularisation and integration

Encapsulation and decoupling are two important activities to ensure business performance while migrating the IT architecture. To enable more permanent strategic agility, a modular IT architecture is required [22]. BankCorp has created several teams and tech families, that mainly develop microservices. Microservices are not only used to develop new services but also to enable access to information from monolithic systems. “We want to track …and to orchestrate the customer journeys from our side”, meaning that looser coupling facilitates faster development of new services emerging in dynamic markets.

Integration and Flow manage the migration process at BankCorp. They design, implement, and maintain systems that promote seamless data flow and connectivity between modular and independent technologies, with a portfolio of shared services. The development of new systems and the continuous improvement of existing systems would not be possible without a set of well-integrated shared services. An important task for Integration and Flow has been to standardise rules and guidelines for communication between services and applications at BankCorp. Those are Rest API’s, API gateways, Developer Portal, Event streaming, Authentication and authorization, and monitoring.

**REST APIs**
REST APIs provide an interface allowing one application or service to connect and access resources of another application or service. They are the most common style of implementing APIs, allowing for stateless communication of resources in many data formats, and functionality for reading, creating, updating, or deleting within resources through different requests. This allows for a large degree of flexibility when both implementing the REST APIs and using them, making them ideal for many different applications and communication between them.

**API gateways**
All externally exposed APIs have to go through an API Gateway. API Gateways serve as a proxy to the back-end logic of the APIs, routing the calls through the gateway, which handles security measures such as authorisation and authentication, validations

of API keys, load management, and logging and monitoring of the APIs. API gateways are a crucial technology for monolithic system encapsulation, as mentioned by Engineering manager, 2023a;

> The approach adopted involves the creation of an API layer that encapsulates the mainframe systems, thereby exposing them as REST APIs to various consuming systems, such as mobile applications. The objective behind developing these APIs is to ensure re-usability, avoiding the creation of APIs specific to a particular consumer.

**Developer portal**

As the amount API's grows, ensuring discoverability will become a significant challenge. BankCorp's solution has been to create a *Developer Portal*, a web page with a searchable overview of all exposed APIs at BankCorp. The various groups publish their APIs and documentation and communicate updates throughout the life cycle. Publicly available APIs are also available in the Developer Portal, such as Payment Services Directive 2 (PSD2) APIs, allowing anyone to view and request access tokens through the portal.

**Event Streaming**

At BankCorp, they use *Apache Kafka* to serve as the event hub for exposing and subscribing to data streams of events to and from systems and data sources. Similarly to the Developer Portal for APIs, BankCorp is working on creating a Data Product Catalog, quite similar to the developer portal displaying data sources available for subscription. Event streaming is also central in standardising integrations, as explained by the Engineering manager, 2023b;

> Integration mechanisms like event streaming standardise how we conduct integrations; otherwise, a proliferation of integration mechanisms may occur. Therefore, it is primarily important that these services are exposed as REST APIs. If data is to be provided, we prefer it to be offered as data streams through Kafka and events.

Events represent changes, meaning that changes in databases can be streamed through the logs, and be subscribed to by consuming applications and services. This allows for real-time data streaming and data capture, enabling a continuous flow of data containing the right information at all times throughout the entire architecture [33].

**Shared authentication and authorisation services**

Having a shared service for authentication and authorisation is important to ensure access control across multiple systems. This is managed by a shared Identity and access management (IAM) solution. Authentication and authorisation are two distinct processes that are necessary for communication between services. BankCorp operates on an internal zero-trust policy, which means you have to be approved to be able to access data. The internal procedure for using an API within BankCorp is carried out through the developer portal as explained by the Engineering manager, 2023b;

> The IAM solution manages authentication and authorisation for all services.

> Onboarding for API consumption is an internal process at BankCorp that is facilitated through the developer portal. Services are provided with API keys for connecting to the API, and approval is granted by the providing service.

## 5.5 Outcome of modernization activities: A more robust OB

These three key activities in operational backbone work have played a central role in BankCorp's current ability to develop new consumer-oriented services and applications, like the development of the mobile banking application or the automated mortgage process. Private Market (PM) is responsible for developing several services on an application layer, which enable access to essential functionalities across various applications. Examples of functionality are *updated customer information and onboarding processes.* This functionality is enabled through REST APIs or a more event-driven design:

> We (Integration and Flow) are currently working on creating a customer-information master for BankCorp. Instead of each system individually querying the master for updated customer information, an event-driven design through Kafka is being implemented.

When an update occurs for a customer, the customer information master pushes the update to all services that require the updated customer information. In addition to application development, PM has the responsibility of developing a set of front-end services that are organized into two distinct areas specific to retail operations. These front-end services consist of reusable components that can be utilised across multiple services in BankCorp. One component is retail *onboarding* for the online banking's front-end systems. Retail onboarding refers to the process of welcoming and establishing new customers within the retail sector. On-boarding typically includes activities such as gathering customer information, verifying identities, setting up accounts, explaining product offerings, and facilitating the initial interaction between the customer and the banking service.

## 6 Discussion

In this section, we return to our research question – *what are the key activities of OB work in monolithic systems migration, and what is the outcome of such a migration process?* Our main goal is to identify and describe modernization activities embedded in OB work. To provide a clearer understanding of the work required, we reference literature on monolith systems modernisation [12], [19] and modular architecture [6], [26], [27].

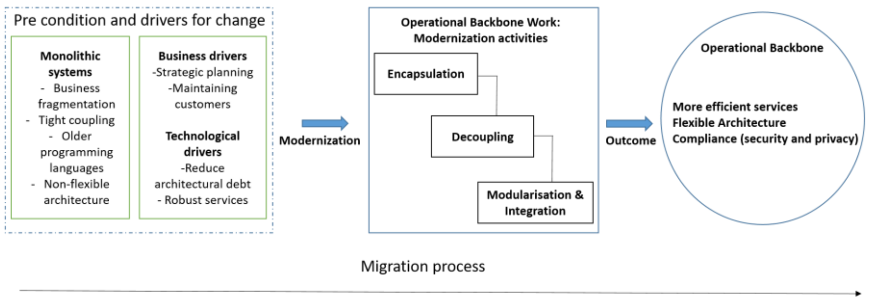

*Figure 1 OB Work: A model for describing key modernization activities in the migration process*

Our contribution is a model for describing the migration process (Figure 1). The Migration is important since the bank cannot close. Thus, BankCorps migration process is designed to transfer the functionality of monolithic systems with minimal disruptions to the operational environment. Creating a good modular architecture also requires thoughtful decomposition of the monolithic systems to reduce complexity, creating separate components with higher visibility, and re-usability [1]. At the same time, it is necessary to create a more flexible architecture that both facilitates more control, and more efficient development of new services [2], [5]. It needs to be more transparent, and easier to use for innovation [4], [7]

Inspired by the context, mechanisms outcome [32], Figure 1 describes three sequences of the migration process: *Precondition and drivers, OB Work, and outcome*. *Preconditions and drivers* describe the initial condition of the IT architecture dominated by tightly coupled monolithic systems. The drivers for change are both financial (strategic planning and maintaining customers) and technical (reducing architectural debt and creating more robust services). With the emergence of online services and faster change, a more flexible architecture is needed.

By *operational backbone work*, we aim to contribute with an overall conceptualization of the various activities attributed to creating a streamlined architectural foundation for Micro in an incumbent organization [2], [5], [22]. OB work consists of several *modernization activities*, which we see as discrete activities engaged in transforming the architecture from a monolith to a more flexible architecture [18], [19], [23]. In our case, we identified three central modernization activities: *encapsulation, decoupling, and modularisation & integration.*

We refer to *encapsulation* as an activity that describes the appropriation of monolithic systems to enable fast and secure data harnessing. This appropriation often entails using interfaces to establish a path for secure communication [8], [19]. Encapsulation is valuable, but not enough in the long run, since the monolithic systems are still in charge. Modernization may therefore also include decoupling. We refer to decoupling as an activity that describes the decomposition of tightly coupled systems, and the loosening up of inscribed and unflexible relations. This is often followed by *modularisation & integration* defined as the process of modularising monolithic systems in the transition process and integrating the modules through modern interfaces. According to core IS literature, modularity is about splitting the design and production of technologies into independent subparts [6]. The outcome of such a process may be digital agility [22] i.e. foundations for efficient development. Additionally, the process also facilitates securing mechanisms [34]. For BankCorp this is important since compliance with regulations such as GDPR are essential. The *outcome*, of such a process is, we find, more efficient and trustful services, a more flexible architecture with less architectural debt, and improvements toward compliance in line with security and privacy issues.

The study has limitations, due to its relatively few interviews. Even though the generalizability of the findings may be debatable, research of this sort may carry important insight that is similar in other sectors.

## 7 References

[1] F. Auer, V. Lenarduzzi, M. Felderer, and D. Taibi, "From monolithic systems to Microservices: An assessment framework," *Information and Software Technology*, vol. 137, p. 106600, Sep. 2021, doi: 10.1016/j.infsof.2021.106600.

[2] I. Sebastian, J. Ross, C. Beath, M. Mocker, K. Moloney, and N. Fonstad, "How Big Old Companies Navigate Digital Transformation," *MIS Quarterly Executive*, vol. 16, no. 3, Aug. 2017.

[3] K. H. Rolland and K. Lyytinen, "Exploring the Tensions Between Management of Architectural Debt and Digital Innovation: The Case of a Financial Organization," Proceedings of the 54th HICSS 2021.

[4] Y. Yoo, O. Henfridsson, and K. Lyytinen, "Research Commentary—The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research," *Information Systems Research*, vol. 21, no. 4, pp. 724–735, 2010

[5] J. W. Ross, C. M. Beath, and M. Mocker, *Designed for Digital: How to Architect Your Business for Sustained Success*. MIT Press, 2019.

[6] C. Y. Baldwin and K. B. Clark, *Design Rules: The power of modularity*. MIT press, 2000.

[7] B. Bygstad and E. Øvrelid, "Architectural alignment of process innovation and digital infrastructure in a high-tech hospital," *European Journal of Information Systems*, vol. 29, no. 3, pp. 220–237, May 2020

[8] S. Newman, *Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith*. O'Reilly Media, Inc., 2019.

[9] A. Martini and J. Bosch, "A Multiple Case Study of Continuous Architecting in Large Agile Companies: Current Gaps and the CAFFEA Framework," 13th Working IEEE/IFIP Conference on Software Architecture (WICSA), 2016

[10] A. K. Ajer and E. Øvrelid, "ARCHITECTURE WORK: MODES OF ARCHITECTING IN LARGE-SCALE INFRASTRUCTURES," *ECIS 2023*

[11] R. C. Martin, *Agile Software Development, Principles, Patterns, and Practices*. Prentice Hall, 2002.

[12] A. A. Almonaies, J. R. Cordy, and T. R. Dean, "Legacy System Evolution towards Service-Oriented Architecture," in *International workshop on SOA migration and evolution*, 2010.

[13] R. Stephens, *Beginning Software Engineering*. 1st ed. Wrox Press Ltd., 2015.

[14] S. Sarkar, S. Ramachandran, G. S. Kumar, M. K. Iyengar, K. Rangarajan, and S. Sivagnanam, “Modularization of a Large-Scale Business Application: A Case Study,” *IEEE Software*, vol. 26, no. 2, pp. 28–35, Mar. 2009

[15] D. Taibi and K. Systä, *From monolithic systems to microservices : A decomposition framework based on process mining*. SCITEPRESS, 2019.

[16] E. Øvrelid, “Exploring adaptive mirroring in healthcare IT architectures,” *Health Systems*, pp. 1–22, 2023.

[17] K. Bennett, “Legacy systems: coping with success,” *IEEE Software*, vol. 12, no. 1, pp. 19–23, Jan. 1995, doi: 10.1109/52.363157.

[18] H. M. Sneed, “Planning the reengineering of legacy systems,” *IEEE Software*, vol. 12, no. 1, pp. 24–34, Jan. 1995, doi: 10.1109/52.363168.

[19] J. Bisbal, D. Lawless, B. Wu, and J. Grimson, “Legacy information systems: issues and directions,” *IEEE Software*, vol. 16, no. 5, pp. 103–111, Sep. 1999

[20] Springflod, “DORA - Digital Operational Resilience Act,” 2023. https://www.dora-info.eu/article-3/ (accessed Jul. 05, 2023).

[21] Advanced, “2022 Mainframe Modernization Business Barometer Report,” 2022.

[22] V. Grover, “Digital agility: responding to digital opportunities,” *European Journal of Information Systems*, vol. 0, no. 0, pp. 1–7, Jul. 2022,

[23] W. S. Adolph, “Cash cow in the tar pit: reengineering a legacy system,” *IEEE Software*, vol. 13, no. 3, pp. 41–47, May 1996, doi: 10.1109/52.493019.

[24] Comella-Dorda, Wallnau, Seacord, and Robert, “A survey of black-box modernization approaches for information systems,” in *Proceedings 2000 International Conference on Software Maintenance*, Oct. 2000, pp. 173–183

[25] A. Tiwana, *Platform Ecosystems: Aligning Architecture, Governance, and Strategy*. Newnes, 2013.

[26] M. Ren and K. Lyytinen, “Building Enterprise Architecture Agility and Sustenance with SOA,” *CAIS, 2008* doi: 10.17705/1CAIS.02204.

[27] T. Erl, *Service-oriented architecture: concepts, technology, and design*. Upper Saddle River, NJ: Prentice Hall Professional Technical Reference, 2005.

[28] D. L. Parnas, “On the Criteria To Be Used in Decomposing Systems into Modules,” vol. 15, no. 12, p. 6, 1972.

[29] D. Silverman, “Interpreting Qualitative Data,” pp. 1–568, 2019.

[30] B. J. Oates, M. Griffiths, and R. McLean, *Researching Information Systems and Computing*. SAGE, 2022.

[31] B. Bygstad, B. E. Munkvold, and O. Volkoff, “Identifying generative mechanisms through affordances: a framework for critical realist data analysis,” *J Inf Technol*, vol. 31, no. 1, pp. 83–96, Mar. 2016, doi: 10.1057/jit.2015.13.

[32] E. Øvrelid and B. Bygstad, “The role of discourse in transforming digital infrastructures,” *Journal of Information Technology*, vol. Vol. 34, no. 3, pp. 221–242, 2019.

[33] “Apache Kafka,” *Apache Kafka*. https://kafka.apache.org/intro (acc Jul 2023)

[34] A. Ghazawneh and O. Henfridsson, “Balancing platform control and external contribution in third-party development: the boundary resources model,” *Information Systems Journal*, vol. 23, no. 2, pp. 173–192, 2013