Tampere University

Sulav Rayamajhi

EVALUATION OF STRATEGIES OF MIGRATING LEGACY SYSTEMS

A systematic literature review

Faculty of Information Technology and Communication Sciences
Master of Science Thesis
Examiners: Kari Systä
Terhi Kilamo
April 2024

ABSTRACT

Sulav Rayamajhi: Evaluation of strategies of migrating legacy systems - A systematic literature review
Master of Science Thesis
Tampere University
Master's Degree Programme in Computing Sciences – Software, Web & Cloud
April 2024

Software systems have become an integral part of any organization, and they operate in a highly dynamic environment driven by a rapid change in technology. With the passage of time, these systems start to age and slowly start to become legacy. Such legacy systems need to be operational as they support the core functionalities of any organization. However, these systems struggle to operate if they are not upgraded resulting major impacts on various organizational activities. Therefore, an organization should consider migrating or renewing them to a modern technology using various strategies or techniques.

The purpose of this thesis is to explore and evaluate various migration strategies that can help the organization to select the most suitable one. These strategies are compared based on their strengths and weaknesses facilitating an organization to choose the best one that suits their requirement. In this research, we used systematic literature review method to extract the data from the studies which concerned the migration of the legacy systems. The collected data were further processed to pick out various migration strategies along with their significance to fulfil the objective of this thesis.

Our research identified various aspects that needs to be considered while migrating a legacy system. Likewise, it gathers various migration strategies and groups them based on the target architectures. It also collects other strategies that can be used irrespective to the target architecture. Furthermore, the comparison is done on basis of strengths and weaknesses to provide an in-depth analysis. However, migrating a legacy system is not so straightforward and is very critical. There are several challenges that relates to each strategy which need to be considered and the strengths needs to be utilized to make a software migration successful.

Keywords: Legacy system, Migration, Modernization, Cloud architecture, Service-oriented Architecture (SOA), Microservices, Strategy, Evaluation

PREFACE

This thesis was done to finalize my Master's Degree in Computing Sciences at Tampere University. The main objective of this thesis is to propose a solution to a problem that is within the case company. This thesis can serve as a reference or guide which can be utilized during migrating the existing legacy systems to a modern architecture.

The thesis was completed with the help of various person. I would like to thank my supervisor Hannu and my people manager Paivi from the case company for their support and guidance. In addition, big thanks to my university supervisor Professor Kari Systä and my examiner Terhi Kilamo from Tampere University who provided constructive feedback, advice, and assistance throughout my thesis process.

Tampere, 24.05.2024

Sulav Rayamajhi

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

ADM	Architecture Driven Modernization
AQ	Assessment Question
CEC	Cloud Environment Constraints
CI/CD	Continuous Integration/Continuous Deployment
CIMO	Context, Intervention, Mechanisms and Outcomes
COTS	Commercial off the self
CSA	Cloud Security Alliance
IaaS	Infrastructure as a Service
LSM	Legacy System Migration
OMG	Object Management Group
PaaS	Platform as a Service
SaaS	Software as a Service
SecR	Security Requirement
SLR	Systematic Literature Review
SMART	Service Migration and Reuse Technique
SOA	Service Oriented Architecture

1. INTRODUCTION

1.1 Background and motivation

Software has become an integral part of human life which has been there since decades and is evolving along with the human civilization. Once the software is made available to the end users, its contest against the time and aging starts [1]. The pace of software evolution is very rapid. It also signifies the saying that “legacy code is code written yesterday” [1]. There has been a continuous change regarding the software development and delivery methodologies.

The software that is developed with the modern methodologies are very efficient in terms of usage, performance as well as resiliency as compared to the legacy systems. Despite this fact, active software which is build using older technologies and methodologies still can be found. Such software or systems are known as Legacy Systems. Furthermore, the software systems which were made years ago with the help of old technology and methods but are still in operation and a crucial part of business can be defined as Legacy Systems [2]. These types of systems running in a modern environment are problematic in various aspects such as resource consumption, security, incompatibility to integrate with other systems and so on. Hence, renewing them to a newer technology should be considered to eliminate these issues.

Although being archaic, these systems lie at the core of any business operations and functions [3]. Therefore, continuous running of the system is essential as they directly affect in the revenue of the organization. In contrast, legacy systems are made from old technologies that have a high operational cost as well as high maintenance cost [2]. Likewise, other discrepancies include platform dependency, high resource requirements etc. resulting from the outdated technologies and methodologies. It is quite risky to replace such system whose uninterrupted running is important. Similarly, an organization also should invest significant amount of cost in terms of time and effort. It should be ensured that the main functionality of the system should not be hampered or should be improved at large [2]. However, an organization should consider modernizing this type of systems.

Modernizing a legacy system or migrating it to a latest technology is not an easy task as there is a high chance of failure due to numerous reasons such as: complexity, engineering process and software technology and so on [1]. It is a long-term process which goes through series of processes having various risks. The risks with software modernization are also high but all the risks cannot be considered as bad risk. Some of the risks should be taken to achieve something important. It would be beneficial to foresee the risks before starting the process of modernization [1]. Hence the modernization process is quite complicated and should be done carefully.

Various things need to be considered while migrating a system to a newer environment. There are various experiences showing that majority of the migration projects failure are resulted from insufficient or substandard planning [4]. A successful migration of any system should be always backed up by a proper plan and strategy that can mitigate the risks and speed up the process of migration. Therefore, this thesis aims to explore and evaluate some migration strategies based on their strengths and their weaknesses.

There are various types of legacy systems that exist in the case organization. The outcome of this research will serve as a reference which can be taken within the organization while migrating the legacy systems. Hence, this research strives to recommend the best possible strategy depending on the nature and functions of a legacy systems.

1.2 Research objectives

This research is guided by the everyday problem that is faced by the stakeholders who are using as well as developing a Legacy System. The outdated technologies of the system could not cope with the modern requirements easily and the delivery process is quite hectic. This created an urgency within the case organization to migrate the existing legacy systems to newer environment.

The first step towards migration would be the planning phase which should be guided by a suitable migration strategy. The migration strategies can vary with numerous factors such as platform, technologies and so on. Therefore, there was a requirement of finding and evaluating various strategies so that a suitable strategy could be selected depending on the type of a legacy system. This requirement helped in sketching the topic as well as the research questions of this thesis.

The main objective of this research is to compare various migration strategies that can be used for renewing a legacy system. To meet these objectives, we need to identify what are the possible migration strategies along with their strengths and weaknesses. It aims to answer the following research questions.

RQ1: What are the most common factors that are considered for migrating a legacy system?

This first research question seeks to find what type of factors play a crucial role while considering the migration of a system. The most crucial factors can be identified which can serve as a basis or strategy for the migration process.

RQ2: What are the possible strategies for migrating a legacy system?

The second research question is the major focus of our study that complies with the objective of this research. The aim is to accumulate and analyze various strategies that have already been used for a successful software migration.

RQ3: What are the strengths and weaknesses of the possible strategies?

The third research question concerns evaluating the strategies selected based on their strengths and weaknesses. This question's purpose is to suggest the best strategies based on the legacy system's nature.

1.3 Thesis structure

This thesis consists of six chapters. The contents of each chapter are described below in brief below.

Chapter 1 Introduction includes the general information about the objectives and the problem this thesis wants to solve. It also consists of other aspects such as motivation, background as well as the structure of this thesis.

Chapter 2 Theoretical background focuses on the establishing a theoretical base for the thesis. Various concepts and facts are collected, analyzed as well as interpreted in this chapter with the help of literature review. It highlights the concepts of legacy systems,

software migration and their related topics. The main motive is to answer the first research question which concerns to the things that need to be considered while migrating a legacy system to a newer environment.

Chapter 3 SLR research methodology analyses the main research methodology that is used for conducting this research. It walks through the steps that is required for the systematic literature review.

Chapter 4 Overview of results is the major chapter of this thesis as it concerns with analyzing the results which are extracted from various data sources with the aim to answer all the research questions that are defined for this research.

Chapter 5 Discussion and conclusion summarizes the overall research process and the result that was gathered with this research. It also consists of what were the limitation of this research and what could be improved or done in the upcoming similar research.

Chapter 6 Research summary gives an overview of overall research. It includes various things such as the goal, research process, findings and so on of this research are discussed in brief. Likewise, also the things such as what went in a right an what went wrong.

2. THEORITICAL BACKGROUND

2.1 Overview of legacy systems

Legacy systems refer to those system that were built in a different way as compared to the methods that are used in the present even though they were the best practices and techniques available at that time. A system can be considered legacy even if they are fulfilling their requirements and is being actively developed but is still using the old methodologies [5]. Sometimes it can be those hefty and old systems that are vital for the organization, but the organization lacks competency to cope with it [7].

The evolution of software as described in [37] states that like other domains, software experiences an ongoing progressive changes or modifications. A software needs to undergo changes to adapt and enhance their fitness in a dynamic environment where they operate. If a software fails to meet these requirements, it will start to face several consequences to catch up with the pace of the growing technology. This will start the aging of the software making it a legacy.

Most of the time, the software is not updated because it has been developed outstandingly and works quite well for many years and the organization or management doesn't see any value on investing for its modernization. This gives birth to a legacy system whose platform is already outdated within couple of years because of the rapid changes happening to the technologies [5]. When a software system is serving its purpose well, the major chunks of the software budget is allocated for the maintenance. The regular maintenance activity will help to cope with the errors, bugs, and some adaptation to the change in the software environment. However, it still cannot prevent the software maturity or being legacy. With the passage of time, there is a requirement of modernization as well as system replacement at large [1].

Legacy systems does not refer to only those systems that are unusable or are not used anymore. These systems still support the core processes and operations of any business organization, but they are highly affected by the issues such as maintainability as well as scalability [10]. These systems are still considered as a precious asset to many organizations although there is an availability of newer or improved technologies and techniques. Any organization is compelled to maintain its legacy systems because they deliver a highly customized functionality for its customers that cannot be easily or quickly

implemented to or replaced by a new software solution. Hence, we can define legacy system as those systems that are still running and meet the substantial portion of the functional requirement of any organization but do not act in accordance with the emerging architectural and technical standards [11].

A legacy system mirrors the strategic intentions and operation logic of an organization during its initial implementation. With the passage of time, these logics gets established socially within the organization that creates a strong balance between the social and technical needs of the organization. This also creates an alignment of task, structure, and people around the operation of the legacy system [8]. If we considered a case example where a retail business had implemented a legacy inventory management system used for tracking inventory levels that utilized manual data entry methods. Therefore, the employees were trained to perform the manual data entry related activities and hence their tasks were shaped in a similar way. Introducing a modern inventory system with real-time inventory tracking can be challenging as new methods, technologies also need to be disseminated to the employees affecting the overall operation of the organization. This creates additional difficulty in renewing or replacing the legacy systems as it will also impact the organization culture and can demand changes regarding various aspects such as workforce, technology and so on.

Likewise, legacy systems are built on the top of various components such as programming languages, codes, technologies, and standards that belong to an earlier time or generation. These components or constituents get older and slowly obsolete with the passage of time creating a difficulty in proper functioning as well as maintenance of any system. They can be considered as the major contributors for making a system legacy [12]. Not only the technical aspects, but there are also other aspects of a legacy system such as business processes, policies, and regulations. Therefore, the legacy system also has the old ways of doing the things that might be quite arduous to change. So, changing or renewing a legacy system does not only mean adapting a newer the technology but also the newer business processes, policies and regulations that are also rapidly changing when a business moves forward [12].

Legacy system’s modernization and modification is not easy. The system’s failure or unavailability can have a significant effect to the business as they are very crucial for the business continuation. These systems have been running since decades and they have already demonstrated their effective functionality in a production [13]. This signifies that a system to be legacy, it should be old and outdated but on the other hand it should also

be the core or critical system of any organization that plays a crucial role in their business and operations [13]. These systems have been there for an extended period and have gone through numerous modifications over time. As a result, they also hold many valuable business information that is substantial to the organization [12].

In a nutshell, legacy systems can be defined as those systems that are developed using old or obsolete technologies, but their continuous running is crucial to any business. The rapid advancement in the technology makes the usage and running of such systems quite daunting and it cannot meet the expectations of the stakeholders [12]. On the other hand, the cost of maintenance, modernization or modification of these systems are quite costly as well as time consuming [12,13].

2.2 Problems with legacy systems

A software system starts to become problematic when it starts aging. With the passage of time, there are myriads of problems creating some kind of urgency in modernizing any system. They even prevent or make it difficult to further develop or maintain such systems. The problem can vary from some general to more serious or crucial ones. The issues or problems with the legacy systems should be identified and taken into consideration when we are taking in a legacy project or system. It will help the stakeholders to be aware of the expected issues and create some ease in dealing with them [5].

These systems resist modification as well as evolution creating are several problems to the organization [9]. Some of the problems of legacy system are listed below:

1. Mostly legacy system operates on an old or obsolete hardware whose performance is slow. This hardware is also costly and hard to maintain [9].
2. There is a lack of proper documentation, or the documentation is quite obscure which is understood only by a few people. [5,9].
3. The interfaces of such systems are quite complex making them difficult to integrate with other systems [9, 10].
4. A Large legacy system’s design and implementation start to become illogical and inconsistent over a longer period creating difficulties in understanding the system [5].

5. Difficulty in maintaining and reengineering the older technology and codebase for any organization because of lack of professionals as these codes and technologies are not included in the modern codebase and technologies [10].
6. In most of the cases, these systems lack high computing capability, optimum utilization of resources and they have a high electrical consumption as well [7].

2.3 Legacy system migration

Software migration focuses on improving any software systems in terms of availability, maintenance as well as its usability [6]. It is a process where a legacy system is required to be redesigned and migrated to a modern platform that supports the new or emerging technology as well as principles. Any software migration should start with a plan that requires a deep understanding of both current and the desired state of any software system [6]. The migration plan needs to be examined meticulously so that the existing functionality of the systems are preserved during the migration process. Therefore, migration should ensure that the new changes are not overwriting the existing functionality but are improving them in terms of usage, efficiency, and scalability.

A legacy system is crucial to any organization as it is the core system that supports the major business processes. Such system cannot be disposed and redeveloped from scratch as it can be costly as well as risky [12]. Hence, the system should be slowly moved towards new technologies ensuring their functioning is not hampered and the organization should not bear additional loss. When a system is developed and released for usage, it also needs periodic maintenance. Such periodic maintenance also helps in modernization to some extent but as the system ages, the maintenance alone is not enough. It creates a necessity to migrate the system to the newer environment or replace it at large [1]. It can be shown with the help of following figure.

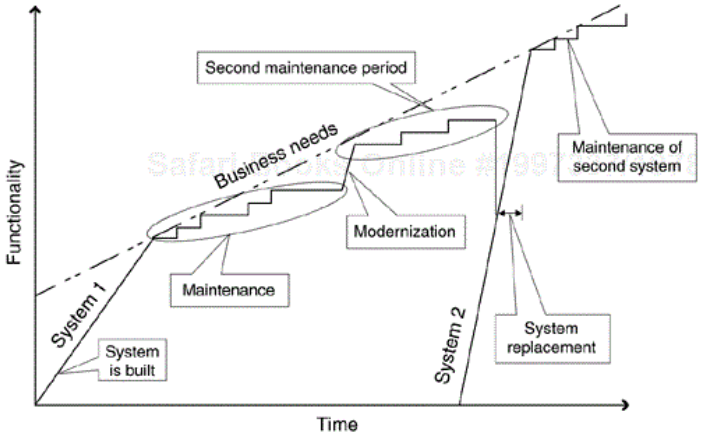

Figure 2.1 System lifecycle with respect to functionality and time [1]

In the above figure, it is shown how a system requires evolutionary activities when it starts to race against the time. The organization should decide the right course of action to select what they want to do with the system. If any enterprise has a tight budget, they must target for the best return of investment. Hence, for choosing the correct actions, the organization must carefully evaluate their legacy systems and must select the most suitable advancement strategies [1].

Migrating legacy system is an ongoing process which involves implementing a suitable approach when the existing maintenance operation prove insufficient to meet the emerging challenges [12]. Making changes to the legacy system is not an easy task. It should be always supported by the organizational culture. The working environment should encourage risk-taking, implementing the changes and occasionally breaking the things [5].

2.3.1 Reasons for migration

Likewise, a legacy application has limitations in directly leveraging modern hardware development such as multicore architecture and adapting the latest software paradigms. There is a rapid emergence of new technologies in the service-oriented systems to support the expanding and diverse needs of the users. Failure to adapt these changes may result in the legacy systems missing out on the possible business opportunities [10].

Software system needs to catch up with the pace of the changing environment where it operates. The software should evolve gradually to support new capabilities along with improving performance, fixing defects, functionality refinement and extension. When the reliance on the software grows within the operations and behaviors of any organization, delay in software renewal activities results from mere frustration to catastrophic outcomes [37].

Poor documentation can lead to various issues including difficulty in maintaining that results inflexibility in the system to adapt the changing and emerging technologies [15]. If a problem is encountered, its solution would be delayed as the person needs to spend most of the time understanding the system to find out the root cause. Therefore, to generate a good documentation, a system needs to be migrated by using a technique like reverse engineering or modern driven engineering [30].

Hence, the migration of legacy system should be considered. It refers to the process of upgrading aging applications to facilitate communication with the latest technology. This process often prioritizes application agility ensuring that the system can adapt or respond to the changes associated with the business requests [12]. The most common or reiterative reasons for modernizing a legacy system includes requirement for integrating legacy systems to modern systems, the necessity of enforcing the software flexibility, insufficient knowledge about the system, and the frequent occurrences of errors when maintaining an existing system [14].

Finally, technical debts are also a major contributor for the need of migrating a legacy system. The technical debt arising from the accelerated software project development leads to various shortcomings, ultimately resulting in increased maintenance costs [20]. The speeding up could be a result of legacy system trying to cope with the modern requirements. For accommodating such requirements, these outdated systems require some workarounds or hacks which gives birth to the technical debts. However, the technical debts could be worse if there is a failure in replacing the system fully or completely [8].

2.3.2 Phases of migration

Migration of a system is a phase wise process that involves various steps forming a cycle. These phases need to be carefully executed for migration to be successful [15, 16]. It can be demonstrated with the help of the following figure.

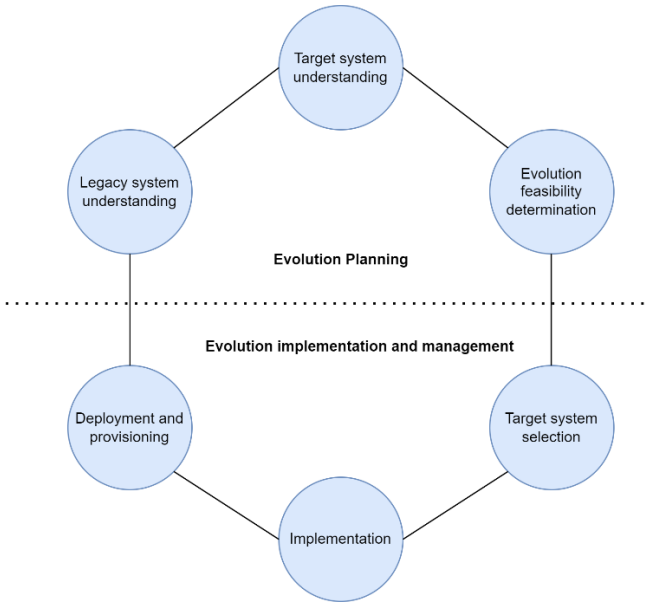

Figure 2.2 Phases of migration. Adapted from [16].

Understanding the legacy system:

This is the first phase where the legacy system under consideration should be thoroughly studied under various grounds such as architecture, codebase, requirements and so on. Also, some techniques such as reverse engineering can be used for understanding the program [16].

Understanding the target system:

In this second phase, the target system also should be studied based on architecture, technologies, standards and so on. It also involves identifying the current functionality of the legacy system [16]. Other activities involve introducing major components or functionalities as well [15].

Determining the feasibility of migration:

Comprehending both the legacy system and the target system is crucial while conducting the feasibility study of the migration. The assessment can be various level such as technical, economic as well as organizational level [15]. It also involves analyzing the code complexity of the legacy system by evaluating various factors such as: coupling, cohesion, reusability, and abstraction [16].

Identifying the candidate service:

Legacy systems have been running since many years and going through various development as well as maintenance by different people. This creates a high possibility of code redundancy. There are also poor or missing documentation which makes studying the legacy system a bit harder. In such case, the service rich areas of the legacy system need to be identified which could be extracted as a service. Then those service can be considered for migration [16].

Target system development / implementation:

This phase analyses and utilizes the requirements which were discovered in phase 1 and 2 and implement them to develop the target system. These requirements and implementation will be scoped using various methods such as program slicing, concept slicing and so on [15,16].

Deploying and provisioning the target system:
After the development of the target system is completed or done to some extent, it should be deployed to its suitable infrastructure. It should be made available to the users or customers who are using the system. Likewise, the provisioning should also be done after deployment which includes activities such as publishing, versioning, billing and so on [15,16].

2.3.3 Legacy system migration challenges

Legacy system migration is not an easy process as it is full of different challenges and risks. It also requires various resources that can be both technical and non-technical, making it also a quite expensive process. There is also a need of proper understanding and awareness of the new platform in which the system is planned to be migrated. Paucity of these things can result in the failure of migration to achieve the goals of the organization [7].

The major challenge for any software migration is to migrate the existing system in a guided way by ensuring that the overall functioning of the system is preserved or improved. The difficulty arises when there is a lack of knowledge within the team having more familiarity with the newer technologies. Different expertise concerning various knowledge and skills such as database environment, middleware, architecture of both previous systems as well as the system after migration is needed. It can also be quite hard to find as some of them have already left their jobs. Also, there is lack of proper documentation which exacerbates the situation [13].

During the migration, the organization not only faces complications regarding the technical part but also for other parts such as business, organizational and environment part. The legacy systems also set a legacy way of working to which employees are familiar. This can result in resistance to change according to the new or improved system which can slow down the organizational processes. Likewise, the integration of the software and business processes introduces another challenge of modifying the business processes which are generally rigid and unfeasible [10, 12].

Time constraint is also a major challenge as the migration process cannot last forever. This constraint is also a result of the ongoing issues of the legacy systems such as dearth of resources for instance experts and documentation. There are cases when the modernization goes for a longer period and needs to be restarted again from the

beginning [13]. Similarly, the legacy systems are built without considering about the architecture of the new platform where it is to be migrated. For example: if we are planning to move to a cloud environment, legacy systems may not have the capability to dynamically scale up and scale down the resources. This creates an additional overhead about studying and planning the new architecture profoundly to make migration successful [7].

2.4 Migration strategy

Legacy system migration is a long process which requires various factors to make it successful. Among various factors, the selection of strategy that is used for migration is very crucial. The word strategy is defined as a course of actions which are designed to accomplish a long-term goal which are usually uncertain. Therefore, the definition of strategy itself complies with the software migration which is no doubt a long process having various uncertainties. Software migration always commences with first evaluating the reasons for migration followed by presenting the migration alternatives or the strategies [4].

Migration strategies are essential to facilitate the transition of a legacy system to a desired environment. The strategies should be meticulously planned and diligently executed as their effectiveness significantly influences a successful migration process. Likewise, such strategies help in preventing the potential disasters and aids organization to update their legacy system [21].

Migration strategies must always ensure that the old functionality of the system is preserved assuming that they are really needed along with some enhancements as well as extended features added. It can be depicted as: New system = old system ++ [4]. The alternative strategies for migration can be seen from the following figure:

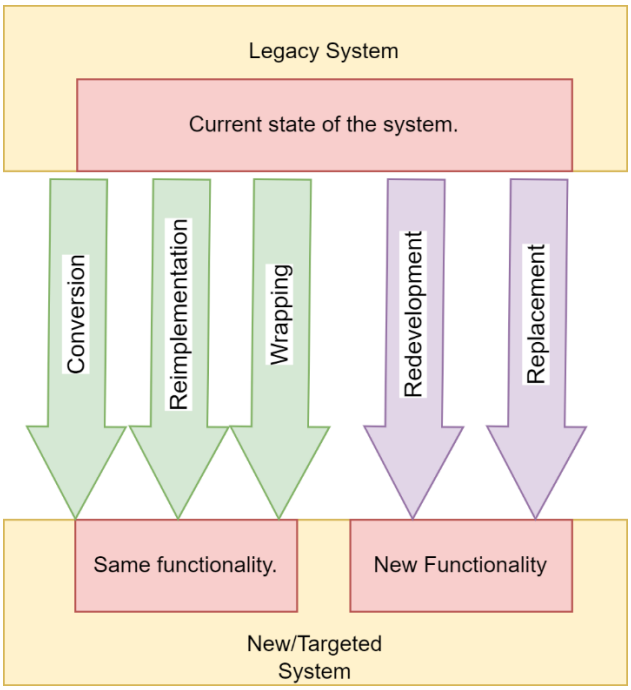

Figure 2.3 Migration strategies alternatives. Adopted from [4]

In the above figure the new or the targeted system is the combination of existing functionality along with new functionality. The existing functionality can be converted by using various methods such as conversion, reimplementation and wrapping. Whereas the new functionality can be done by either redeveloping the feature from the scratch or replacing some existing functionality by a new one through replacement.

Firstly, conversion refers to either a manual or automated process to convert a software code directly from one language to another. Reimplementation is different than conversion as it is a three-step process that includes: first translation to intermediate design language, second processing of language to improve its structure and the third is to use it as a basis to produce new code once the desired quality is achieved. Lastly, wrapping involves a process of adding a new interface by keeping the base code unchanged [4, 25].

For implementing the above-mentioned strategy, we can use the various migration technique or method. Among them the most pragmatic one is incremental migration. It can be considered as the most pragmatic method in case of big systems. In this approach a system is divided into multiple smaller components which could operate independently and could be tested within a short period [4]. It is a prevailing technique which is adopted by various strategies to mitigate the risk as the whole system is not affected [22]. It breaks

down the migration process into smaller, more manageable steps that allows to address each aspect methodically by ensuring a smooth transition of a system [25]. However, using this strategy for migration can take a long time to complete and involves a high risk [4, 10].

Above mentioned strategies are the core strategies on top of which other strategies are derived. These strategies should be chosen by considering various aspects such as size of system, available resource and so on. There are additional strategies which can be used for migration. They are discussed in the section 4.3.2.

3. SLR RESEARCH METHODOLOGY

This chapter deals with the main methodology that is the foundation for this research that is Systematic Literature Review (SLR). It is a process which involves identifying, assessing, and interpreting all the existing research that is pertinent to a specific research question [17]. This chapter starts with a review of the previously stated research questions in the introduction chapter and elaborating them more in detail. There are also other several steps followed which includes creating a review process, carrying out systematic review, analyzing the data, reporting, and visualizing the data and discussing the findings and conclusions [18]. The SLR process can be visualized with the help of the following figure:

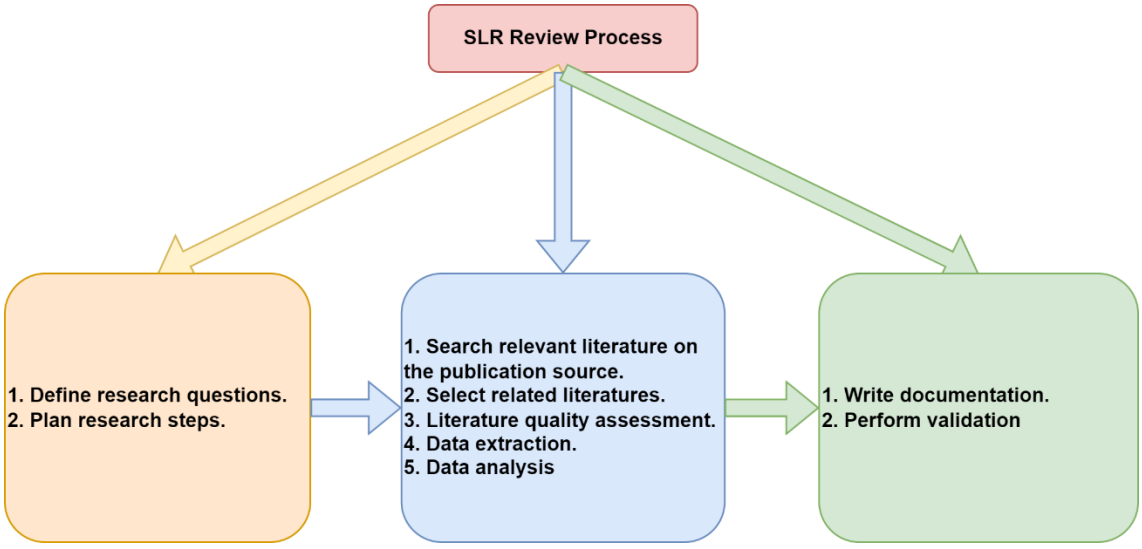

Figure 3.1 SLR review process. Adopted from [17].

The systematic literature reviews are mainly performed to mark any lacking in the ongoing research with an aim to recommend the areas for further research. It also summarizes the research that has been done on a particular technology, such as the advantages and disadvantages of a particular agile approach [17]. Nevertheless, we will be examining all the relevant research papers or literature thoroughly to meet our research goal.

3.1 Research questions

One of the most integral parts of systematic review is to specify the research questions as it steers all the systematic review methodology [17]. This methodology begins with the search process which should pinpoint the major studies that would address the stated research questions. Likewise, both the data extraction and the data analysis process retrieve and amalgamate the data respectively with the prime motive of answering the research questions [17].

Briner & Denyer have created a framework called CIMO (Context, Intervention, Mechanisms and Outcomes) to provide precisely defined and addressable research questions [19]. With reference to this framework, the research questions can be shaped as follows:

- Context - All research papers that concerns migration of legacy system.
- Intervention – Legacy system migration strategies.
- Mechanisms – Literature survey.
- Outcomes –Significance of migration strategies.

The research questions stated in the first chapter are also shaped based on above framework. They can be mapped with the above scheme using the following table:

Research Questions:	**C**	**I**	**M**	**O**
RQ1. What things need to be considered while migrating a legacy system?	✓		✓	
RQ2. What are the possible strategies for migrating a legacy system?	✓	✓	✓	
RQ3. What are the strengths and weakness of the possible strategies?	✓	✓	✓	✓

Table 3.1 Mapping research questions with CIMO scheme.

3.2 Search strategy

Systematic reviews mainly concern with discovering different available primary studies that are relevant to the stated research questions with the help of neutral search strategy [17]. Selecting a suitable search strategy helps to find the suitable literatures which could be used for the comparison or conducting a systematic review. The strategy that was used for this research is manual searching and browsing through various digital libraries which was itself an iterative process.

The initial iteration began with a simple search strategy which started by splitting the topic and finding the basic keywords such as:

evaluation	*strategy*	*migration*	*legacy system*
analysis	*technique*	*modernization*	*archaic system*
assessment	*plan*	*upgradation*	*outdated system*
	tactics		*obsolete system*

Table 3.2: Splitting of topic for keywords

In the next iteration the above keywords were grouped together to form a basic search term such as:

"Strategies of migration of legacy systems"

"Evaluation of plan of migrating outdated systems".

"Analysis of techniques of modernizing an obsolete system".

Finally, in the last iteration, more refined or improved search strings were created using the Table 3.2. The above table was modified further in the following way:

Terms joined by OR	**AND**	**Terms joined by OR**	**AND**	**Terms joined by OR**	**AND**	**Terms joined by OR**
evaluation		*strategy*		*Migration*		*legacy systems*
analysis		*technique*		*modernization*		*archaic systems*
assessment		*plan*		*upgradation*		*outdated systems*

Table 3.3: Combination of keywords

With the help of the above table, the below search strings were formulated. They are more logically connected.

("strateg*" OR "plan*" OR "method*" OR "technique*") AND
("migration" OR "upgradation" OR "modernization") AND
("legacy systems" OR "outdated systems" OR "archaic systems")

The above given search strings were used for querying various digital libraries to search for different research articles, papers, books. Due to the large number of articles, only the papers which were published between the period of 2013 to 2023 were considered. With the guidance of the inclusion and exclusion criteria mentioned in the section 3.4, all the relevant literatures were extracted and considered for the systematic literature review.

3.3 Data sources

Literary database are the major sources of data for this research. These databases play a crucial role in supporting Systematic Literature Reviews (SLRs) by reducing the biasness arising from individual literature choices (selection bias). Likewise, their organized information makes it easier to do a methodical search that minimizes retrieval bias by seeking to find all relevant entries on a certain subject. The following literature database were used for retrieving all the relevant literatures:

1. IEEE Xplore
2. ACM Digital Library

3. Science Direct
4. Scopus
5. Springer Link

The above literature databases were selected as they were accessible from the university account. They were suitable as several literatures could be browsed without any hassles. Likewise, the search string generated in the section 3.2 were used for searching the documents.

3.4 Inclusion and exclusion criteria

Going through all the literatures which were found in the database would be overwhelming as well as prevented by time constraints. Therefore, some inclusion and exclusion criteria were defined to narrow down the literatures which were best for the research and closer to the research questions. The inclusion or exclusion of the literatures were determined by going through the topic and the abstract. For some texts, there was a requirement of reading some additional chapters.

Inclusion criteria

- Literatures which were published between 2013 to 2023.
- Literatures that are written in English.
- Literatures concerning migration of legacy system to a newer architecture.
- Literatures mentioning strategies for migration or modernization.

Exclusion criteria

- Literatures which were published prior 2013.
- Literatures that were not written in English.
- Literatures that are not a scientific publication.
- Literatures having duplicate titles.

3.5 Data extraction

Data extraction is a systematic process of gathering or retrieving data from a range of sources using some tools or techniques. The objective of data extraction is to identify and gather all pertinent information required to address the Systematic Literature Review's research questions from the incorporated studies.

For extracting the data efficiently, a data extraction form was used. The form serves as an organized framework that is used to record key details about each study that we include in our systematic review. It ensures uniformity in the data extraction process and offers a standardized way to record the obtained information. The form which was used for the data extraction consists of the following fields:

Field	Description	Usage in research
Title	Title of the research papers, articles, journals.	Reference
Authors	Authors of the studies.	Reference
Publication year	The year when the literature was published.	Reference
DOI/Link	Digital Object Identifier of the studies.	Reference
Publication Type	The publication type used for conducting the studies.	Reference
Most common factors that are considered for migrating a legacy system.	The most factors that are taken into consideration for legacy system migration.	RQ1.
Legacy system migration strategies.	Available strategies/techniques for migrating a legacy system.	RQ2.
Strengths of the strategies.	Strengths or significance of the legacy system migration strategy.	RQ3.
Weakness of the strategies.	Weakness or drawbacks of the proposed strategies.	RQ3.
Other relevant information.	Other related information of Legacy system used as the theoretical background.	Other.

Table 3.4: Data extraction form.

The above specified data extraction form was filled while performing study for each literature. These fields were used for different purposes throughout this research such as referencing the information of the study, forming the theoretical background as well as collection information that can be used for answering the specified research questions.

3.6 Data synthesis

Data syntheses is next step that comes after extracting and collecting the data from the various sources. It involves gathering and summarizing the outcomes of the incorporated primary studies [17]. It is a process of integrating the information which comes from multiple data sources.

Usually, the synthesis is a descriptive approach or non-quantitative, but it can be made quantitative by adding some statistical techniques which is also known as the meta-analysis. In the descriptive approach it is crucial to ascertain whether the findings from the different studies exhibit consistency (homogeneity) or variation (heterogeneity). Likewise, in quantitative synthesis, it is necessary to present the quantitative outcomes from various studies in a comparable manner [17].

The approach used for synthesis either descriptive or quantitative depends on the nature of the research questions. In the context of this research, the descriptive synthesis qualifies for all the three research questions. Whereas the quantitative synthesis can be applied to the third research question where the strengths and weaknesses of each strategy can be expressed in terms of different matrices such as cost, time, resource, security and so on.

3.7 Limitations

SLR is an excellent research approach which provides a comprehensive and a structured approach to review the literature that are considered. However, there are several limitations with this method. To begin with the shortcomings, it was experienced during the data extraction phase. The extracted literatures were heterogeneous as they varied in terms of structure, interventions, results and so on. Likewise, the inclusion criteria helped to narrow down the research focus but also affected for selecting some good literatures that were not in the inclusion criteria for example due to date of publication.

4. OVERVIEW OF RESULTS

4.1 Search results

Searching for appropriate literatures across various literature libraries or database is one of the major activities involved in systematic literature review. The results from all the databases were collected using the search string that was created in the section 3.2. There were quite many literatures that were found during the search. The search criteria defined in the section 3.4. helped in scoping the search results and selecting only those which were needed.

On top of that different filtering options in each digital libraries were selected to get a more precise result. The number of related literatures found were quite high and only those meeting the requirements were selected. The highest number of documents were found in ACM Digital Library which required quite significant effort to select the literature that was brought into consideration.

Literature Database	Literature found	Literature considered
IEEE Xplore	57	5
ACM Digital Library	185	6
Scopus	182	2
Science Direct	86	6
Springer Link	97	3
Total	607	21

Table 4.1: Total literature found, and total literature considered for SLR.

The literatures found were first analyzed by going through the title and then by reading the abstract. These search result also consist of the literatures from above mentioned database which made the search results quantity quite large. Among the 607 literatures, 21 were considered initially dropping 586 literatures. Likewise, two articles published in Wiley online library were among the search results which are also taken into consideration. One relevant article was also found in Google scholar which was also included making it total 23 literatures. By going through the literatures thoroughly, some looked quite promising when reading title, but the contents were not much relevant to this research. The inclusion and exclusion criteria also came handy to filter out the literatures. Among the 23 literatures which were considered, only 18 were selected for

conducting the literature survey. Further 5 literatures were dropped as some were vague. They were irrelevant when going through as they did not mention any strategy clearly and some were more specific to a particular programming language. All those articles which were selected were available online. The articles used for this survey are listed in table 4.2.

Ref no.	Literature title	Authors
[4]	Cost-driven software migration: An experience report.	Harry M. Sneed, Chris Verhoef
[23]	A Framework for Architecture-driven Migration of Legacy Systems to Cloud-enabled Software	Aakash Ahmad, Muhammad Ali Babar.
[24]	Pattern-based Process for a Legacy to SOA Modernization Roadmap	Frank J. Frey, Carsten Hentrich, Uwe Zdun
[25]	Modernization of Legacy Systems: A Generalized Roadmap	Suman Jain, Inderveer Chana
[15]	A Survey on Survey of Migration of Legacy Systems	A.Sivagnana ganesan, T.Chithralekha
[26]	Towards a Topology for Legacy System Migration	Stefan Strobl, Mario Bernhart, Thomas Grechenig
[18]	Migration of Existing or Legacy Software Systems into Web Service-based Architectures (Reengineering Process): A Systematic Literature Review	Siti Rochimah, Alhaji Sheku Sankoh
[10]	Systematic review of legacy system migration	Bashair Althani , Souheil Khaddaj
[27]	Cloud Migration Research: A Systematic Review	Pooyan Jamshidi, Aakash Ahmad, Claus Pahl
[28]	Model-driven software migration: A methodology; Reengineering, recovery and modernization of legacy systems	Christian Wagner
[29]	Re-implementing a legacy system	Harry Sneed, Chris Verhoef
[30]	Architecture Driven Modernization: A Review on Reverse Engineering Techniques based on Models' Approach	Mohamed Karim Khachouch, Ayoub Korchi, Younes Lakhrissi
[31]	Legacy systems to cloud migration: A review from the architectural perspective	Muhammad Hafiz Hasan, Mohd Hafeez Osman, Novia Indriaty Admodisastro, Muhamad Sufri Muhammad

[32]	A Framework for Secure Migration Processes of Legacy Systems to the Cloud	Luis Márquez, David G. Rosado, Haralambos Mouratidis, Daniel Mellado, and Eduardo Fernández-Medin
[33]	A taxonomy of service identification approaches for legacy software systems modernization	Manel Abdellatif, Anas Shatnawi, Hafedh Mili, Naouel Moha, Ghizlane El Boussaidi, Geoffrey Hecht, Jean Privat, Yann-Gaël Guéhéneuc
[34]	Microservices Migration in Industry: Intentions, Strategies, and Challenge	Jonas Fritzsch, Justus Bogner, Stefan Wagner, Alfred Zimmermann
[35]	From Monolith to Microservices: A Semi-Automated Approach for Legacy to Modern Architecture Transition using Static Analysis	Mohd Hafeez Osman1, Cheikh Saadbouh, Khaironi Yatim Sharif, Novia Admodisastro and Muhammad Hadri Basri
[36]	Towards an Architecture-Centric Methodology for Migrating to Microservices	Jonas Fritzsch, Justus Bogner, Markus Haug, Stefan Wagner, and Alfred Zimmermann

Table 4.2: Literatures considered for review.

4.2 Literature quality assessment

Quality assessment is a crucial activity that needs to be done before we include the results or findings of the literatures that are taken into study. It is beneficial in various ways in terms of effective result collection. It can be taken as an extended inclusion/exclusion criterion as well as it will provide an effective guideline to minimize the gap between the expected results and the actual results [17].

4.2.1 Assessment questions

The quality assessment is done with the help of the assessment questions that are provided by Kitchenham B. [17] in terms of various phases of research such as Design, Conduct, Analysis and Conclusions. Among the various questions, one question per phase were taken which were more relevant to the research questions stated for this thesis.

S. N	Phase	Assessment Question
AQ1.	Design	Are the measures used in the study the most relevant ones for answering the research questions?
AQ2.	Conduct	Are the data collection methods adequately described?
AQ3.	Analysis	Were the basic data adequately described?
AQ4.	Conclusions	Are all study questions answered?

Table 4.2: Assessment questions for quality assessment.

The first assessment question concerns the design phase of the research. It tries to make sure that the measures used in the studies are enough to answer the research questions of that research. It will help to examine the overall design of the research and mark it as good or bad. The quality score on the scale of 1-10 based on the coverage of measures on the number of research questions mentioned. If there are three research question and the measures used covers them all, then the research can be given score 10 and if only 50% is covered then we can give score 5.

The second assessment question deals with the conduct phase which seeks for the answer whether the methods used for collecting the data for research were sufficiently defined. The quality of data collection methods can be scored based on coverage, for example: full coverage, partial coverage, and no coverage. Full coverage will be given score 10, partially coverage will be given score 5 and no coverage will be given score 0.

Likewise, the third assessment question is related to the analysis phase where the collected data are examined. The response AQ3 expect is if the basic data used are defined adequately or not. This question can be answered either yes or no. The scoring technique for this assessment question is like AQ2 where the score of description of basic data can be scored from the range of 0 to 10.

The last question concerns the conclusion part of the research which is also focused on the defined research questions. This question makes sure that all these questions are answered in the conclusion part which is very crucial. The answer can also be expressed in the scale of 0-10 like AQ1, where full score will be granted only if all the stated questions are properly covered and so on.

4.3 Result of data analysis

A careful examination of all the selected literatures were done to extract results to answer the research questions. Each research question is covered in each sub section of this section to have a good overview. This is a crucial part of this research as it reveals the actual outcome for doing of this thesis. It tries to collect the research situation regarding the legacy system migration to find out trends and the most common practices. It focuses to extract the results in terms of consideration for migration, strategies of migration as well as the strength and weakness of those strategies.

4.3.1 Factors considered for migrating legacy systems

Mainly in the introduction section of most of the articles, listing of various problems with the legacy system along with the benefits after migrating the legacy system to a so-called modern architecture or environment can be found. There was not any specific mention of these things as considerations before migrating a legacy system, but they were the main catalyst that directly affected the migration process. Hence, those factors were grouped under one name as considerations.

While moving forward with migration, these aspects need to be considered for the legacy system which can highly affect the selection of any strategy. They can also be marked as the constraints as well as a guide to select an appropriate strategy. Such aspects can be defined as the considerations for migrating a legacy system. These factors are also responsible for leading the selection of a suitable environment or architecture of the targeted systems [25].

The prime consideration always will be the business needs and requirements. Other most common ones include programming language or code, operating system, architecture, data, user interface, cost, scalability and so on. A migration can be done either considering each of these components separately or can be done by combining them together. These components can be taken as the drivers of the migration which mainly assist in selecting an appropriate course of actions [15]. Hence these components can also be entitled as the considerations for migrating a legacy software. They directly affect the migration process.

In one of the articles [25], such factors were kept in the center of the decision making for migrating an application or were defined as decision-making modules. Furthermore,

this decision-making module not only consists of what is within the legacy system but also about the desired state of the migrated system. In addition, there are also other elements involved which make the migration planning more accurate as well as appropriate.

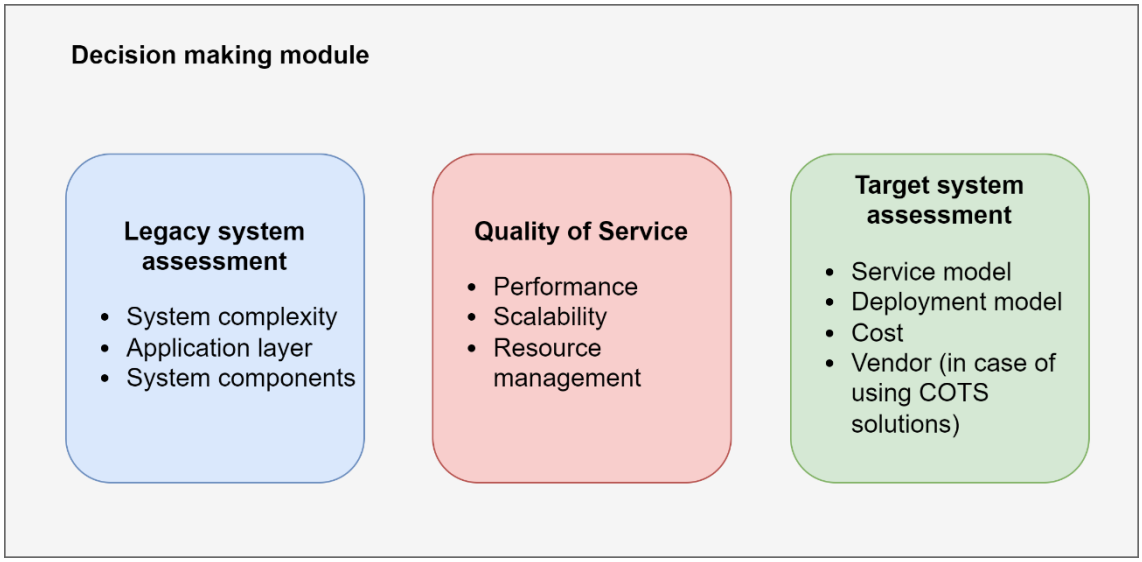

Figure 4.1 Decision Making Module. Adopted from [25]

In figure 4.1, the considerations for migrating a legacy system are classified into three categories: Legacy system assessment, Quality of service and Target system assessment. It tries to signify that there are various factors which can affect during the selection of any migration strategies. Firstly, the legacy system assessment is done that consists of various aspects such as system complexity (size, technology), the major three layers of a legacy system namely presentation, business and data layer and the various components of systems. Secondly, the quality metrices like performance, scalability, resource management. Finally, the target system assessment involving the elements such as service model, deployment model, cost, and vendor.

Cost, time, and effort estimation are the most crucial factors that help to decide why and how the migration of any system is done. In many articles, these are the most crucial elements that drive a migration of legacy systems. The running and maintenance costs of any legacy system are quite high. One aspect can be the resource consumption as legacy application cannot support the acquiring and releasing of resources as per need of the system. Therefore, the problem of over allocation of resources can be seen which are basically unused. However, it must be insured that the cost of operating the system in the target environment is lower than the system running in the legacy environment. In

addition, risks were also taken as one of the drivers of migration along with cost. In [27], it is mentioned that the major factors (organizational needs and requirements), which influences the strategy selection is predominantly shaped by the interplay of risk and costs.

Another major consideration mentioned is the type of the system that is to be migrated. The example given includes migrating of safety-critical and embedded systems to cloud architecture is not suitable as its operation can be highly affected due to factors such as latency. Therefore, there is a need to analyze the type of the legacy system and select the best modern substitute which can enhance the functioning of the system and minimize the cost and effort for maintenance.

There are several considerations or factors that are considered for migrations which were mentioned in the articles. Most of them were similar whereas some were unique or were more finely detailed. All the possible considerations which were found in the literatures are collected to the table 4.3 along with their presence in the selected articles.

Ref	cost	time	effort	scalability	architecture	type	size	risks
[4]	✓	✓	✓				✓	✓
[23]	✓	✓	✓	✓	✓	✓		
[24]	✓	✓			✓			
[25]	✓		✓	✓	✓	✓	✓	
[15]					✓	✓		
[26]							✓	
[18]					✓	✓	✓	
[10]	✓	✓	✓		✓	✓		✓
[27]	✓			✓				
[28]	✓	✓	✓					
[29]	✓	✓	✓					✓
[30]						✓		✓
[31]	✓			✓			✓	✓
[32]	✓			✓				✓
[33]				✓			✓	
[34]	✓	✓		✓	✓	✓		
[35]		✓					✓	
[36]	✓	✓						

Table 4.3: Consideration before migrating legacy systems.

In a nutshell, we can identify that there are various aspects we need to take care of before migrating a legacy system. These elements are very crucial for both legacy system as well as for the target system. It helps to visualize the things that are needed to be carefully analyzed and make the formulation or selection of suitable migration strategy more convenient.

4.3.2 Legacy system migration strategies

The selected articles comprise of variety of strategies which are used for migrating a legacy system. While going through them, it was found that most of the strategies concerned a specific target architecture. The selected target architectures include cloud architecture, service-oriented architecture and microservices architecture. In addition to that, some general strategies are also discussed as they are frequently used by these strategies. Some of these strategies are the underlying basis on top of which other strategies are formed.

To have more visibility on different types of strategies discussed among various articles, the strategies are grouped as four major groups and they are: general migration strategies, cloud migration strategies, SOA migration strategies and microservices migration strategies. The categorization of strategies with reference to the cited articles is presented in the table 4.4 below and discussed in detail in the following subheadings:

Strategy type	Description	Cited reference
General migration strategies	These strategies are categorized as general as there were no specific target architecture mentioned. These strategies can be used in all other listed strategies. They can also be called as other migration strategies.	[4][10][15][25] [28][29][30]
Cloud migration strategies	These strategies are mainly focused for migrating a legacy system to a cloud.	[10][15][23][25][27] [31]
SOA migration strategies	These strategies are mainly focused for migrating a legacy system to a Service Oriented Architecture.	[15][24][33]
Microservices migration strategies	These strategies are mainly focused for migrating a legacy system to a Microservices Architecture.	[34], [35], [36]

Table 4.4: Categorization of strategies with reference to target architecture.

General migration strategies

The articles from which the general type of strategies are extracted do not mention a particular target architecture. They are more focused on migrating systems to modern architecture by using various tools and techniques. These strategies can also be used by other migration strategies listed in Table 4.4. The general migration strategies include strategies such as conversion, reimplementation, replacement, redevelop, architectural driven migration, gateway, and non-gateway migration, cost driven migration, reengineering, model driven software migration and so on.

To begin with basic strategies which includes techniques like conversion, reimplementation, replacement, and redevelopment. Firstly, conversion refers to directly converting the old code to a new code either manually or automatically by using a tool. It can be considered as a cheap, fast, and straightforward solution which is mentioned in articles like [23], [29], [15] and so on.

The reimplementation method is thoroughly described in article [29] but it can also be found mentioned in another selected article [25] as well. It is a straightforward approach which suggests rewriting the existing source code to another language and redocumenting it. In article [29], there are ten steps mentioned which could be utilized for migrating a system using reimplementation. Likewise, replacement is also another solution where a legacy system is replaced by some COTS software that can be purchased from other vendors and can be used instead of the existing system. Finally, redevelopment is like reimplementation but in this approach, the totally new requirements are collected from the user and the system is developed from scratch.

Article [23] advocates the architectural driven migration which has extended the model stated by OMG's Architecture Driven Modernization (ADM) and mainly modified it as a horseshoe framework for migrating the components of a legacy system. Even though the article is more focused towards legacy to cloud migration, the steps mentioned in the initial process can also be used for general migration irrespective of the architecture. The major goal of this strategy is to transform an existing legacy architecture in an incremental way to the desired target architecture.

It begins with architecture migration planning where several things are considered such as feasibility study, amount of required effort and cost of migration etc. to select a migration strategy. The next step involves the recovery of architecture model which is

basically extracted in three step process: examining the legacy code, converting the code into model, and extracting the legacy architecture. This method resembles another migration strategy presented in [29] that is the model driven migration. The third step concerns the architecture transformation, that is the transformation of legacy architecture to the desired architecture which is basically a modern and most suitable one.

In article [15], two migration approaches or strategies are presented which were predominantly used for migrating a monolithic legacy system to a distributed system such as client server, service-oriented architecture and so on. They are Gateway Migration Approach and Non-Gateway Migration Approach. These approaches have the term “gateway” which refers to the requirement of accessing the legacy and the target system concurrently during the process of migration. Therefore, in the Gateway Migration Approach the legacy system needs to access the target system or vice versa during migration process whereas in the Non-Gateway Migration, the target system nor legacy system requires to access each other.

The gateway approach utilized the “gateways” which can be either forward or reverse. It refers to the concurrent access of both legacy and target system to facilitate an interoperation between the two dissimilar systems. If the legacy system needs to access the target system during the migration process, then it is called forward gateway and vice versa. There are four major approaches within the gateway described in table 4.5.

Gateway Approach Name	**Description**	**Gateway method**
Database first approach	As the name of this approach, the data is migrated in the beginning and other components such as interface and application logic are migrated afterwards in an incremental fashion.	It is a forward gateway method as the legacy system needs to interact with the migrated database.
Database last approach	Opposite of the previous method where the application is migrated at first and the database is migrated at last.	It is a reverse gateway method as the migrated application needs to communicate with the legacy database.
Composite database approach	Database and application are migrated hand in hand. There is a requirement of transaction coordinator which ensures the integrity between legacy and target databases.	Implements both forward and reverse gateway method as data is accessed back and forth.

Chicken little approach	Using this approach, the operational system will be the combination of both legacy and target system. Therefore, the system is migrated incrementally by ensuring the interoperability through required gateway method.	Both forward and reverse gateway method is implemented as it requires interoperability.

Table 4.5: Gateway migration approach.

Similarly, the non-gateway approaches do not care about any type of forward or reverse gateway as it is concerned with the whole migration process. There are two major types of non-gateway approaches mentioned in [15] which are described in table 4.6.

Non-Gateway approach name	**Description**
Big Bang Approach	In this approach, the whole system is re-developed from scratch using the latest architecture, tools and database which is compatible with running on the new platform.
Butterfly Approach	Likewise, this methodology is adopted throughout the whole migration process which focuses on gradual transition of system rather than keeping both (legacy and target) system in operation. Its focus is on legacy data migration.

Table 4.6: Non-Gateway Migration Approach

Article [4] talks address a different kind of strategy for migration that is the cost driven software migration. It is mentioned in the article that a migration process is always guided by the user requirements. The two major drivers of migration hidden inside user requirements are cost and time. If the user has more time but has a tight cost, at that time the cost driven migration strategy comes handy and vice versa.

According to the authors, the cost driven software migration estimates the migration cost based on four metrics. They are size, complexity, quality and productivity. Firstly, the size of the system can be calculated from the size of the codebase or source code. Therefore, the higher the size, the higher will be the cost, but the size of source code cannot alone estimate the cost as it is affected by complexity and quality.

Likewise, the system complexity is another metric that can affect the cost. The complexity can come in terms of coupling, cyclomatic complexity and so on. Code complexity plays a crucial role in estimating the cost of migration. The quality metric comprises of various aspects such as modularity, reusability, testability, flexibility, portability and convertibility. The higher the quality of the legacy codebase, lower will be the cost of migration. The last metric is productivity, which cannot be extracted from the existing source code. It can be observed from aspects such as the size of the system. The productivity in migrating a larger system is higher than migrating a smaller system.

Reengineering strategy is another way to migrate from a legacy system to a modern architecture. The concept of reengineering is defined in the article [28]. The reengineering consists of two types of engineering namely reverse engineering and forward engineering for converting a legacy source code to a modern or target source code. It begins with reverse engineering where the source code is analyzed to extract the legacy requirements. Then the legacy requirements are implemented using the new technologies to generate the target source code by using forward engineering methods.

In the same article, a more refined reengineering strategy is coined and is named as model driven software migration strategy. In this strategy, a meaningful model is extracted from the source code. These models focus on abstraction and remodeling of the functionality. The target source code is now generated with reference to these models. This method can be used at base level for migrating any legacy system.

Cloud migration strategies

With reference to the articles reviewed, cloud migration is found to be one of the most preferred migration strategy. The number of articles which are focused on migration to cloud environment are quite high which are published between 2013 to 2023 which meets the inclusion criteria of this research. A system is preferred to be migrated to cloud as it has significant benefits in terms of cost, performance, scalability and so on [10].

The article [23] demonstrates a legacy-to-cloud migration horshoe framework where a legacy system is slowly moved to a cloud-enabled software in an incremental manner. It is the same architectural driven migration methodology as mentioned in general migration strategy section but now the focus is on migrating to a cloud architecture. The other processes are quite similar but some more things are considered which are native to a cloud environment for example, in the first process which is architecture migration

planning, the options regarding the cloud providers are evaluated and the suitable one is selected. Likewise, the architecture recovery is also done on similar basis as explained in previous chapter. In the third process, the legacy architecture is now converted into a cloud-service architecture using the techniques such as atomic change operations and composite change operation. The fourth or the last process mainly focuses on architecture-based development where the target or cloud architecture is converted to source code which is cloud enabled.

Similarly in article [25], a rather simple process is explained. By using the decision-making module explained in figure 4.1, a suitable cloud delivery model is selected such as SaaS, PaaS and IaaS. The migration can happen either in wrapping the interface or reengineering where the main processes such as code identification, model extraction, decoupling are executed. There should be a high understanding of the legacy system and the stake holders are also involved during the process. The migration process involves various manual processes as it is not fully automated. The process of identifying the cloud environment constraints (CEC) such as security, bandwidth, downtime impact and so on is a manual process.

The first stage focuses on installing and configuring essential migration tools which are crucial for a smooth transition to the cloud environment. Then the part of code is converted incrementally based on priority which reflects the functionality of the existing system. The changed components will be hosted on the cloud using the selected service model and the specified deployment configuration. The testing of the application is also done at a regular interval to ensure that the migrated system is working as expected.

There are several techniques that can be used for migrating a legacy system to cloud. The articles [15], [27] and [31] describes various techniques which makes the migration process easier. These approaches include Cloud Migration Approach (CloudMIG), SOA Migration Adoption and Reuse Technique (SMART), Reuse and Migration of legacy application to Interoperable Cloud Services (REMICS), Advanced Software-based Service provisioning and Migration of Legacy Software (ARTIST), Cloud-Reference Migration Model (Cloud RMM).

To begin with the CloudMIG, in article [15], it is defined as an approach which uses the reengineering techniques for migrating a legacy system to cloud based application. There are various steps included in this step such as extraction of legacy code, generation of models or architecture, selection and evaluation of suitable cloud type,

service and provider and finally transformation into cloud system. The Cloud RMM method mentioned in [27] is a reference model which utilizes both the bottom-up approach and top-down approach and extract the common migration task. The bottom-up approach identifies and categorizes the low-level activities to form a high-level task and processes whereas the top-down approach derives a framework which includes various processes, techniques and activities. The Cloud RMM includes various process which concern various aspects of migration such as planning, execution, evaluation and addressing the cross-cutting concerns.

Likewise, the SMART approach is a derivative of SOA migration framework to implement in a cloud-based environment. As said in [27], the SOA and cloud migration are complementary areas, some techniques can be utilized for cloud migration as well. This method helps in examining the legacy system to migrate them to a cloud or a service-oriented platform. Likewise, in [15], it is further defined as a framework that is suitable for migration of a computing platform that supports performance software testing.

Finally, the REMICS approach [15] is beneficial in various ways as it is a model-driven methodology and utilizes both cloud computing and SOA patterns to migrate from a source architecture to a target cloud environment. The service can be an amalgam of replacing as well as wrapping of the legacy components. It is also capable of using various metamodel and UML modelling termed as PIM4Cloud to forecast the target cloud architecture. Likewise, the ARTIST [15] specializes on making use of advantage of cloud architecture rather than running on top of it. It comprises of three steps such as pre-migration, migration, and post-migration.

A legacy system which has very critical information or is very secured can migrate using a different strategy which is mentioned in [32]. This article proposes a strategy called SMiLe2Cloud which can be used for a secured migration of legacy system to cloud. It comprises of five activities which comply with sixteen security domains. The first activity is to use the reverse engineering strategy to extract and visualize the existing security issues. It is then followed by analyzing the security requirements (SecR) with the help of Secure Troop methodology for the cloud which includes cloud related security constraints. The third activity is to design a deployment and service model with reference to the CSA Security, Trust and Assurance Registry (STAR). The cloud deployment model can be selected among public, private, hybrid cloud or community cloud. Likewise, the service model concerns selection of cloud models such as SaaS, PaaS or IaaS. Fourth

activity concerns with the deployment to the cloud after a suitable service model is selected by ensuring security by signing a Service Level Agreement (SLA), custom security development and setting all the security controls active. The final step is to evaluate the target system in terms of various aspect. Major one would ensure that the basic system requirements are fulfilled as well as are secured. Furthermore, it also ensures that the new requirements coming would maintain the security standard.

Service Oriented Architecture (SOA) migration strategies

Service Oriented Architecture is another preferred architecture for migrating a legacy system. It is mainly a distributed system which is made from a monolithic legacy system. It enables any organization to build a convoluted and inter organization systems by integrating and composing reusable, distributed services that are independent and are most of the time diverse [33]. There are several strategies discussed in the article [15] [24] and [33]. They are described in brief below.

Beginning with the basic strategy explained in [33], the article introduces three strategies to migrate to a legacy system namely top-down strategy or forward engineering, bottom-top strategy or reverse engineering and hybrid strategy. The top-down strategy begins with high level breakdown of the legacy artifacts. The next step involves modelling the services needed that would make the SOA. The third phase is to implement the system and the last would be to execute the orchestration process needed for controlling the distributed services.

Likewise, the bottom-top strategy begins with extracting the dependencies from the existing legacy system. Then it is followed by searching for the reusable functionalities that can be extracted as service. The next step would be wrapping these functions as a service to ensure their reuse and remove their dependencies over the legacy infrastructure. The last step would be to make all available compatible with new services which might require rewriting them as well. Finally, the last strategy is the hybrid strategy which comprises of three steps. In the beginning the functions of the application are grouped to a rough functional block. Then these functions are mapped to the available services along with detaching them from the legacy infrastructure. At the end the process for orchestrating these services is implemented.

The roadmap for migrating a legacy system to SOA is mentioned in [24] also suggests similar processes that was discussed in [33]. It proposes a process executed in five stages: Setting Direction, SOA Architecture Definition, Service Identification, Service

Design and SOA Roadmap Planning. It derives its concept from the ArchPad design. It basically refers to making architectural decisions on every stage of migration. Among the five steps, first two stages utilize the first phase of the ArchPad design and later three stages are irrelevant to the ArchPad design.

The first stage that is Setting Direction is generally driven by executive decision which concerns business objectives and priorities as well as the IT transformation paradigm. The second stage SOA Architecture Definition is guided by conceptual decision and technology directive decisions. It includes aspects such as discovering reference architecture, selecting an integration, routing, interaction, and data transformation paradigm. Likewise, in technology directive decisions, the basic technologies which would be used for system development such as programming language are also decided.

The third stage Service Identification is also guided by the conceptual decision which concerns the service identification. The service identification can be either top-down or bottom-top approach as mentioned in [33]. In the fourth stage which is Service Design, design decisions are made. It follows the service design paradigm and decide either to build a new service or using the existing one. The fifth or the last stage is the Roadmap Planning which is guided by planning decisions. In this stage, a set of course of actions related to migration is created which decides the things such as consideration for fully replacing a legacy system and necessary aligning with the development, testing and rollout of the services.

Similarly, in [15], five approaches for migrating a legacy system to SOA architecture are mentioned. The first one is Service Migration and Reuse Technique (SMART) which is focused on analyzing and exploring the legacy components which can be potentially used for extracting as a service. It is made up of four constituent which includes planning process and tools that can be used. Second one is the Sneed Approach which also deals with illustrating how a legacy component can be reused for constructing various services. Likewise, the MASHUP (MigrAtion to Service Harmonization compUting Platform technology) technique covers both the architectural and behavioral aspects of the migration and it is done through six steps. The first is modelling the target system, second is analyzing the legacy system and infrastructure, third one is mapping the enterprises requirements with the system components and identifying the component that can be moved as service, fourth one is to design a concrete target architecture, fifth one is

defining the service level agreement and the last one being implementation and deployment of the target system.

Microservices migration strategies

Legacy systems generally are monolithic in nature which are quite tightly coupled and have grown over years to become large and complex [34]. A big monolithic system can be broken down into several lightweight services that are basically loosely coupled, and each service is entitled to serve a common business purpose. These services connect with each other to provide a common business goal. Migration to microservices architecture is not so straightforward and needs a well-planned migration process. There are several strategies discussed in [34], [35] and [36] which support a smooth migration of legacy system to a microservices architecture.

In [34] various common strategies for migrating have been presented with reference to various case studies conducted on different type of system. The most common strategy was rewriting the current application to support micro-services architecture, and some also added new functionality on top of it. Similarly, another way was using the Strangler Pattern for migration which refers to the gradual or incremental replacement of existing system by the new one. Therefore, the migration process was either greenfield or brownfield. Greenfield migration process refers to the redevelopment of application from scratch without taking in any legacy components whereas brownfield migration process refers to reusing the existing system and its components along with integrating it to the new components. Another preferred way was to simply replace them with COTS solutions. The Microservices migration principle defines several decomposition approaches which facilitated in breaking down a huge system into smaller parts which can be converted into micro services. The decomposition can be either systematic or non-systematic. The systematic decomposition includes functional decomposition, system structure decomposition and domain driven decomposition.

The article [35] coins a semi-automated technique to migrate a legacy system to a microservice using a refactoring or modernator framework. This framework processes the legacy code and convert it into a modernized code using various processes in between. The legacy code first goes through a controller filter and then is passed to a refactoring manager. The controller filter identifies to which controller the code is going and passed to the responsible refactoring manager. The refactoring manager refactors the code using different refactoring rules and reference API which will be now passed to

the microservices creator. The microservices creator takes the refactored code and organizes it to automatically generate the microservices. The microservices packager then takes the generated microservices and packs it into a compressed file such as iso or docker image. Finally, the microservices exporter transports the ready service to the user.

Similarly, in article [36] a framework which facilitates the microservices migration is discussed as well. In this framework, the migration process is broken down into three phases. The first phase is system comprehension phase where several activities are performed such as understanding legacy system, setting goals, identifying quality attributes and so on. Not only the current system, but also the architecture alternatives are accessed. The second phase concerns Strategy Definition where the application development strategy is defined along with the related processes. There might be various available strategies to choose from but the one which meets the requirements and has certain attributes in terms of quality, automation potential, availability of tools and so on is selected. The third phase is Architecture Definition where a series of activities are performed. For microservices architecture, we need to identify and group the services which can be the microservice candidate. Such candidates are identified in this phase. The implementation of such services is to be done incrementally and at a priority order. Likewise, the most suitable microservice architecture is also defined carefully. If there is any deviation with the expected system, the process needs to switch back to phase 2.

4.3.3 Strength and weakness evaluation of strategies

The third research question in this research concerns evaluating the strength and weakness of the strategies explained in the section 4.3.2. Some strategies can be suitable for some purpose but might have some flaws for other purposes. Therefore, it needs to be evaluated to give a clear picture of the strategies. The analyzed articles had highlighted how these strategies can be helpful during the migration process. The consequences or drawbacks were discussed in the conclusion part in most of the articles. Hence to have a visibility on strengths and weakness of the strategy side by side, they are evaluated in the table 4.7 below.

	Strategy Name	Strengths	Weaknesses
General Migration Strategies	Architecture driven migration strategy.	- A process driven approach for migrating a legacy system. - A round-trip process engineering which can be used as reference model to develop tools required for making the migration process automated and convenient.	- Complex process as it needs to go through several steps such as legacy to SOA or cloud migration. - Time consuming - Technical debt is not addressed during the migration process.
	Database first approach	- Incremental strategy which also enables the reusability of legacy system. - Favors systems that are disposable.	- Time consuming method also the system is not operational during migration process.
	Database last approach	- Incremental strategy suitable for decomposable system. - Reuse of legacy system components are possible.	- Consumes significant amount of time and the system is not functional during data migration
	Composite database approach	- Incremental migration of data and application removes the overhead of migrating large data. - Suitable for various type of system such as fully decomposable, semi decomposable and non-decomposable systems.	- Requires coordinator which increases complexity for maintaining data integrity across different database.
	Chicken Little Strategy	- The system is operational during migration and will be part of both target and legacy system. - Suitable for decomposable, non-decomposable and semi decomposable system	- Complex process as it implements both gateway approaches and interoperation of system is also quite complex.
	Big Bang Methodology	- System can be developed in a clean manner as it is developed from scratch.	- Takes longer time and lacks reusability of legacy systems.

	Butterfly Methodology	- Gateway free approach which removes the need for constantly accessing the legacy system	- The target system will not be available until migration is completed.
Cloud migration strategy	Cost driven migration strategy.	- Suitable for estimating the cost for migrating a legacy system with tight budget.	- Time consuming process which cannot be applied in case of a tight schedule.
	Reengineering strategy	- Simple solution which uses both forward and reverse engineering methods for migrating a legacy system.	- The extracted artifacts are not adjusted which creates complexity during maintenance.
	Model driven strategy	- It uses model at the core of development which increases reusability as well as error free migration	- Generating model can take significant time and hence is time consuming.
	Legacy to cloud migration horseshoe framework.	- A planned Architecture driven framework which provides a complete solution to migrate a legacy system to cloud.	- The cloud architecture is formed with reference to the vendor's architecture which makes it difficult to switch the vendor later.
	Cloud Migration Approach (CloudMIG)	- A semiautomatic approach which facilitates cloud migration by analysing the various aspects of cloud.	- Migration of legacy system with large data is time consuming and resource intensive.
	Cloud-Reference Migration Model	- Incremental approach of cloud migration which minimizes the required effort.	- It is complex as it has various stages with several tasks that needs to be executed during migration
	SMART	- It encourages component reusability along with the feasibility study which is beneficial in various ways.	- It only concerns the conceptual level rather than the implementation level which questions its significance.
	REMICS	- This approach enables interoperability of cloud services and supports reusability.	- Complex as it needs to establish a smooth connection between the legacy and cloud components.
	SMiLe2Cloud Strategy	- Secured migration of system is ensured as the migration will be security driven.	- Significant amount of resource is required in terms of resource as well as personnel.

SOA migration strategy	Roadmap for SOA migration.	- A good roadmap plan which covers the various aspects of migration of Legacy System to a SOA architecture.	- It is a long lead process which is not suitable for projects with tight schedule.
	Service Migration and Reuse Technique (SMART)	- The aspects such as costs, risks are considered along with feasibility study and component reusability.	- It is a resource intensive process which needs skilled personnel and appropriate tools.
	Sneed Approach	- A suitable framework that provides reusability and equipped with top-down and bottom top approach for extracting the service.	- Complex as it requires various steps such as understanding of existing system, designing new services and ensuring their smooth execution.
	MASHUP	- The roadmap provided by this approach is structured which can be used for legacy system migration and ensures the integration across all the layers.	- Establishing harmony between the services extracted from Legacy systems having deep dependency is complex.
Microservices migration strategies	Strangler Pattern	- Incremental migration method which supports addition of new service alongside refactoring the old ones.	- Refactoring can lead to adapter hell leading to complexity, maintenance overhead and issues.
	Modernator Framework	- A thorough framework which explains all the steps to extract micro services from existing legacy code.	- Complex as various things should be handled such as refactoring rules and so on.
	Architecture-centric migration framework.	- A good road map that guides a legacy system to migrate to micro services architecture by understanding legacy system.	- Prone to error or deviation in different phase that might need stepping back to previous phase.

Table 4.7: Strength and Weakness evaluation of strategies.

In the above table, each collected migration strategies in section 4.3.2 are placed in the first column. Then the advantages and disadvantages with each strategy are listed in the adjacent columns. It shows that none of the strategies for migration are perfect as they also bring some challenges with them. The strengths of the strategies can be seen if they support incremental approach, reusability, scalability, plan or framework use.

Likewise, the weaknesses can be seen in terms of high complexity, time consumption, error prone and so on. However, a strategy can be selected in terms of strengths it has. Nevertheless, the weaknesses which comes with it also should be kept in mind. It helps to formulate corrective action when they occur. This increases the chance of making the migration process successful.

5. DISCUSSION

5.1 Discussion on data extraction results

The main aim of the data extraction is to answer the stated research questions in the context of case organization. The extracted results were collected with the help of the 18 literature which made up to the final selection. All articles were carefully examined to explore and extract the information that were used for answering questions. The first question concerns about the factors that need to be considered for migrating a legacy system to a new environment, second question was for exploring and extracting various types of migration strategies that were discussed in the literatures and the third question was the comparison of various strategies based on their strengths and weaknesses.

There are several things that need to be considered while migrating a legacy system. The major one found during the research include cost, time, effort, scalability, architecture, risk, size and so on. Among these all, cost is found to be the most discussed one in the literatures. The cost is accessed in the terms of the existing cost for operating and maintaining a legacy system against the cost that is required for migrating the system to a modern architecture or technology. The migration is considered only if the cost of migration is lower and feasible. Likewise, other factors such as time and effort also play an important role. The time and effort estimation should be among the preliminary activities that needs to be done before the migration. Risk analysis is also among the discussed factors that needs to be evaluated. There can be various risk involved which affects the decision for migrating a system.

The migration of legacy systems to a modern environment is possible through various strategies which were found during this research. There are now various architectural options to which a legacy system can be migrated. The process of migrating a legacy system can be done through various strategies which consists of different steps. All the articles suggested various strategies which were made on top of the core migration strategies. Most of the strategies were defined in terms of framework or roadmaps which consist of several similar steps.

The commonly preferred target architecture for a legacy system includes cloud, service-oriented architecture and microservices. The concept with service-oriented architecture and microservices is quite similar as the SOA has a broader scope whereas

microservices are more granular and modular. Among these three, migration to cloud architecture was the most researched topic where various research was done. All this research suggested that the scalability, agility, cost efficiency etc. were the major contributor that made the demand for cloud architecture quite high, and hence more researches were done.

Similarly, by going through all the articles it was found that the general migration strategies which are mentioned in the chapter 4.3.2 lay in the gist of all other migration strategies. These general migration strategies were used in all the roadmaps or suggested frameworks at some point. Only differences found was during the architecture modelling. Among all, those strategies which used the incremental migration approach had a significant advantage as they facilitated migration of system along with interoperability with the target system.

The cloud migration strategies were different in terms of their name, but the core processes were somehow like each other. The primary aim of all these strategies is to leverage the benefits of cloud services such as auto scaling, pricing model, rapid development, CI/CD and so on. The cloud migration provided an easy migration procedure which took less time and is also highly scalable. It also has several options to choose upon namely public, private, community and hybrid cloud. Nevertheless, there were many cloud-related constraints which include things like bandwidth, security, vendor lock in, inflexibility and so on. Cloud architecture is a good solution for migrating a general application using any one of the strategies, but it is not preferred for migrating a safety critical system as it can be highly affected by external factors.

Likewise, SOA is also one of the common types of architecture which helps to break down a big legacy system into smaller units to enable loose coupling along with high maintainability. The major advantage of SOA is its reusability where the components of the existing legacy systems can be broken down into several services. These services will serve for the common business goals by replacing a monolithic legacy system. It is also one of the most preferred migration options before cloud architecture was developed. The relation between legacy systems and SOA is quite close. SOA supports reusability that allows to expose some functions from legacy systems as service interfaces. It can be utilized by that software which operates within the same domain but having different functionality. The strategies for migrating to a SOA were also quite complementary to each other which mainly concerned with analyzing the legacy system and extracting various services out of it. The method for extracting the services were

different in different articles. The problem with this architecture can be that it will introduce a significant organizational change and requires governing mechanisms.

Finally, the microservices are small and loosely coupled distributed services which can exist independently and perform a specific business function. These types of systems can be developed, deployed, and scaled independently. These distributed services require proper communication mechanism to know about the state of other systems belonging to a same domain. The strategies suggested in the articles regarding microservices architecture were quite like SOA. The legacy system is converted to a microservice by analyzing the legacy system and extracting the possible distributed system out of it which could operate independently. Like other strategies, this strategy also followed the same core process although having a different name in the article.

5.2 Conclusion of extracted results

The problem with the legacy system is well known to all the organizations operating it. The major contributor to this problem is the fast-paced technological advancements. The legacy systems are facing various consequences and are struggling to deal with the modern requirements. These systems are still found as they are responsible for core business functions of any organization.

Migration of a legacy system is a procedural activity which goes through various stages. There are various considerations done before moving forward with migration process as shown in table 4.3. A migration process is highly affected by selection of the target architecture. There are some basic strategies of migration which are utilized by various strategies which concern a specific architecture. We should always consider the things such as cost, time, risks and so on before migrating any system. If such things are not analyzed beforehand, then the whole migration process can fail. The selection of migration strategy largely depends on the type of the system.

The straightforward solution of migrating any system would be either to utilize the existing system through techniques such as conversion, reimplementation or wrapping or to start from scratch using the techniques such as redevelopment or replacement. These techniques are then utilized by other migration strategies that are specific to an architecture. These strategies can be considered the core strategies.

Cloud migration is the way to go for most of the systems as we can utilize various benefits such as scalability, cost effective and efficient. An organization is not required to maintain an expensive hardware and resource but can subscribe a cloud provider where they can easily deploy their service. However, cloud system goes through various cloud related constraints such as latency, security, strict monitoring and so on. Therefore, cloud architecture would not be favorable for safety critical systems.

Likewise, the SOA migration is another close substitute of cloud architecture which has a high support for existing legacy system components. This helps to make a legacy system divided into various services making it loosely coupled and manageable. These services will require a proper orchestration to function properly. Businesses can migrate their legacy system to SOA without many hassles as it can represent various business units in terms of service making the division easier. The drawback with SOA includes high initial investment, challenges in managing the service and so on. This architecture should not be chosen in case of tight budget as well as for the system not having multiple functionalities.

Microservices architecture is on the other hand another migration alternative for legacy systems which requires further breaking down of the legacy system into smaller, independent, and loosely coupled services. It will make the legacy system more manageable and resilient as each smaller service can develop, deploy, and scale automatically. A whole legacy system can be either decomposed into smaller services based on function or can be replaced by any COTS application.

5.3 Future research

During the research process, various research gaps were identified which can be considered as good candidates for future research. Likewise, this research was not able to address some of the aspects due to time constraints. Further research in these aspects will be more beneficial in terms of legacy system migration.

To begin with, the research gap can be seen in case of handling the technical debt during the process of migration. Most of the above-mentioned strategies or articles did not suggest any solution to reduce the technical debt. Technical debts can be removed from technique such as architectural refactoring, but they were not mentioned explicitly.

There is no proper mapping of a migration strategy to a type of system. For example, the options for migrating a legacy desktop-based application to a modern environment. Likewise, the strategies needed for migrating various safety critical systems have not been well discussed and well defined.

5.4 Evaluation of research

The research is conducted with the help of Systematic Literature Review which establish to be very effective to tackle the research questions. Among various other methods such as experimentation, case study, SLR proved to be an unbiased way of conducting any research. For example, in case of experimentation, the findings need to be demonstrated using a specific technology or within a given time. Therefore, there is a high chance of research being biased towards the use of a specific technology.

The research question also went through some validations and were shaped according to the CIMO scheme which made them more concrete. It helped to map the research questions closely with their coverage on the research results. Similarly, SLR enabled this research to explore various strategies, technologies, methods etc. by going through numerous amounts of literatures.

The total number of literatures found were quite large, 607. The search operation was conducted with the help of search strategy presented in section 3.2. The high search number were reduced to 10 using the inclusion and exclusion criteria mentioned in section 3.4. It helped to select only those literatures that are the most relevant to this research. The reduction rate is quite large as compared to the articles found showed that the inclusion of criteria was set quite strictly. Nevertheless, the collected results with the help of these articles were adequate to solve the problems identified by the research.

Finally, SLR has well defined steps that can be followed to conduct full-fledged research. Some of the steps mentioned were quite complex such as literature quality assessment, defining the inclusion and exclusion criteria. In some of the cases, the unbiasedness of this method can cause difficulty in defining the scope of the research making the research process cumbersome.

6. RESEARCH SUMMARY

Systematic literature review is the method chosen for this research. The main objective of this research is to provide a strategic guideline for the case organization to migrate their existing legacy system to a newer environment. There are various types of legacy systems that exist in the organization which differs in terms of architecture, size, technology and so on. To cover most of the legacy systems, various strategies based on different target architecture were analyzed. Therefore, this research is targeted to provide an evaluation to various strategies that are required to migrate a legacy system.

The first research question concerns about the consideration that needs to be done for migrating a legacy system. The major factors include cost, time, effort, risks and so on. All these factors should be considered carefully by the case organization for migrating their legacy system. Cost of migration should be lower than running and maintaining the legacy system. Likewise, the time and effort for migration needs to be estimated. Majority of the legacy system must be going through an active development and delivery. For such systems, these considerations are even more critical.

The second research question explores the various migration strategies that are available in the reviewed literatures. The strategies extracted are grouped into four types. The strategies that belong to the first type are general in nature which could be used irrespective to the target architecture. The later types are grouped based on target architecture namely cloud migration strategy, SOA migration strategy and microservices migration strategies. The general strategies can be further utilized by other strategies. Among all the strategies, those strategies which are incremental in nature, reuses legacy systems, scalable, loosely coupled etc. are the ones having the upper hand.

In case of the organization, the nature and technology used in the legacy systems are the major driver for selecting a particular target architecture and the suitable corresponding migration strategy. In case of monolithic desktop-based application, they can be broken down into various services or can be migrated to a service-oriented architecture by using the strategies such as SMART can be used. Likewise, the legacy web-based system can utilize the advantages of cloud computing. Bigger systems can be broken down into various microservices. For microservices, various technologies such as containerization can be considered.

The last research question, the strengths and weaknesses of these strategies are evaluated. All the strategies are not so straight forward as they need to go through various stages or phases making them quite complex. The overall strengths with the general migration strategy include incremental, reusability, forward and backward compatibility whereas the weaknesses include complex, time consuming, non-functional system during migration process. Cloud migration strategies have advantages such as cost effectiveness, scalability, reusability, plan based, incremental, interoperability and have disadvantages such as complex, vendor lock in, time consuming, resource consumption.

SOA migration strategies are done through a good roadmap or plan. The pros with SOA migration strategies are incremental, loose coupled service formation, good integration between services whereas the cons include complex and time consuming, resource intensive. The last one is microservices migration strategies. They also have advantages like SOA migration strategies, but they are even more loosely coupled, more scalable and have a well-defined service decomposition. They also have capability of independent deployment. The disadvantages with microservice migration strategies includes similar factors such as: complexity, adapter hell, maintenance overhead and so on.

In conclusion, the case organization can choose among various strategies that have been listed, explained, and evaluated within this research. The legacy systems within the organization should be migrated as soon as possible. With the passage of time, the products or systems will grow and become more complex. More complexity in the legacy system will result in more complex migration processes. Therefore, the case organization should also focus on modernizing their legacy systems along with the ongoing software development cycle.

REFERENCES

[1] Seacord, R. C., Plakosh, D., & Lewis, G. A. (2003). Modernizing legacy systems: software technologies, engineering process and business practices. In *Addison-Wesley Longman Publishing Co., Inc. eBooks*.

[2] Vijaya, A., & Neelanarayanan, V. (2018). Modernizing legacy systems. International Journal of Web Portals, 10(2), 50–60.

[3] Heuvel, Willem-Jan van den. *Aligning Modern Business Processes and Legacy Systems : A Component-Based Perspective*, MIT Press, 2007. *ProQuest Ebook Central*,http://ebookcentral.proquest.com/lib/tampere/detail.action?docID=3338554.

[4] Sneed, H. M., & Verhoef, C. (2020). Cost-driven software migration: An experience report. *Journal of Software*. https://doi.org/10.1002/smr.2236

[5] Annett, R. (2019). Working with Legacy Systems: A practical guide to looking after and maintaining the systems we inherit. Packt Publishing

[6] Chondamrongkul, N., Sun, J., & Warren, I. (2021). Software architectural migration: An automated planning approach. *ACM Transactions on Software Engineering and Methodology*, *30*(4), 1–35. https://doi.org/10.1145/3461011

[7] Gholami. (n.d.). In *Mahdi Fahmideh; Daneshgar, Farhad; Beydoun, Ghassan; Rabhi, Fethi*. Challenges in migrating legacy software systems to the cloud - an empirical study

[8] Rinta-Kahila, Tapani, Penttinen, Esko, Lyytinen. (2023) Getting Trapped in Technical Debt: Sociotechnical Analysis of a Legacy System's Replacement. *Kalle MIS Quarterly*, *47*(1), 1–32.

[9] Bisbal, J., Lawless, D., Wu, B., & Grimson, J. (1999). Legacy information systems: issues and directions. *IEEE Software*, *16*(5), 103–111. https://doi.org/10.1109/52.795108

[10] Althani, B., & Khaddaj, S. (2017). Systematic review of legacy system migration. 2017 16th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES).

[11] Y. Alkazemi, B., & Department of Computer Science, Umm Al-Qura University, Makkah, Saudi Arabia. (2014). A framework to assess legacy software systems. *Journal of Software*, *9*(1), 111–115. https://doi.org/10.4304/jsw.9.1.111-115

[12] Implementation phases in modernisation of legacy systems. (2019, December 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9073628

[13] Khadka, R., Batlajery, B. V., Saeidi, A. M., Jansen, S., & Hage, J. (2014). How do professionals perceive legacy systems and software modernization? *Proceedings of the 36th International Conference on Software Engineering*.

[14] Agilar, E., Almeida, R., & Canedo, E. (2016). A systematic mapping study on legacy system modernization. *Proceedings of the 28th International Conference on Software Engineering and Knowledge Engineering*.

[15] Ganesan, A. S., & Chithralekha, T. (2016). A survey on survey of migration of legacy systems. *Proceedings of the International Conference on Informatics and Analytics*.

[16] Khadka, R., Saeidi, A., Idu, A., Hage, J., & Jansen, S. (2012). Legacy to SOA evolution: A systematic literature review. In *Migrating Legacy Applications* (pp. 40–70). IGI Global.

[17] Kitchenham, B. and Charters, S., 2007. Guidelines for performing systematic literature reviews in software engineering. Keele University and Durham University Joint Report

[18] Rochimah, S., & Sheku, A. (2016). Migration of existing or legacy software systems into web service-based architectures (reengineering process): A systematic literature review. *International Journal of Computer Applications*, *133*(3), 43–54. https://doi.org/10.5120/ijca2016907801

[19] Briner, R. B., & Denyer, D. (2012). Systematic review and evidence synthesis as a practice and scholarship tool. In *The Oxford Handbook of Evidence-Based Management* (pp. 112–129). Oxford University Press.

[20] V. Lenarduzzi, F. Lomio, N. Saarimäki et al., Does migrating a monolithic system to microservices decrease the technical debt?. The Journal of Systems & Software (2020) 110710, https://doi.org/10.1016/j.jss.2020.110710

[21] Aslam, M., Rahim, L. A. B., Watada, J., Rubab, S., Khan, M. A., AlQahtani, S. A., & Gadekallu, T. R. (2023). Cloud migration framework clustering method for social decision support in modernizing the legacy system. *Transactions on Emerging Telecommunications Technologies*. https://doi.org/10.1002/ett.4863

[22] Müller, H. A. (1997). Reverse engineering strategies for software migration (tutorial). *Proceedings of the 19th International Conference on Software Engineering - ICSE '97*.

[23] Ahmad, A., & Babar, M. A. (2014). A framework for architecture-driven migration of legacy systems to cloud-enabled software. *Proceedings of the WICSA 2014 Companion Volume*.

[24] Frey, F. J., Hentrich, C., & Zdun, U. (2014). Pattern-based process for a legacy to SOA modernization roadmap. *Proceedings of the 19th European Conference on Pattern Languages of Programs*.

[25] Jain, S., & Chana, I. (2015). Modernization of Legacy Systems: A Generalised Roadmap. Proceedings of the Sixth International Conference on Computer and Communication Technology 2015.

[26] Strobl, S., Bernhart, M., & Grechenig, T. (2020). Towards a topology for legacy system migration. *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*.

[27] Jamshidi, P., Ahmad, A., & Pahl, C. (2013). Cloud migration research: A systematic review. *IEEE Transactions on Cloud Computing*, *1*(2), 142–157. https://doi.org/10.1109/tcc.2013.10

[28] Wagner, C. (2014). *Model-driven software migration: A methodology; Reengineering, recovery and modernization of legacy systems*. Not Avail. https://doi.org/10.1007/978-3-658-05270-6

[29] Sneed, H., & Verhoef, C. (2019). Re-implementing a legacy system. *The Journal of Systems and Software*, *155*, 162–184. https://doi.org/10.1016/j.jss.2019.05.012

[30] Khachouch, M. K., Korchi, A., & Lakhrissi, Y. Architecture Driven Modernization: A Review on Reverse Engineering Techniques based on Models. (n.d.). *Architecture Driven Modernization: A Review on Reverse Engineering Techniques Based on Models' Approach*

[31] Hasan, M. H., Osman, M. H., Admodisastro, N., & Muhammad, S. (2023). Legacy systems to cloud migration: A review from the architectural perspective. *Journal of Systems and Software*, *202*, 111702. https://doi.org/10.1016/j.jss.2023.111702

[32] Márquez, L. G. G., Rosado, D. G., Mouratidis, H., Mellado, D., & Fernández-Medina, E. (2015). A framework for secure migration processes of legacy systems to the cloud. In *Lecture notes in business information processing* (pp. 507–517). https://doi.org/10.1007/978-3-319-19243-7_46

[33] Abdellatif, M., Shatnawi, A., Mili, H., Moha, N., Boussaidi, G. E., Hecht, G., Privat, J., & Guéhéneuc, Y. (2021). A taxonomy of service identification approaches for legacy software systems modernization. *Journal of Systems and Software*, *173*, 110868. https://doi.org/10.1016/j.jss.2020.110868

[34] Fritzsch, J., Bogner, J., Wagner, S., & Zimmermann, A. (2019). Microservices Migration in Industry: Intentions, Strategies, and Challenges. *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*.

[35] Osman, M. H., Saadbouh, C., Sharif, K. Y., & Admodisastro, N. (2022). From Monolith to Microservices: A Semi-Automated Approach for Legacy to Modern Architecture Transition using Static Analysis. *International Journal of Advanced Computer Science and Applications*, *13*(10). https://doi.org/10.14569/ijacsa.2022.01310107

[36] Fritzsch, J., Bogner, J., Haug, M., Wagner, S. J., & Zimmermann, A. (2023). Towards an Architecture-Centric methodology for migrating to microservices. In *Lecture notes in business information processing* (pp. 39–47). https://doi.org/10.1007/978-3-031-48550-3_5

[37] Lehman, M. M., & Ramil, J. F. (2003). Software evolution—Background, theory, practice. *Information Processing Letters*, *88*(1–2), 33–44. https://doi.org/10.1016/s0020-0190(03)00382-x