Re-engineering strategies for legacy software systems

Ronnie Brandtberg

Helsinki June 6, 2020

Master's thesis

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Ronnie Brandtberg

Työn nimi — Arbetets titel — Title

Re-engineering strategies for legacy software systems

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's thesis	June 6, 2020	55 pages + 25 appendices

Tiivistelmä — Referat — Abstract

Re-engineering can be described as a process for updating an existing system in order to meet new requirements. Restructuring and refactoring are activities that can be performed as a part of the re-engineering process. Supporting new requirements like migrating to new frameworks, new environments and architectural styles is essential for preservation of quality attributes like maintainability and evolvability. Many larger legacy systems slowly deteriorate over time in quality and adding new functionality becomes increasingly difficult and costly as technical debt accumulates. To modernize a legacy system and improve the cost effectiveness of implementing new features a re-engineering process is often needed. The alternative is to develop a completely new system but this can often lead to loss of years of accumulated functionality and be too expensive.

Re-engineering strategies can be specialized and solve specific needs like cloud migration or be more generic in nature supporting several kinds of needs. Different approaches are suitable for different kinds of source and target systems. The choice of a re-engineering strategy is also influenced by organisational and business factors. The re-engineering of a highly tailored legacy system in a small organisation is different from re-engineering a scalable system in a large organisation. Generic and flexible solutions are well suited for especially smaller organisations with complex systems.

The re-engineering strategy Renaissance was applied in a case study at Roima Intelligence Oy in order to find out if such a strategy is realistically usable, useful and valuable for a smaller organization. The results show that a re-engineering strategy is possible to be used with low overhead in order to prioritize different parts of the system and determining a suitable modernization plan. Renaissance was also shown to add value especially in the form of deeper understanding of the system and a structured way to evaluate different options for modernization. This is achieved through assessing the system from different views taking into account especially business and technical aspects. A lesson learned about Renaissance is that determining an optimal scope for the system assessment is challenging. The results are applicable for other organisations dealing with complex legacy systems with constrained resources.

Limitations of the study are that the number of different kinds of re-engineering strategies discussed is small and more suitable strategies than Renaissance could be discovered with a systematic mapping study. The amount of experts participating in the process itself as well as the evaluation was also low, introducing some uncertainty to the validity of the results.

Further research is needed in order to determine how specialized and generic re-engineering strategies compare in terms of needed resources and added value.

Avainsanat — Nyckelord — Keywords
software engineering, re-engineering, reverse engineering, refactoring, modernization, legacy

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Introduction

Software development is expensive, and the larger and more complex a system the more investments it requires. Over time systems grow and accumulate more and more functionality, making their replacement even more expensive and risky than further development. This means that several years old and constantly aging software systems, or parts of systems, are not uncommon. The issue with legacy systems is not a new one and already in 1995 Bennet [1] was concerned that there will soon be software over 40-years old still in operation. One can only wonder what the situation is now more than twenty years later. As a large part of software development involves maintaining, updating, replacing or integrating these old complex systems, we need well defined processes for managing the evolution of legacy systems.

Legacy systems can be defined as existing systems which are well established, contain vital core functionality and business logic and are viewed to be unfit for future needs of the organisation [2]. Legacy systems are according to many software professionals generally well established, feature full and have many users [3]. The problem with legacy systems is that they can deteriorate in terms of maintainability and architectural soundness over time. This is amplified as more and more functionality is added over the years, and non optimal changes are made to accommodate new requirements, that have not originally been planned, or that the existing technology stack does not fully support.

Eventually an existing system becomes a legacy system in the sense that it is seen as unfit for future needs due to low maintainability, evolvability and disappearing knowledge. The degradation of systems over time is a common problem in software development and is mainly referred to as technical debt [4]. Essentially “not quite right code” is accumulated over time and the cost of fixing the code and paying of the debt grows in interest and becomes more and more expensive. In order to cater to future needs and pay off the technical debt, a legacy system has to be replaced or modernized (re-engineered). A complete replacement of an extensive business critical system can be near impossible due to very high costs and risks. Risks include not being able to preserve all accumulated and hidden functionality where undocumented and implicit logic can exist. Hidden functionality can for example be an undocumented hard coded value that affects a function call’s result with specific inputs. To mitigate costs and risks of a complete replacement, modernization or re-engineering in one way or another is often the only realistic option to remedy issues present in legacy systems.

Re-engineering can be defined as a process where an existing system is updated and changed to meet new requirements. Re-engineering usually includes reverse-engineering the source system and forward engineering new functionality to a target system [5]. The term modernization can be used interchangeably with re-engineering, in the context of legacy systems, which this paper mainly refers to. The new requirements that demand re-engineering are usually large in scope and include, e.g., changing frameworks, changing runtime platforms, meeting new quality attributes like evolvability or even supporting completely new development processes like DevOps [6].

Re-engineering should not be confused with other commonly used terms "restructuring" and "refactoring". Chikofsky and Cross try to alleviate confusion over terminology and have defined several key terms [5]. Restructuring involves the transformation of a representation form to another at the same relative abstraction level without modifications from new requirements [5]. Another closely related term "refactoring" is defined as improving the internal structure of a system without altering the external behavior in an object oriented environment [7]. In essence the starting point for re-engineering is new requirements and the need for new functionality, which then triggers the need for changes that can be addressed by restructuring and refactoring parts of the existing system.

Different approaches or re-engineering strategies are suitable for different kinds of source systems. The choice of a re-engineering strategy is also influenced by organisational and business factors. The re-engineering of a highly tailored legacy system in a small organisation is clearly different from re-engineering a scalable system in a large organisation. This study aims to answer specifically what kind of processes and criteria can be used to choose an appropriate re-engineering strategy for small and medium enterprises that develop complex web applications. Roima intelligence is an enterprise that fits into this category and different re-engineering strategies are evaluated in the context of a a case study conducted in the company.

The remainder of this study is organised as follows. In chapter 2 background information of the re-engineering landscape is provided. Chapter 3 describes the research approach and Chapter 4 presents results of the literature review. Chapter 5 focuses on the Roima case study using the literature review as a basis. Finally Chapter 6 discusses the results of the case study.

2 Background

The choice of a suitable re-engineering strategy is not simple as different factors contribute in various ways. First of all organisational and business factors set several constraints in terms of available resources and know how. Secondly the architecture, complexity and technical quality of the legacy system undergoing re-engineering affects the choice of a suitable strategy. Finally, the starting point that initiates the need for re-engineering, and what main goal the re-engineering process should achieve, obviously affects what kind of strategy is suitable. This section describes general triggers for re-engineering, what risks and benefits re-engineering has, what kind of different types of re-engineering strategies exist and what possible pros and cons they have.

2.1 Triggers for re-engineering

The need for legacy system re-engineering can arise for several different reasons. Firstly, companies can "hit roadblocks" as the old system is not compatible with new requirements and therefore limits innovation and growth [8]. Khadka et al. [3] empirically investigated the general perception of re-engineering by software professionals with a mapping study and interviews. The findings of Kahdka et al. in terms of how legacy systems are perceived in the industry is mostly in line with academia as most people stated that a legacy system is old, core to the business and unfit for future requirements in terms of functionality and business strategy. Most practitioners stated that legacy systems are rigid and inflexible. Changes that require flexibility are varied and include organisational changes, mergers and acquisitions, faster time to market and new requirements stemming from these changes. In practice this can often materialize in a need for scaling and integrating with other systems. For example a legacy system might not support a common integration interface REST [9] that is used in most modern service oriented systems.

Another important reason for starting the re-engineering process is that maintenance costs of legacy systems are too high. More than half of the interviewed practitioners in the study on different views of re-engineering also accentuated that high maintenance cost is an important reason for modernization [3]. The importance of cost reduction is further underlined by the estimation that as much as a fifty percent decrease in maintenance costs can be gained by re-engineering [8].

A further reason for re-engineering is that it can become difficult to recruit new personnel if the legacy system consists of old programming languages and architectures. In fact more than 90% of the interviewed practitioners in Khadka et al's study also highlighted the lack of knowledge as a key driver for modernization [3]. In practice this means that the unavailability of documentation and experts cause a lack in resources available for legacy system maintenance and development. This category can also include the lack of suppliers or vendors in terms of, e.g., third party components not receiving updates.

Legacy systems can also have the problem of not being compliant and regulated according to newer standards. For example issues of compliance in personal data handling and storing have arisen for many software providers due to EU-regulations like GDPR (General Data Protection Regulation) [10]. These issues can be insurmountable for older legacy systems and might require extensive re-engineering to fix.

The different triggers for re-engineering are not mutually exclusive and a legacy system can often have many different triggers that together start the re-engineering process.

2.2 Risks of re-engineering

The main reasons for using legacy systems are that they often contain business critical core functionality of organisations. The main hurdles for modernisation are high financial costs of re-engineering, risks of the modernisation not succeeding and the availability of skilled staff. The problem is that a legacy system becomes more and more unfit for future needs over time, as it may have problems utilizing the latest hardware such as multi core or simply being incompatible with the latest software architectures. As of now the most prominent software architectures include service oriented systems and micro-services, which try to meet the ever increasing requirements of customers. An incompatible legacy system might lead to missed business opportunities especially in the future and severely hinder growth even though the legacy system works well for current systems and customer requirements.

The largest perceived challenge is, according to many practitioners, time constraints to finish modernization which often related to a scarcity of resources [3]. Another common issue that increases the time of re-engineering is ongoing changes in requirements during the process. The second largest perceived issue, amongst industry experts, is predicting return on investment of a modernization process [3]. The third largest identified issue is data migration.

The perceived challenges of legacy system modernization in Khadka et al's study varied more from academia than the triggers [3]. Traditionally re-engineering has focused much on the technical aspects of a system but Khadka et al. suggest that industry practitioners actually consider different business aspects most important in contrast to much of academic research.

2.3 Different types of re-engineering strategies

Many different kind of re-engineering strategies exist and a systematic literature review by Althani and Khaddaj [8] gives a broad overview of the current re-engineering landscape. The authors discovered strategies that range from simple to comprehensive in scope. The smaller strategies consist of simply wrapping old legacy systems while the most comprehensive ones are full migration strategies.

There is no "one fit all" solution for re-engineering legacy systems and every case has to be evaluated from different perspectives taking into account time constraints, risks (including lost functionality and availability during the process), financial costs and other business needs. Some legacy systems need a full migration while others may only require a partial migration or a simple wrapper with new interfaces. Re-engineering strategies can be divided into four categories; complete, incremental, partial and wrapping[8].

A complete migration is the complete redevelopment from scratch where all functionality, interfaces and data is brought to a new platfrom or architecture using modern technologies. The definition of a completed migration is when the old legacy system can be switched off and the new system becomes operational. One example of this strategy is the so-called "Big Bang" (or "Cold Turkey") approach [11], where the old system is terminated at the same time as the new replacing system is taken into use.

An incremental re-engineering strategy aims to keep the old legacy system running while smaller parts are modified and re-engineered to meet new requirements. One example is the Renaissance method which takes into account different technical and organisational requirements. Another example is the architecture-driven modernisation method (ADM) which also transforms the legacy system incrementally to the target system but instead "focuses on the interoperability concept between domains" [12] in order to validate that the re-engineered result is successful.

The third category is partial migration where the whole legacy system is not re-engineered, and instead only a part is modernised to support updated requirements. This can for example consist of a modernisation effort where some specific components are migrated to a cloud environment, and the legacy system is updated with a wrapper with new interfaces to communicate with the re-engineered cloud components.

Wrapping is simply encapsulating the existing system without changing it at all, and providing new interfaces through a wrapper. This is obviously a more short term solution as the underlying legacy system remains the same.

2.4 Pros and cons of varying strategies

Each different re-engineering strategy has its own pros and cons and it really comes down to the specifics of the source and target systems when determining which strategy is most suitable. The most important factors in evaluating which strategy should be chosen are business value and the risks of an failed or incomplete re-engineering effort [13]. Very little previous work has been broad and generic enough to be suitable for most legacy systems, as many stragegies are bound to specific programming languages or architectures. An aspect that is found lacking in previous strategies according to the authors is the proper evaluation of risks and costs.

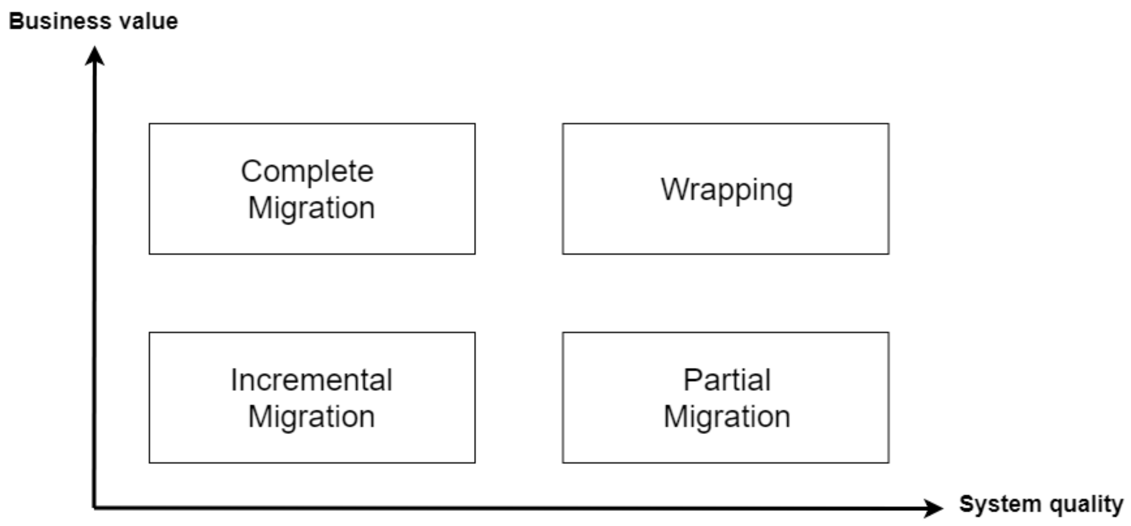


Figure 1: Re-engineering strategy comparison [8]

As presented in Figure 1, different types of re-engineering strategies can be simplistically evaluated on a scale of business value and system quality of the source system. Business value can be estimated in different ways but ultimately boils down to monetary value. System quality can also be estimated in many different ways including, e.g, quality attributes like availability, reliability and usability. The figure can be interpreted as incremental being good if the business value is low and does not support new requirements while also having low quality in general like for example performance. A partial strategy is good if the business value is low but the quality in general is good. Then the legacy system can support the new part that is re-engineered without becoming a bottle neck for, e.g., performance. A complete migration is good if the current business value is high but the underlying system is of low quality. The complete migration ensures that the existing functionality is transferred while improving quality factors like performance and maintainability. Wrapping can be a good strategy if both the business value and the underlying quality of the legacy system is high and the main need is to integrate the old system to a newer system.

A combination of the mentioned strategies can also be used, and is according to Althani and Khaddaj often the most realistic path for especially larger heterogeneous systems with varying characteristics between different components [8].

2.5 Modern re-engineering strategies

Khadka et al. estimate that 180-200 billion lines of legacy code are still in active use (as of 2014) [3]. This underlines the importance of re-engineering and motivates why new strategies to handle modernization are constantly being developed and studied.

It is commonly understood that software development technologies, architectures and requirements constantly change which means that many different aspects have to be accounted for in a re-engineering process regarding costs, benefits, risks, target frameworks, languages and architectures. However the common themes for the needs of re-engineering remain constant and include scalability, evolvability, performance and maintainability.

A mixed opinion was observed when industry experts were questioned about the relevance of programming language in re-engineering as approximately 50% thought that it was a deciding factor [3]. This is also somewhat reflected in academia as many studies on re-engineering focus on specific programming languages and architectures. In the past many re-engineering strategies were targeted especially for COBOL and more recently many strategies like Cloudify[14] focus on specifically cloud architecture migration. Althani and Khaddaj claim that a component and service oriented architecture is probably the safest route for migrating to modern technology [8]. While these specialised re-engineering strategies can be useful, they can quickly become irrelevant as even newer technologies emerge. This is why it is essential to understand the broader generic principles and best practices of re-engineering. This study will therefore focus on finding and analysing re-engineering strategies that are more generic and customizable in nature.

3 Research approach

The goals of this thesis are to map, describe and evaluate varying re-engineering strategies suitable for small organisations dealing with complex legaccy software systems. This study is based on the design-science research guidelines presented by Hevner et al. [15] and consists of a theoretical and empirical part. The main methods for this study are to conduct a literature review of re-engineering strategies, and then extracting a suitable re-engineering strategy to be applied in the empirical case study at Roima. The suitability is based on a comparison and analysis of the results of the literature review, while also taking into account opinions of software experts at Roima. The general focus is to find and test a generic and customizable re-engineering strategy that has the potential to be applied across organisations with similar characteristics to Roima. Testing the re-engineering strategy through the case study will help to evaluate how resource heavy, how adaptable and how useful the strategy really is in practice. The scope of the case study is limited to testing the re-engineering strategy itself focusing especially on planning the modernization, and will not include the practical implementation of the modernization plan.

The main research questions of this study are presented below:

- RQ1: What re-engineering strategies exist that are easy to understand and applicable for many different kinds of systems?
- RQ2: What specific re-engineering strategy can be applied at a small organisation dealing with complex legacy systems?
- RQ3: How can a complex web application be re-engineered in order to support a new framework?

- RQ4: Is the re-engineering strategy useful for solving different issues and providing added value?
- RQ5: How easy to use is the re-engineering strategy and what resources are needed?

3.1 Methods and materials

Design-science guidelines include seven different principles [15]. Principles focusing on the most relevant guidelines for this study are; "design as a search process", "problem relevance" and "design evaluation". This means that the study will in practice search for an existing artifact that relates to the specified research questions RQ1 and RQ2. The artifact is a re-engineering strategy and it will be chosen by conducting a literature review and comparing different re-engineering strategies in order to find a suitable one.

The evaluation of the artifact aims to follow both an experimental and descriptive design evaluation method [15]. The instantiation of the artifact is studied in a controlled environment for different qualities like usability, ease of use and added value. The artifact is applied in a real-life scenario at Roima answering RQ3. A descriptive design evaluation method uses information from the existing knowledge base to evaluate the artifacts utility. This descriptive assessment phase consists of interviews with a group of software development professionals at Roima and answers RQ4 and RQ5.

Measuring the benefits and results of re-engineering is difficult and different measures can be time consuming to produce [8]. As the need for re-engineering most often stems from new requirements that are unsupported [3] the result becomes binary as the re-engineering either succeeds in supporting the new requirements or not. The evaluation criteria will therefore not focus on the results of the re-engineering effort but rather on the usability of the process itself. The evaluation of the process is conducted using two kinds of empirical sources. Firstly the system assessment questionnaires include a feedback section. Secondly interviews are held where the process is evaluated in a semi-structured manner through baseline questions and free form discussion.

The interviews are started with a short recap presentation of the re-engineering strategy and how it has been applied in practice. After the short recap the following questions serve as a structure for the interviews:

- What is your view on metrics for re-engineering
- Do you think the re-engineering strategy helps to reach greater transparency and formality in decision making?
- Is the chosen re-engineering strategy easy to understand?
- Do you think the re-engineering strategy is usable in practice?
- Do you think the re-engineering strategy is usable for most of the varying systems in Roima?
- Do you see yourself using this process?

3.2 Literature review protocol

This section describes the literature review conducted in order to find a suitable re-engineering strategy to be applied in the case study for Roima.

3.2.1 Material search

The electronic sources used are:

- IEEExplore
- ACM Digital library
- Google scholar
- Scopus

The search strings are constructed by including alternative spellings for each of the question elements and linking them using the Boolean OR e.g.:
Population: software OR application OR system OR development
Intervention: re-engineering OR reengineering OR migration OR modernization OR modernisation OR modernising OR modernizing
Contrast: legacy
Outcome: model OR modeling OR strategy OR process OR approach
The search strings were constructed by linking the four OR lists using the Boolean AND.

Final search string(144 results): TITLE-ABS-KEY ((software OR application OR system OR development) AND (re-engineering OR reengineering OR modernization OR modernizing) AND (legacy) AND (model OR strategy OR approach) AND (migration OR migrating OR transition OR transitioning))

Additional search constraints used were limiting the results to the subject area of computer science and the document types to articles and conference papers.

The initial search yielded 144 hits. These were further narrowed down by excluding papers focusing on migration issues out of the scope of this study like for example migrating from monolithic to object oriented. A narrowed down selection of papers was then studied more closely and some forward and backward snowballing was used to find additional sources. The previous work of Althani and Khaddaj [8] was especially valuable as a starting point for snowballing.

3.2.2 Material selection

The re-engineering strategies chosen for a closer look were chosen by using specific inclusion and exclusion criteria. The criteria are presented below.

Exclusion criteria: The exclusion criteria aims to limit the scope of the study and take into account the limitations and constraints introduced by real-world applicability in Roima. The exclusion criteria concerns studies that focus on specific or irrelevant architectures or programming languages like COBOL, C or object-oriented strategies etc.

Examples of existing re-engieering strategies that were excluded are the CelLEST [16] process which migrates legacy system services to the web without any modification to functionality and little need for legacy code understanding. Another example is Lavery's [17] meta-process for incremental development of legacy systems and modelling the system to prepare for future evolution using UML to provide generic and specific views of the system. Some further migration methods are discussed by De Lucia et al. who present the methods and results of a technology transfer project [18]. They incrementally re-engineered the user interface to a newer web technology and created a wrapper for reusable parts of the legacy system.

Inclusion criteria: The inclusion criteria is directly related to the research questions and concerns any study that presents a re-engineering strategy or strategies. Any study that compares or analyses existing re-engineering strategies.

The primary sources and their identifiers are presented in Table 1.

Table 1: Primary Sources

S1	A perspective on architectural re-engineering [19]
S2	Migrating Multi-page Web Applications to Single-Page AJAX Interfaces [20]
S3	Iterative Reengineering of Legacy Systems [21]
S4	Planning the Reengineering of Legacy Systems [22]
S5	Renaissance: A Method to Support Software System Evolution [23]

4 Literature review

A large amount of research has been done on re-engineering legacy systems but no clear cut standard strategy has emerged over the years most likely because of highly differing source systems and constantly evolving technologies which change the re-engineering landscape. This section presents and compares a selection of re-engineering studies.

4.1 Different re-engineering strategies

This section presents a selection of different re-engineering strategies found in the literature review. The different strategies have varying focuses and approach re-engineering from different directions.

4.1.1 Automated analysis through architectural connections, S1

Sanchez et al. present an automated tool for discovering architectural connections [19]. Legacy software is in constant need of maintenance, updated functionality and assessment of quality. The architecture can often be undocumented or evolved in such a way that many dependencies and connectors are only visible in the code layer [19]. The CoodInspector tool is one way of automating the process of identifying or discovering the architecture on the basis of code [19]. However these kind of tools often produce low level models and have to be further analysed to achieve a higher level of abstraction in order to support re-engineering.

4.1.2 Migrating from traditional to single-page web application, S2

The authors present a "reverse engineering technique for classification of web-pages" [20] where similarly structured pages are grouped together and then further examined to build a set of potential interface components for the target application. The technique aims to support the migration process from traditional multi-page to single-page. In short the pages with small transitions in navigation are grouped together and the changing part between them is considered as a potential ajax component. An automatic tool RETJAX (Reverse Engineer To AJAX) for constructing these clusters of similar pages linked by navigation and a similar schema is also used.

The authors conclude that having an automatic tool for extracting information about needed UI-components and navigational paths can be valuable. The limitations are that the algorithm in the tool does not work for longer and more complex pages and also that the whole approach is focused on the client side and does not give any additional input for the backend.

4.1.3 An iterative re-engineering strategy, S3

Bianchi et al. present an iterative re-engineering process model and apply it in a case study [21]. They define legacy system as a “backbone of an organization’s information flow” and “main vehicle for consolidating business information”. These core systems usually decay over time in quality with maintenance becoming more and more costly. In order to improve the legacy system quality and enable adoption of new functionality a re-engineering process is needed. If a system is completely re-engineered the target system must be equivalent to the source system, which means that all maintenance and development of the source system must be halted during the re-engineering process. Alternatively all changes must be made to both systems causing considerable overhead. According to Bianchi et al. this problematic loop between existing maintenance and re-engineering can be avoided by an iterative re-engineering strategy [21].

The iterative re-engineering process means that the system will contain both re-engineered and legacy components at the same time. The main advantages to this process in contrast to complete re-engineering is that the system will continue to work during the re-engineering process. This preserves the know-how of different stakeholders including software maintenance and users as only small gradual changes are introduced with each iteration. Bianchi et al. experimentally apply their process on an industrial legacy system for supporting chemistry item distributors. The system is written in COBOL but Bianchi et al. claim that the iterative process is not technology or platform bound. They however work with the assumption that data is the most important part of a legacy system and especially the semantics and database schemas. In their iterative process the legacy database will be emptied gradually in tune with the other legacy components under re-engineering resulting in a completely new system in the end.

It is good to note that this iterative re-engineering process is not equal to a partial re-engineering strategy mentioned in [8] as the goal is to in the end achieve a complete revamp. Bianchi et al. especially emphasize that this iterative process aims to solve maintenance problems in legacy systems which for example the wrapping strategy in [8] does not change. However, the wrapping strategy is recommended to be used only in systems with both high business and technical quality. Therefore the comparison Bianchi et al. try to make is perhaps not relevant as maintenance is a larger issue in systems with low technical quality.

4.1.4 A five-step plan for re-engineering, S4

Sneed proposes a five-step process plan for re-engineering efforts which includes quantifying costs and benefits [22].

Legacy systems are often expansive in both lines of code and functionality which imposes many challenges for a re-engineering effort. Therefore careful planning is required ahead of time to succesfully re-engineer a legacy system. Another large aspect of planning is to quantify the costs and benefits of such a large untertaking in order to determine if re-engineering is in fact worth the effort. Sneed proposes that the key metric in analysing the benefits of re-engineering is maintainability where an increase also leads to increased system quality and productivity [22].

Project justification involves the analysis of the source legacy system, its maintenance process and overall business value. The re-engineering effort should pay itself back in the future in terms of better quality, easier maintainability and improved business value. The challenge is to measure and predict the expected outcome in advance. Justifying the project faces different challenges with different stakeholders for the system. Management is concerned with costs, sales wants new functionality, research and development departments want to completely rebuild the system in order to use all the newest architecture and programming language trends while existing system maintainers might want to keep the system as is. This means that no stakeholder in the project is usually keen on re-engineering according to Sneed [22].

Usually re-engineering is a sort of last resort or compromise when maintenance and development of new functionality has become too expensive or out right impossible due to technical constraints. The key for justifying a re-engineering project is measuring and quantifying the current state of the system. Firstly the measuring of maintenance needs to be assured by determining appropriate metrics and installing measurement tools. Sneed does not give any practical examples but a metric could for example involve gathering data from logged employee work hours. Hours of work completed in tasks related to bug fixes or other maintenance could be compared before and after the re-engineering project. Another aspect in project justification is measuring quality. Sneed proposes that a trained quality-assurance engineer trained in measurement theory should be in charge of collecting quantifiable data and generating reports based on typical quality attributes like code complexity, error proneness, coupling and so on. The last aspect of the justification step is assessing business value. In this step Sneed proposes that a business analyst lists the real monetary value of different applications and ranks them according to how much value the application generates for the company relative to each other [22]. This step can also be applied at lower level in assessing different parts of a system.

Portfolio analysis is a prioritisation technique where different applications or parts of the legacy system are ordered according to technical quality and business value which have been defined in the previous step. The expected gain of re-engineering an application low in both business value and quality is low and could instead be redeveloped or replaced by a commercial product. An application high in quality and low business value is not in need of re-engineering. In contrast, an application with high quality and high business value can be re-engineered, but the expected gain is lower while the priority of re-engineering a high business value application of low quality is of the highest priority.

Cost estimation is done by taking into account all the different components that are being re-engineered and estimating a total of time and resources needed to complete the project. Sneed claims that the estimation of costs is easier for re-engineering than a new system as the source system already exists giving a baseline for how many lines of code are needed. The cost of re-engineering is then calculated by multiplying the non-commented lines of code with a complexity factor and looking at previous metrics to determine how many hours are needed to produce that amount of code.

Cost-benefit analysis ties tightly into the project justification step and involves the comparing of cost estimates to the projected future cost savings. Here it is also important to estimate the life span of the system in order to determine between redevelopment, re-engineering or doing nothing at all. Simplified a short life span means that nothing should be done, a medium life span means that re-engineering is most cost-efficient and a very long timespan means that a complete redevelopment could be most beneficial.

Contracting is a step where the re-engineering project as a whole is divided into smaller more manageable pieces and ideally distributed to different development teams.

All in all these five steps together can take at least half a year according to Sneed [22].

Sneed believes that it is important to separate re-engineering into technical and functional re-engineering. He also claims that it is essential to start with a pure technical re-engineering project where only so-called "one-to-one" transformations are made so that the source and target systems have exactly the same functionality and that functionality can be validated by comparing the two systems. Sneed says that functional changes or improvements should be done in a separate functional re-engineering project after the technical re-engineering project is completed. This is a major distinction from many other re-engineering strategies which often also rely on forward engineering improved or new functionality.

With the constraint of limiting re-engineering solely to the technical aspect means that the possible outcome of a successful project is either improved maintainability, a migration to a new operating environment or programming language, greater reliability, preparation for new functionality in the future or a combination of the aforementioned. The largest problem with re-engineering according to Sneed is often that the focus of the project can easily be on solving technical issues while the actual goal of improved maintenance is forgotten.

4.1.5 Re-engineering with the Renaissance strategy, S5

Renaissance is an incremental re-engineering strategy presented by Warren and Ransom [23]. The simplified steps in Renaissance is to first determining a "stable basis" of the source system with re-engineering and then making smaller incremental changes to gain improved or new functionality. The strategy also tries to account for all different aspects regarding the re-engineering effort including technical, business and organisational factors. The authors also claim that Renaissance is flexible and generic in nature which means that it can be tailored to many different organisations, software architectures and languages.

The authors have identified four key requirements of a good re-engineering strategy [23].

- The method should support incremental evolution.
- Where appropriate, the method should emphasize re-engineering, rather than system replacement.
- The method should prevent the legacy phenomena from reoccurring
- It should be possible to customise the method to particular organisations and projects

The authors present several different kinds of strategies for re-engineering that have many similarities to the classifications in [8]. Continued maintenance is one strategy where nothing is done if the legacy system is of high technical and functional quality. Revamp is a strategy where the system's user interfaces are modified while leaving internal logic untouched. In contrast, restructure is modifying the internal structure without changing user interfaces. Re-architecture transforms a system by migrating to a new architecture. Redesign with reuse is transforming a system by redeveloping it from scratch but utilising some legacy components. Finally, replace is a total replacement of the original system.

The Renaissance process on a high abstraction level consists of four phases in a continuous development loop which goes from "plan evolution" to "implement", "deliver", "deploy and use" and finally back to "plan evolution".

Evolution planning The "Plan evolution" phase starts off the Renaissance re-engineering process and involves assessment of the current system from different viewpoints taking into account technical, organisational and business needs in order to choose an appropriate re-engineering strategy. To do this evaluation the authors suggest using their own "attribute rating scheme" [13]. The evolution plan should be done carefully as it is the base of the whole re-engineering effort. The three main measures used are technical quality, business value and organisational factors.

Technical quality can be evaluated with documentation and individuals with deep knowledge of the system. Another more structured evaluation method is to create context models using UML where the system and its components are abstracted from different views.

Business value can be evaluated with the help of business goals as the future goals will reflect future requirements and functionality. For example, if the future goal of the business is to start offering a cloud SaaS-platform (software as a service) an existing desktop application will eventually become redundant, which means that continued maintenance is the only viable option. In contrast, the existing business logic might exist largely in a specific database which would be extremely costly to migrate which means that the database has to survive in the future in some shape or form regardless of other business goals.

Organisational factors can be evaluated by understanding the viewpoints of the system users, operators, maintenance and developers. Challenges in the organisation can include change resistance, staff availability and workload and know how.

Evolution planning can be conducted at different levels of the system with different intensities. Quick estimations can be done at a high level to give approximate guidelines while more detailed estimations are achieved by lower level inspections. The estimations are conducted iteratively starting from high abstraction levels to reduce unnecessary work as clearly unfit re-engineering targets are filtered out early. The lower level estimations which require more work are conducted lastly.

The aims of the evolution planning phase are to answer relevant questions about the system that increase the knowledge and enables choosing a suitable re-engineering strategy. Firstly is the system critical for the business and what are the business goals? The business goals help to understand the modernisation requirements. To fulfil the requirements it is also essential to understand the technical state or quality of the system. Other questions that should be answered are what the planned lifetime of the system is and does the organisation have sufficient resources to successfully complete the modernisation. The available resources are also affected by how receptive to change the organisation and its workers are as modernisation requires adapting new architectures, technologies and ways of working. The evolution plan should be done carefully as it is the base of the whole re-engineering effort. The three main measures used are technical quality, business value and organisational factors.

A successful assessment requires experienced experts in different roles, so that deep business and technical information can be collected. Ransom et al. divide experts into three categories; application business experts, legacy functional experts and legacy implementation experts [13]. The application business experts are individuals who understand the business and related processes at a deep level (senior staff).

Legacy functional experts are individuals who understand how the system is used by end users (system operators). Legacy implementation experts are individuals who have development and maintenance experience with the system (developers). The first two categories are usually easy to fill but the older legacy system is the higher chance that the original developers have left or retired. If this is the case alternative sources of information have to be used like documentation or version history.

Implement plan Based on the results of the planning phase an evolution or re-engineering strategy can be picked. Each of the strategies has different risks associated with it which are presented in Table 2. The table can be used for risk management and cost estimation.

Table 2: Associated risks table [23]

	Continued Maintenance	Revamp	Restructure	Rearchitecture	Redesign with Reuse	Replace
Lack of system knowledge	x	x	x	x	x	x
Lack of experienced maintenance personnel	x	x				
Poor documentation	x	x	x	x	x	x
New technology skills shortage				x	x	x
Legacy technology skills shortage	x	x	x			
Errors introduced during evolution			x	x	x	x
Technology immaturity				x	x	x
Loss of embedded business rules						x
System will not meet evolution requirements	x	x				
Obsolete operational environment	x	x	x			

The Renaissance method continues with the results of the evolution planning and constraints being saved in a so-called information repository. The idea is to over time accumulate an evolution record of the system, where different new functionalities and constraints and possible associated re-engineering efforts can be clearly seen.

Lastly comes the implementation phase where the plan is executed and the legacy system is re-engineered. This step will vary largely based on the chosen strategy and the different requirements of each system. The Renaissance method does not involve any specific technical guidelines for the actual implementation and testing.

Renaissance was evaluated by several companies and the authors conclude that it was well received and easy to understand and follow. In practice the strategy selection phase usually resulted in a hybrid model where different strategies were applied within a single legacy system depending on the nature of specific components. A benefit of Renaissance that was mentioned is also that it is scalable in the sense that the same rules and phases can be applied both at low or high levels of abstraction. This means that a relatively quick overview of a legacy system is easy to produce and the relevant candidates for re-engineering are therefore simple to identify. As negatives the process introduced overhead especially for the first time used. The authors claim that this is to be expected for a new process and conclude that the benefits of better cost efficiency and lower risk in re-engineering compensates for the overhead introduced by Renaissance.

4.2 Comparing existing re-engineering strategies

The results of the literature review are presented and categorised in this section. The different re-engineering strategies are categorised as either potentially applicable or non-applicable for small organisations with complex software systems. An applicable strategy is generic, lightweight and easy to understand. A non-applicable strategy is generally too complex and difficult to understand or clearly incompatible with smaller organisations. In practice this excludes strategies which require many resources in terms of developers, testers, and software architects.

4.2.1 Non-applicable strategies

Some strategies are not directly applicable in the case of this study but can have interesting observations none the less. These non-applicable strategies are valuable for this study in regards to extracting general best practices.

A common issue with re-engineering strategies in contrast to the set requirements is that they are too complex or difficult to use. Complexity and difficulty often arises from the need to learn new mathematical or architectural languages. One example of this is S1 where connections in code are analysed through a special mathematical notation. These kinds of automated tools are unfit for the needs of many smaller organisations as they are extremely heavy in terms of needed resources and know how.

A key issue is that some re-engineering strategies have a lot of overhead which can increase the relative cost of the already expensive re-engineering effort to unbearable levels for smaller organizations. There are many issues in S4 relating to the scope and resources required in a comprehensive planning process that Sneed describes. The process seems to be more suited for very large organisations that have strict quality requirements and clear estimations of the monetary value of different applications which is understandable as Sneed has been part of large re-engineering projects for large banks.

Other issues arise from a lack of genericity and a focus on too narrow types of source systems or components. For example the process in S3 works on the assumption that the whole system will be revamped with the database as a focus. As such it is not in scope for this paper and does not meet the inclusion criteria. In S2 the authors differentiate between previous legacy system migration to web-based systems and older web-based systems to single-page AJAX applicationsauthors [20]. The article is as of now 11 years old, and it could be argued that todays challenge is to migrate the older SPA-applications to micro service architectures. So the constant changing software development landscape and fast changing technologies bring a constant need for re-engineering as more and more systems "fall" into the legacy category. In fact the definition of legacy system can also be applied to older web-applications if they no longer are fit for future needs even though legacy has traditionally been used to describe older pre-web systems.

A summary of the non applicable studies is presented below:

- S1: A perspective on architectural re-engineering [19]
- S2: Migrating Multi-page Web Applications to Single-Page AJAX Interfaces [20]
- S3: Iterative Reengineering of Legacy Systems [21]
- S4: Planning the Reengineering of Legacy Systems [22]

4.2.2 Potentially applicable strategies

This section aims to answer research questions RQ1 and RQ2; “What re-engineering strategies exist that are easy to understand and applicable for many different kinds of systems?” and “What specific re-engineering strategy can be applied at a small organisation dealing with complex legacy systems”?

The issues with legacy systems and the cost of developing completely new systems without losing business critical functionality, or the costs becoming unbearable are well known. Many re-engineering strategies are narrow and limited only focusing on specific transformations from for example monolithic to cloud. However general and generic re-engineering or evolution strategies for supporting a replicable and controlled process are few and far between. Many strategies are also difficult to grasp and require many resources both in order to master them and apply them successfully in practice.

Based on the results of the literature review in this study RQ1 is answered as follows. Renaissance (S5) is a strategy that does fill the requirements of working for different source systems and being easy to understand. Renaissance is generic in nature and not bound to specific types of source or target systems. Renaissance is also easy to understand relative to other re-engineering strategies, as it does not require using special mathematical or architectural languages or models as a part of the process.

Based on the results of the literature review RQ2 is answered as follows. Renaissance is a strategy that can be applied at a small organization dealing with complex legacy systems. Renaissance is a strategy that could work especially well for a smaller organization, as it claims to be lightweight in terms of needed resources. The Renaissance strategy also aims to enable future long-term evolution of the system i.e., multiple and continuing re-engineering cycles should be made easier. This can further help to reduce needed development resources in the future. Finally, Renaissance is flexible as the re-engineering effort can be adapted freely by the organization, by choosing relevant metrics and assessment techniques. This flexibility is important in order to support a complex system.

Re-engineering strategies that best fit the requirements in this study are:

- S5: Renaissance: A Method to Support Software System Evolution [23]

4.2.3 Common best practices

This section presents general best practices that are the same both in non-applicable and applicable strategies.

Most of the different re-engineering strategies have some similarities and try to address the main challenges of re-engineering. A common aspect of many re-engineering strategies is careful system assessment and planning to determine what exactly should be done. This often involves the classification of the system on the basis of technical and functional quality. In turn these classifications can be used to choose an appropriate re-engineering strategy going forward (replacement, wrapping, doing nothing etc.). As many re-engineering strategies focus on technical challenges Sneed presents many interesting aspects of evaluating the real costs and benefits of a re-engineering process [22]. This brings us to another common theme where different metrics of especially business and organisational aspects are used to motivate a re-engineering effort and prioritize which components should be re-engineered first.

A common theme is to clearly define what the requirements, constraints and challenges are in order to make informed decisions. Re-engineering can mean many things and therefore it is necessary to crystallize what exactly is meant to happen to the source and target systems. For example all stakeholders can be unanimous about a plan to update an existing system. However what the updating really means can vary wildly between stakeholders. The different categories are useful to get a common bearing on what the current situation of the system is, so that an informed decision or at least an non ad hoc decision can be made.

4.3 Choice of re-engineering strategy for the case study

This section motivates why a specific re-engineering strategy was chosen for the case study. Results from the comparisons of the literature review (providing answers to RQ1 and RQ1) served as a basis and only applicable strategies were considered. As only one strategy S5: Renaissance was defined as applicable, it was the one chosen. The search for additional applicable re-engineering strategies was not continued as several aspects of Renaissance appeared suitable.

Flexibility was described by the Lead Software Architect as an important factor as the whole product stack of Roima is broad and uses many different kinds of systems with varying characteristics. As resources are scarce and continuous tailoring of existing software for customers takes much of the developing time another important factor is that the strategy scales according to available resources. These requirements were especially highlighted in the Renaissance strategy. The choice of Renaissance was also affected by it being perceived positively by Roima professionals while time constraints further encouraged to start the case study instead of continuing the literature search.

5 Roima case study

This section instantiates and evaluates the Renaissance re-engineering strategy in a case study at Roima Intelligence Oy.

5.1 Case context

Roima Intelligence Oy (Roima) is a Finnish software developer and integrator that mainly focuses on serving the manufacturing industry. Roima's main areas of expertise are enterprise resource planning (ERP), manufacturing operations management, product management, intra logistics and machine vision. Roima has recently undergone a series of acquisitions and mergers that has led to an ever larger product portfolio implemented with wildly varying programming languages and frameworks which is a challenge for integration and synergy gains. The broader long-term goal for Roima is to deepen the integration between the different systems and improve the possibility of reallocating programming resources between different projects efficiently as workloads shift. To help reach this goal a preliminary decision to move towards a company wide usage of React has been made with the help of a consulting company.

Lean System is a relatively large system that has many different ERP-modules. Many parts of the system have evolved and undergone re-engineering to different extents while others are essentially legacy. As shown in Figure 2 the technology stack is quite extensive especially regarding the user interface (UI). The figure only shows Lean System's technologies meaning that a comprehensive view of the whole Roima stack including all departments and systems is multitudes larger. Continuous integration between different components within Lean System itself is needed as well as between other systems in Roima. The goals of the re-engineering effort are to increase maintainability, performance and further evolvability in the most costeffective way. Another goal of the re-engineering effort is to come up with a process that can be applied broadly for any part of the system.

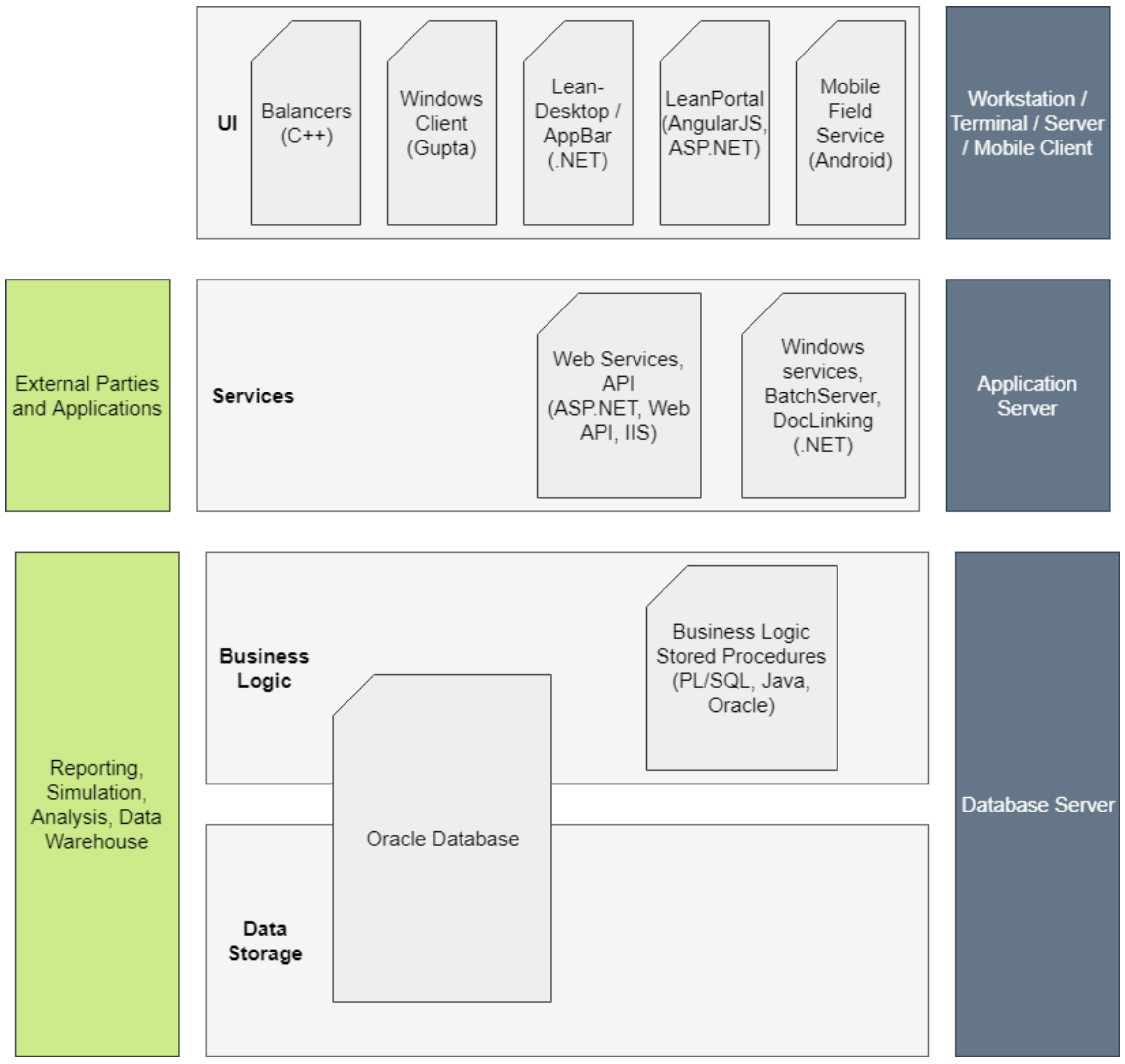

UI
Balancers (C++)
Windows Client (Gupta)
Lean-Desktop / AppBar (.NET)
LeanPortal (AngularJS, ASP.NET)
Mobile Field Service (Android)
Workstation / Terminal / Server / Mobile Client
External Parties and Applications
Services
Web Services, API (ASP.NET, Web API, IIS)
Windows services, BatchServer, DocLinking (.NET)
Application Server
Reporting, Simulation, Analysis, Data Warehouse
Business Logic
Business Logic Stored Procedures (PL/SQL, Java, Oracle)
Oracle Database
Data Storage
Database Server

Figure 2: Lean System Architecture and Technologies

The web-application “LeanPortal” of the system has become fragmented as a large part consists of older ASP.NET and a newer part being a single page application (SPA) using the more modern AngularJS framework. The two different parts of the system are integrated in such a way that an installation can include one or both systems depending on the needed functionality. The two different systems share some back end components and rely on the same databases where much of the business logic is handled. LeanPortal will be the main focus of this case study and the scope is further limited to three portals including two older “classic portals” and one newer “SPA portal”.

Much information about different fine grained functionality only exists in the source code and maintainability and testability is becoming increasingly difficult. AngularJS as a framework has also not been as long lived as originally expected and is already showing weaknesses in performance. These issues in combination with increasing organizational pressure to move to an even newer modern web-technology React, are triggers for the re-engineering effort.

The goals of the case study are to study and observe the chosen re-engineering strategy in a real world scenario. These observations will serve as a starting point for an evaluation of the strategy itself by Roima professionals.

5.2 Experimental instantiation

This section describes how the chosen artifact (re-engineering strategy Renaissance) is instantiated in a real-world scenario at Roima. The instantiation of the Renaissance strategy is studied in a controlled and restricted scope. First of all the scope is limited largely by only completing the first phase of the Renaissance process (Evolution planning). The scope is limited further by focusing on a specific small part of LeanSystem’s web based UI-layer LeanPortal. LeanPortal consists of many different subsystems “portals” that provide different functionality that are split into older “classic portals” and newer “SPA portals”. This study only includes three portals including two classic portals; “Travel Expenses”, “Service” and the third SPA portal “Manufacturing”. The instantiation is conducted by doing evolution planning for these three portals in order to prioritize re-engineering needs and evaluate different risks and requirements related to each one. The chosen portals are varied in used technologies and business value and should be relatively representative of the LeanPortal system as a whole.

Travel Expenses Portal contains different functionality to manage employee travel expenses. It was selected to be included in the study as a re-engineering effort has already been started and it is being currently migrated to React. The main goal is to find out if the Renaissance strategy can validate that the already made re-engineering decision has been correct.

Service Portal functions as a management tool for different service tasks related to factory workstations and machines. It was selected to be included in this study as it is a main focus of the business strategy and has not been targeted by any recent re-engineering efforts. A goal is to evaluate if Service Portal should have been higher in priority compared to the Travel Expenses portal when the decision to re-engineer Travel Expenses was made.

The Manufacturing Portal is a tool for managing work and task queues for specific work stations on the factory floor. More specifically the portal supports different functionality related to states/phases, materials, quality measurements, documents and images of ongoing tasks. The manufacturing portal was chosen as a target for the experimental instantiation as it is a part of the system that according to general consensus has high business value and also has some performance issues in certain situations. The performance issues are mainly caused by large amounts of data being displayed for some tasks.

All of the three portals are based on frameworks and third party components that are nearing end of life. In addition to this and some performance issues the new requirement that has sparked the re-engineering process is a need for React support in LeanPortal.

The goal is to follow the Renaissance strategy, study the usability of the process and observe if new insights compared to previous re-engineering efforts can be discovered. The observations of the experimental instantiation should help to evaluate if the chosen process adds any value, introduces overhead and is easy to use.

5.3 Evolution planning

The Renaissance process starts with the evolution planning phase where different parts of the system are assessed and classified as re-engineering candidates. By applying the assesment method presented in [13] two main choices have to be made. Firstly an assesment technique must be chosen. The assesment can be based on expert opinion or quantitative metrics. Metrics give the most objective results but can be too constrained [13] and expert opinion is recommended if available. No meaningful and realistically usable metrics could be defined for LeanPortal. Therefore expert opinions gathered through questionnaires were used in this case study as the assessment technique.

The second decision regarding the assessment method is determining the assessment level. As many different properties of a system can be assessed with varying degrees of granularity the level should be determined based on how detailed information is desired. A high level assessment is recommended in all cases as it is relatively quick and rules out clearly low value targets while also exposing potential candidates for a lower level detailed assessment. In order to test the Renaissance process as fully as possible both a high and low level assessment were conducted. Different assessment levels are depicted in Figure 3.

The high level assessment targets the web layer and includes all three of the portals; Travel Expenses Portal, Service Portal and Manufacturing Portal. The low level assessment focuses solely on Manufacturing portal's Resource tasks view and it's components. The high level assessment is conducted with questionnaires consisting of questions in three different categories highlighted[13] in the Renaissance method. The categories are Business value, Technical quality and Organisation factors. In these assessments Roima experts with deep knowledge of each category were consulted.

The lower level assessment follows the same base principles of business value and technical quality related questions as the higher level questionnaire. However, the lower level questionnaire also includes questions related to functional, structural and environmental aspects of the system. In a strict sense functional factors show how the business processes are implemented, structural factors show main configuration elements and environmental factors relate to physical devices. This questionnaire takes advantage of the flexibility of the renaissance process and instead of physical devices focuses on issues regarding the software framework and back-end API.

5.4 Results

This section presents the results of the assessment questionnaires and also presents results of the interviews evaluating the Renaissance process itself.

```
LeanSystem
└─LeanPortal
   ├─LeanPortal classic
   │  ├─Travel Expenses
   │  └─Service
   └─LeanPortalSPA
      ├─Timereporting
      ├─Purchase
      └─Manufacturing
         ├─Assembly
         └─Resource tasks
            ├─Materials
            ├─Serial numbers
            └─Quality failures
```

Figure 3: LeanSystem assessment levels

5.4.1 High level assessment

This section presents how an assessment of LeanSystem was conducted in practice using principles discussed previously.

Pilot assessment A pilot test round of assessment was done first with a small group in order to gather experiences and feedback. The first assessment questionnaire[Appendix 1.] received feedback about having some questions that were too vague and open to interpretation. For example several participants had interpreted the term "organisation" to mean the customer instead of Roima itself. Some participants had also interpreted questions regarding business value (How important is the system for the organisation) as how much the system is being used by Roima internally, instead of how valuable the system is in terms of sales. The improved second questionnaire[Appendix 2.] aimed to specify questions more clearly and also provide some additional guidance in form of descriptions for some questions.
Examples of changes (The relevant portal name is added to all revised questions):

- The question "Is the system important to the organisation?" was changed to "Is Travel Expenses Portal an important part of LeanSystem?" with an additional description "For example a subsystem can be deemed unimportant if removing it would not affect sales of other subsystems or the system as a whole".
- The question "How large is the system?" was changed to "How large a part

of LeanSystem is Service Portal?” with an additional description “Relative to other portals”.

- The question “Estimate the availability of experienced experts” was changed to “Estimate the availability of experts who have experience working with Manufacturing Portal.” with the additional description “Especially experts with deep understanding of the architecture and code structure”.

Main assessment The second and main round of high level assessment was expanded to include a larger group of experts with more varied roles. The results of this assessment questionnaire are presented below.

The results were combined into Figure 4 by averaging the answers for each portal. The average value was corrected by taking into account that a score in some questions affected the result negatively and some positively. For example a high score in the question “Is Travel Expenses Portal difficult to maintain?” affects technical quality negatively while a high score in the question “Does Travel Expenses portal have good performance?” affects technical quality positively. Generally the results from sections contribute to a corresponding bar in the chart. A deviation is made for the Risk factor bar which is affected by results in Organisational Factors and also the system size estimate from the Technical quality segment.

The figure can be interpreted as Travel Expenses having the least business value and Manufacturing the most. In contrast, Travel Expenses has the highest technical quality and Manufacturing the lowest. The risk factor is highest for the Service portal and lowest for Manufacturing.

5.4.2 Low level assessment

As Manufacturing was deemed to be the most potential re-engineering target in the high level assessment round it was a natural choice for a more granular assessment. This low level assessment was conducted on a component level with another questionnaire[Appendix 3.]. More specifically the low level assessment only included some components from one view of the portal (Resource tasks view). The components and their identifiers are presented in Table 3. The results are plotted on a business value and technical quality chart in order to prioritize the different components as re-engineering candidates.

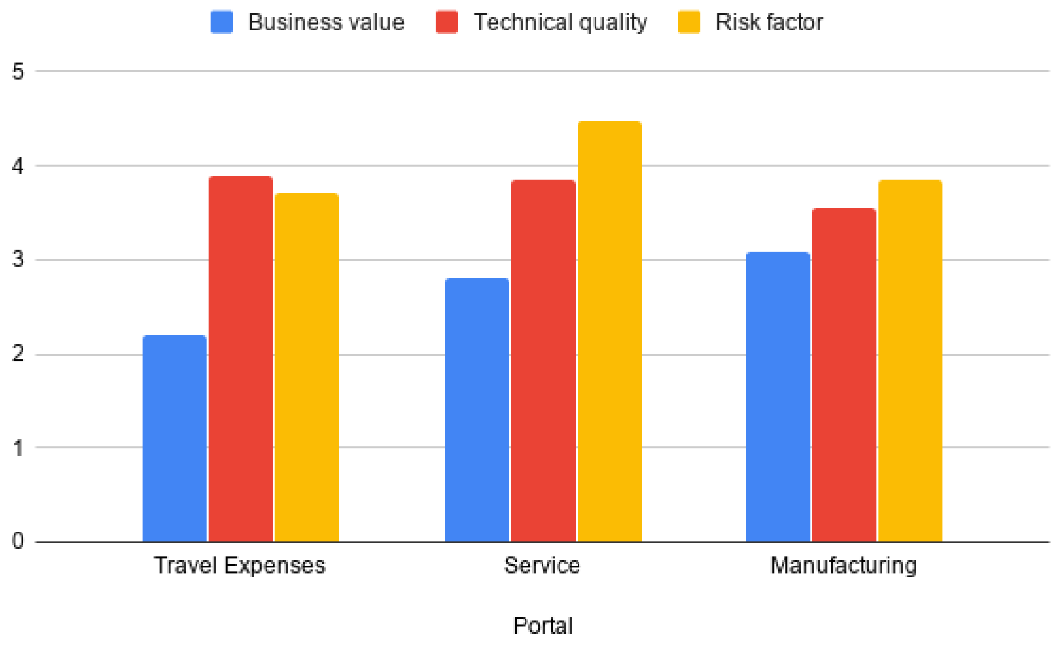


Figure 4: Assesment results

Table 3: Manufacturing portal components

c1	Task cards
c2	Quality failures
c3	Serial numbers
c4	Materials
c5	Texts and documents

By reading Figure 5 we can categorize the components to four different categories; Replace with commercial package, no re-engineering, low priority re-engineering and high priority re-engineering. Replace with commercial package category includes targets that are both low in technical quality and business value. In Figure 5 we see that no such targets are present. The no re-engineering category includes targets with high technical quality but low business value. The component c3 could be placed in this category. The low priority re-engineering category includes targets that have high technical quality and high business value. This category includes c1 and to a lesser extent c2 and c4. Lastly the high priority re-engineering category consists of targets that are low in technical quality but high in business value. This category includes c5. The components matching categories are visualised in Table 4.

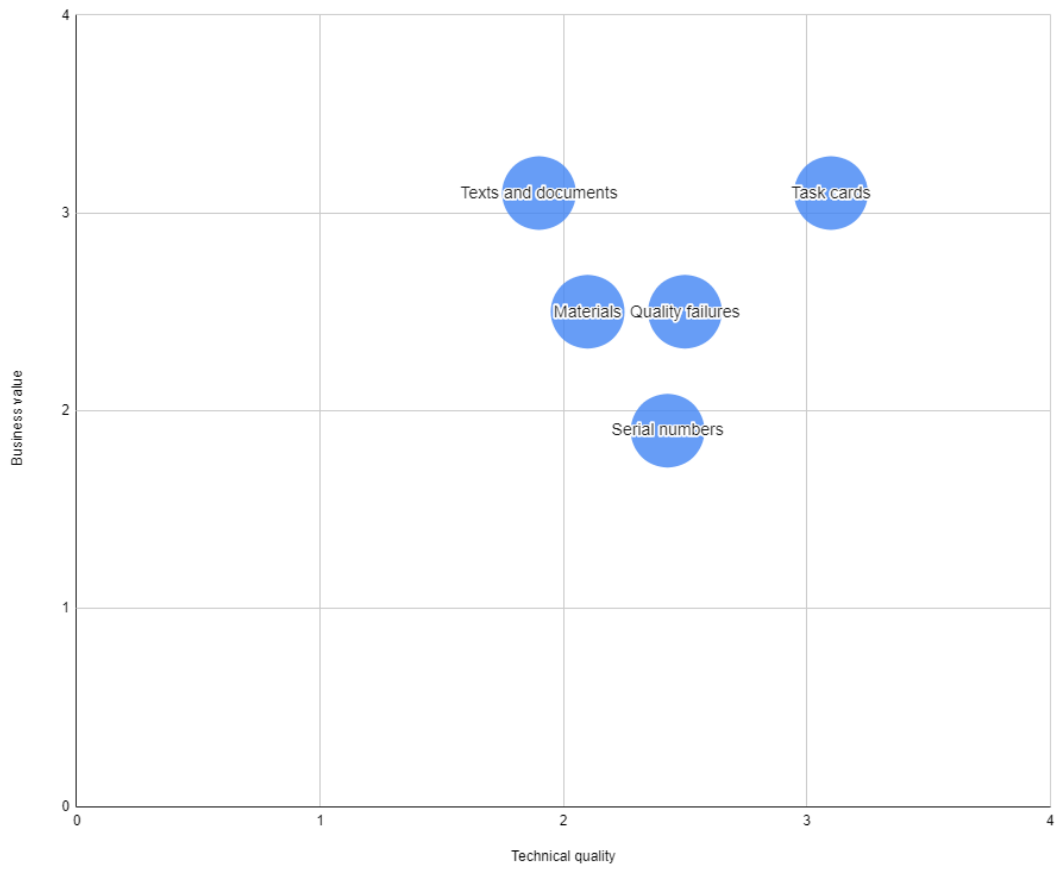


Figure 5: Component level business value and technical quality

These categorizations should not be naively interpreted as other results from the questionnaire and especially textual answers can lead to other conclusions. The evolution planning phase continues with analysing the results of the assessments and determining what the best course of action is. This analysis is presented in the discussion section where the results are interpreted in greater detail.

5.4.3 Feedback and interviews

This section presents feedback gathered from the assessment questionnaires and results of interviews with Roima experts.

High level assessment evaluation The final amount of participants in the main high level assessment questionnaire was 9. Feedback included several comments

Table 4: Categorized re-engineering candidates

Replace	-
No re-engineering	c3
Low priority	c1,c2,c4
High priority	c5

about the chosen scope where the three systems chosen were web-pages (portals). Especially sales and business oriented experts thought that it could be beneficial to scope the questionnaire based on whole business logic segments instead of the web-page based portal scope. On the other hand some Technical experts found that a more fine grained scope with for example specific components of the web-page could also be useful. Several experts commented that the questionnaire gave a good structured basis for discussion. A general consensus also was that the questionnaire was quick to do and required minimal effort.

Low level Manufacturing portal assessment evaluation The lower level assessment focusing on different components in Manufacturing Portal was regarded much in the same way as the higher level assessment. One comment stated that "For many of the questions it was very hard to select an answer from a one-dimensional scale/quantity because the thinking/reasoning process would automatically start to intertwine other dimensions to the question".

Several respondents commented that thinking about and answering the questions was useful and resulted in some "thinking out of the box".

Interviews This section presents results of interviews conducted with experts evaluating the whole Renaissance process and the conducted assessment questionnaires. Three interview sessions were held with different experts. Some sessions were kept in groups and some one on one. The sessions were generally free form discussions where the main questions were being used as conversation points. Answers of the respondents are presented below. The questions were not explicitly presented in the form they are written here but answers are still grouped under generally corresponding questions.

1. What is your view on metrics for re-engineering?

   (a) Metrics would be good but current system does not support it.

   (b) Capabilities for extracting metrics from work hour reporting theoretically exist but need development and integration.

   (c) Version control does not provide good metrics at the moment.

2. Do you think the re-engineering strategy helps to reach the set goals of greater transparency and formality in decision making?

   (a) Could be a good addition to the toolbox, the value does not lie only in the result but also in doing the analysis and discussions.

   (b) There's a need to prioritise different tasks and components.

   (c) Provides a chance to think and discuss different things.

   (d) I like it! Could we get this into everyday work? Things that should be done have been brought up more systematically and transparently.

   (e) Big challenge is to balance between new functionality and improving existing functionality.

3. Is the chosen re-engineering strategy easy to understand?

   (a) Personally yes but generally a bit abstract and requires familiarization.

   (b) A bit difficult to grasp.

   (c) Quite all right but would be better with more circles and broader scope.

   (d) It is easy to understand.

4. Do you think the re-engineering strategy is usable in practice?

   (a) Makes sure that different options and aspects are considered.

   (b) Prevents mistakenly forgetting to consider something important.

   (c) Helps to understand and perceive different aspects of the system.

   (d) Are there risks that this process limits thinking to the options provided by the process?

   (e) Questionnaire usability parts would benefit from usability analysis from the UI-team. Perhaps usable new metrics.

   (f) Could be usable in a structured time frame but needs a broader group of respondants/attendees especially of people who are in contact with customers.

   (g) More high level business goal related information would be good to gather with this process.

   (h) Could help to better understand client needs. Important to be close to the customer. Push things that are important to us (Roima). Some things are known to be important but are waiting for resources/approval/decisions.

5. Do you think the re-engineering strategy is usable for most of the varying systems in Roima?

   (a) Looming Lean Client re-engineering effort could maybe make use of this.

   (b) Could be usable especially with Lean Client.

6. Do you see yourself using this process?

   (a) I can't see myself constructing a questionnaire like the one used in this process. To be usable in practice the process itself would have to be raised to some kind of strategic role in the organisation. There would be a need for people to invest themselves into the process. If I had time I would not have anything against studying and using this process, but going in cold it couldn't be done.

   (b) Someone would have to really think about what is needed in order to use this process.

   (c) I could envision myself being a part of constructing questions for the questionnaire.

   (d) Would need to practice and study how the results are displayed but could think about new questions.

   (e) I would participate in scoping and creating questions for the questionnaire.

7. Miscellaneous

   (a) Massive business, difficult to manage with only a hunch.

   (b) Some architectural decisions have been overruled from higher up.

   (c) There has been challenges in spreading information.

   (d) There is no consensus about how configurability should be handled.

   (e) We should work on identifying what is important and what parts need to be configurable.

Table 5: Rough interview result summary

Question	Negative	Mixed	Positive
1. What is your view on metrics for re-engineering?	1	2	0
2. Do you think the re-engineering strategy helps to reach the set goals of greater transparency and formality in decision making?	0	2	3
3. Is the chosen re-engineering strategy easy to understand?	1	1	2
4. Do you think the re-engineering strategy is usable in practice?	0	1	7
5. Do you think the re-engineering strategy is usable for most of the varying systems in Roima?	0	0	2
6. Do you see yourself using this process?	1	2	2

The interview results are summarised in Table 5 with rough categorizations into negative, mixed or positive results. The table should not be considered to be exact due to the nature of many answers being ambiguous. Reading the table it can be summarised that most respondents found some challenges with using metrics for re-engineering Lean System. Most respondents had a generally positive view on the Renaissance process especially in relation to improving decision making. The process was also generally deemed as easy to understand even though one answer stated that it was "A bit difficult to grasp".

Most respondents also found the strategy to be useful in practice in different ways, ranging from "helps to understand and perceive different aspects of the system" to "Could help to better understand client needs". On the other hand, the scope of assessments and how Renaissance could be used in everyday work produced some mixed opinions. Answers related to the applicability of the strategy for varying systems was universally positive, even though not all respondents had any opinion on the topic at all. Finally, respondents were mixed in the view of using the process themselves where some were completely on-board stating "I would participate in scoping and creating questions for the questionnaire". Others were ready to use the process after some further studying, and conditions like stated in "Someone would have to really think about what is needed in order to use this process".

6 Discussion

This section analyses and discusses the results and implications of the experimental instantiation of the Renaissance re-engineering strategy.

6.1 Analysing assessment results

The Renaissance strategy's evolution planning phase continues with analysing the results of the assessments in order to determine a suitable evolution strategy.

High level assessment We start the analysis by plotting the results on a business value and technical quality chart in the same way that Figure 5 on page 32. The resulting Figure 6 illustrates how the absolute scores are similar and all three portals can be said to have both high business value and technical quality. As the results are absolutely scored quite close to each other the portals are also set on a relative scale in Figure 7 to see differences more clearly. The first outcome of the assessment should be prioritizing different systems as candidates for re-engineering.

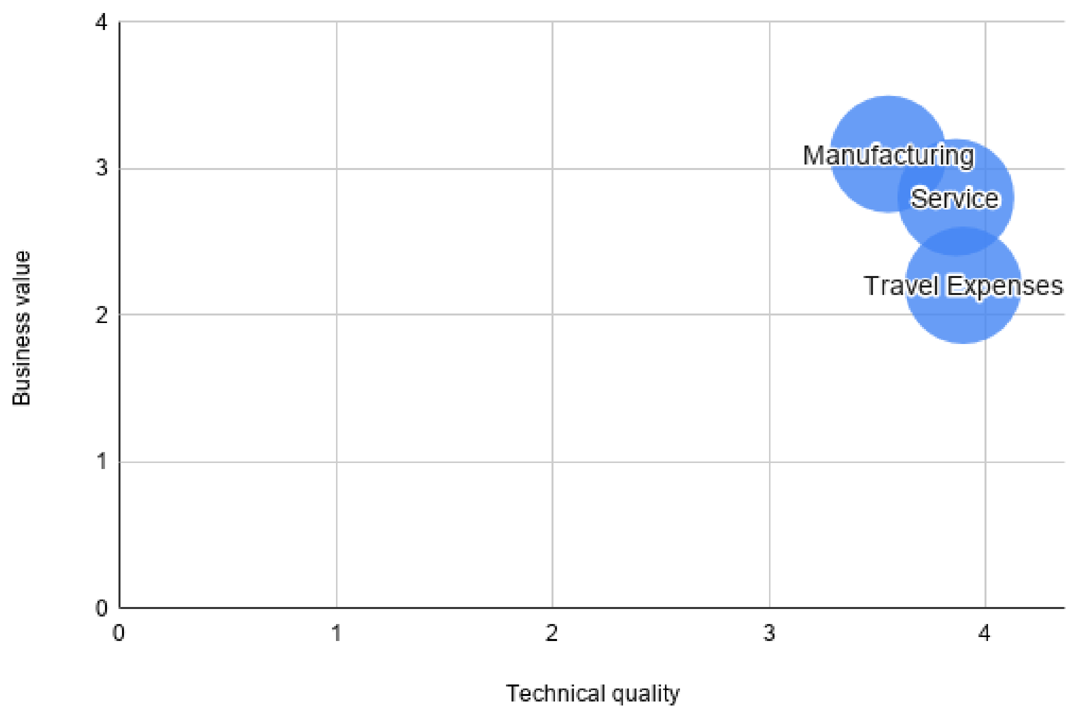


Figure 6: Absolute business value and technical quality

Manufacturing Manufacturing Portal has the lowest technical quality score. This can be seen as surprising in the sense that it is the newest of the three portals and also developed with the latest framework and programming language. The lower technical quality of Manufacturing can be in part explained by the higher complexity and difficulty in maintenance caused by a more tailored approach. As the other older portals are based on a more configurable easily maintained framework the highly tailored and feature rich Manufacturing portal gets a lower score. Configurability is one of the largest technical challenges for LeanSystem at the moment and there are different views between the benefits of the very configurable older portals and the more tailored new portals. A high amount of configurability makes many common and simple customer customizations fast and easy but when a new requirement is not supported by the configurable framework a very high development cost is suddenly introduced. Another issue with the configurable framework has been that it is "too easy" to use it resulting in some functionality being held back by the configurable frameworks restrictions especially in terms of usability when many things are formed through the same mould.

Manufacturing has the highest business value score which is even further validated by several written answers where it was described as "Key Solution for the Paperless Production", "Important spearhead for entering new customers and markets" and "Most of our customers are in manufacturing business and manufacturing operations support is Lean System products key strength". Manufacturing can be seen as the prime candidate for re-engineering with the highest business value and lowest technical quality.

Travel Expenses and Service Travel Expenses has the lowest business value and highest technical quality. This can be interpreted as it being the lowest priority re-engineering target of the three. As a re-engineering effort has already been started targeting this portal it can be deemed to be a suboptimal choice. A better target for re-engineering based on business value and technical quality would have been Service Portal. This is further underlined by written answers given to the questions about "other business perspectives that are important". Travel Expenses is described as a feature that "is a must for every company" but on the other hand can be handled by many different focused third party solutions. It can also be difficult to remain competitive against these focused tools and "integration to other tools" was described as one possible edge. Contrasting to this the Service portal was described to be one important "focus area" for the business. It can be argued that since Travel Expenses is a replaceable tool and not a business focus area it might give better returns to focus development effort on Service which is not as easily replaceable and is included in the focus area of the business.

However, when we also consider the risk factors in Figure 4 on page 31. the Travel Expenses portal rises as a re-engineering candidate compared to Service portal. The scores creating a greater risk factor for Service are higher complexity and also being a larger part of LeanSystem. Even though the benefits of re-engineering Travel Expenses are probably less than Service the risks are also lower meaning that it can be deemed as a good first pilot for the upcoming React migration. When the quirks and technical challenges of migration are first resolved with a lower risk target the risks should be lower for subsequent re-engineering efforts.

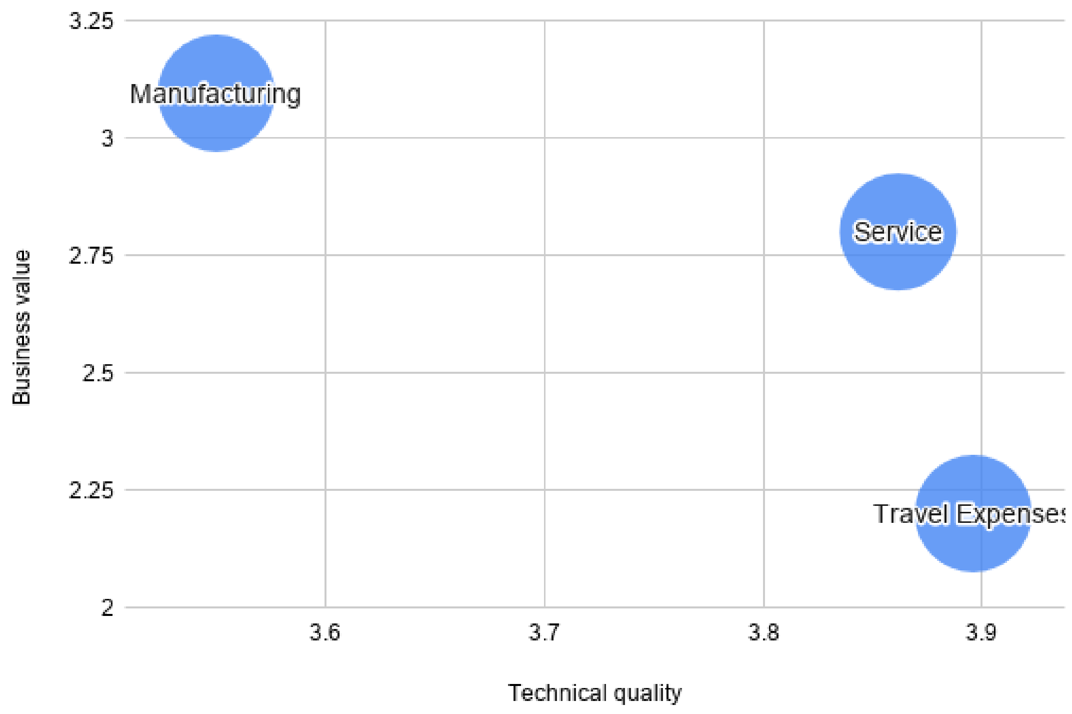


Figure 7: Relative business value and technical quality

Low level assessment The questionnaire starts with asking what is an important attribute for Manufacturing Portal. Usability and functionality were the key attributes according to respondents. The main need that is solved for the customer purchasing Manufacturing portal is enabling efficient operations at the factory floor with the help of real time task lists with instructions. The usability and functionality attributes are highly relevant in order to satisfy this need. A good level of usability and functionality ensures that factory floor personnel can quickly get the info needed in order to complete operations and be informed in real time about issues or changes.

The causes for different tailoring needs were split between the respondents almost evenly. Therefore UI-changes, using different features, customizing functionality and differing business processes all contribute to tailoring needs. The Renaissance process aims to create an evolution plan that mitigates future modernisation needs and in this context the configurability of Manufacturing portal was taken into special focus at it is widely seen by different experts at Roima as being the most challenging and complex issue to handle efficiently.

In terms of business value the respondents were asked to estimate where more monetary value has been generated; selling the standard product version or selling tailor made versions of Manufacturing portal. The answers heavily lean towards the tailor made versions generating more value. However, the answers to the next question which asked if Manufacturing Portal needs less or more configurability to generate value in the future somewhat contradict this proposition. The answers indicate that more configurability is needed to generate value in the future.

The tailor made versions specifically don't require as much configurability as the product version. The product version is more configurable as it has to fit many different customers while the tailored versions only have to support a specific customer. The two seemingly contradicting views can come together if we assume that it would be business wise more valuable to increase the configurability of the product version to such an extent that tailored versions become obsolete. The issues regarding configurability still persist and it is difficult to evaluate and measure how it affects monetary value. Configurability is also difficult to evaluate from the technical stand point as it definitely decreases development time in the short term when UI-changes, varying feature use and tweaks to business logic can be easily achieved. However it is very challenging to measure how this short term cost reduction relates to the long-term increase in complexity resulting in high costs of implementing changes outside of the configurable scope and even higher costs in future migrations.

Technically the important aspect for a successful re-engineering effort that also supports future evolvability is determining which factors affect the maintainability and robustness of the system the most. The respondents were unanimous in the view that the highest cause for bugs in currently is lack of testing. Programming mistakes and configuration errors were also seen as contributors to bugs. The lack of testing can partly be attributed to the high amount of configurability as especially end to end inputs and outputs can vary to a high extent which makes testing difficult. Another observation for testing difficulties was lacking testing data which is often attributed to the customer. As data structures and different data models can also often be configured and used differently at different customers it is challenging to efficiently generate realistic testing data in order to validate the wide scope of functionality present.

The configurability in and of itself was also estimated by the respondents to assist in dealing with version control and lacking specifications. The version control of the Lean Sytem source code is based on an outdated system which has great challenges in forking and re-merging features. The configurations remedy this issue to a degree as fewer repositories are needed and especially many smaller changes can be made without having to touch the source code. Customers often don't know exactly what they want or need which makes trying out different functionality in the course of development a necessity. As changes can be made with configuration tweaks instead of programming and delivering new versions the development time is reduced. The respondents also answered how configurability could be better handled and several answers indicated that the documentation of different configuration points was lacking. One answer summed up the configurability issue in more detail: "Configurations are currently not in version management system and they are separate (error prone) deliverable in customer projects (except when in project config files). Configurability should be implemented in small/limited scale (for example hiding some specific fields or modules) but larger customizations like view composition and workflows might be better to handle with alternative methods (pre-defined set of alternative ways, versioning/branching etc.). Performance-wise configuration should probably be applied as a pre-compile/generation step instead of manipulating the view content at runtime".

Technical factors regarding the environment and broader architecture were also considered in the questionnaire. A focus was on current dependency between front and back end, how the dependency should look like in the future (stand alone or not) and the third party library used for back end queries. However these questions were all very divisive and no clear consensus is present. A few answers indicated that striving for a REST API in the future would be ideal but questions were also raised about the broader Roima-level vision for back end and the whole server stack. All in all it can be deducted that a closer look and evaluation focused on the back end and its interfaces is necessary in the future. Therefore the evolution plan for Manufacturing portal can not solve issues related to these aspects but none the less needs to take into account that the current architecture could change in the future.

Lastly the respondents answered questions about React and it became clear that there is a high interest amongst developers towards it even though current familiarity is generally not very high. This is natural as the current stack is focused on AngularJS and .NET, but introduces risks in the re-engineering effort. Developers also estimated that it would be unrealistic to start exclusively using React for all new development as it would slow down development speed too much. An incremental approach is therefore preferred.

6.2 Determining an evolution strategy based on analysis

Based on a closer assessment of Manufacturing Portal different evolution strategies are discussed in this section.

The six different evolution strategies described by the Renaissance method are presented in Table 6 where they are ordered according to increasing cost, benefit and risk. The total replacement of the system is the most expensive and risky but is also capable of bringing the greatest benefit. By comparing the results from the closer level assessment questionnaire of Manufacturing portal with the goals of different evolution strategies and their associated risks we can determine what kind of strategy is most suitable.

If we start narrowing down suitable strategies from the main trigger for re-engineering in this case study which is the need to support React components, we can immediately rule out Continued Maintenance and Revamp. Continued maintenance does obviously not align with new React support and the Revamp strategy where only user interface changes are made is also not suitable as deeper changes have to be made. We can also rule out total replacement of the system as it is the most risky and costly option and results from the questionnaire indicate that full time react

Table 6: Evolution strategies [23]

Continued Maintenance	The accommodation of change in a system, without radical change to its structure, after it has been delivered and deployed
Revamp	The transformation of a system by modifying or replacing its user interfaces. The internal workings of the system remain intact, but the system appears to have changed to the user.
Restructure	The transformation of a system's internal structure without changing any external interfaces.
Rearchitecture	The transformation of a system by migrating it to a different technological architecture.
Redesign with Reuse	The transformation of a system by redeveloping it utilising some legacy system components.
Replace	Total replacement of a system.

development is not feasible at the same time with ongoing customer work. The risks of a total replacement are also further emphasised by partially lacking React skills.

Redesign with Reuse which is described as redeveloping the system utilizing some legacy components can be seen as applicable. This approach has also been used in the on going migration of Travel Expenses Portal from LeanPortal (ASP.NET) to React where some common components are used with LeanPortalSPA (AngularJS). However redesigning the whole Manufacturing Portal is questionable as the questionnaire results displayed in Table 4 indicate that only few high priority re-engineering candidates exist within the portal meaning that the benefits of re-engineering are likely not high enough to warrant the cost.

Rearchitecture which is described as migrating to a different technological architecture can at first seem to be a good fit for the React support requirement but the higher level architecture remains the same as both AngularJS and React are single page JavaScript frameworks. This strategy is mostly associated with larger architectural leaps like for example moving to microservice or cloud architecture. However this comes down to semantics and as React especially in combination with TypeScript and other related framework changes is a big change with many new technologies, Restructure can be argued to be a suitable strategy.

Restructure which is described as transforming internal structure without changing external interfaces can perhaps be seen as most suitable for the new React support. As Travel Expenses Portal is already migrated to React and partly integrated into the AngularJS framework it is likely that only changing high priority candidates like the Texts and Documents component to React could yield good results. Costs and risks are also small when the general framework of the system is unchanged and only certain components are re-engineered. Risks are not completely mitigated however and certain risk factors are still present.

If we look back at Table 2 at page 19 we can identify which factors increase the risk related to the Restructure or Rearchitecture strategies. The first associated factor is lack of system knowledge, which should not be an issue as nearly all original developers and architects of Manufacturing Portal are still available. The second risk factor is poor documentation. This factor is relevant as poor documentation was mentioned by several respondents in the questionnaire. Great care has to be taken to not lose hidden functionality or configurations when existing components are re-engineered. This risk is somewhat mitigated by the fact that there is no lack of knowledge about the current system. The third risk factor is new technology skills shortage. This factor is also relevant as React skills are not currently at a high level across the board. The next factor is legacy technology skills shortage which in this case is not a relevant risk as current AngularJS and .NET skills are at a high level. Errors introduced during evolution is a risk factor that is highly relevant especially in the light of several respondents mentioning lack of testing as a high contributor to bugs. This means that it can be difficult to validate that a re-engineered component works like the old one without new bugs. Lastly the technology immaturity and obsolete operational environment factors can be deemed to be irrelevant in this case.

After analysing the assessment results and determining an evolution strategy the following phase in Renaissance would be to implement the evolution plan and document the assessments into an information repository for future reference. This implementation is not a part of this study and will possibly be completed as a different project at Roima if the Renaissance process is deemed to be a worthwhile tool based on the evaluation.

6.3 Evaluation of Renaissance

This section evaluates the Renaissance process based on experiences from completing the evolution planning phase. The evaluation is done through discussing results from interviews and feedback gathered from the assessment questionnaires.

High level assessment evaluation Feedback related to the high level assessment questionnaire was broadly connected to the scope of the questionnaire. Experts in different roles suggested different focuses from larger business segments to smaller technical parts. This illustrates the flexibility of the Renaissance system as it can be used at different levels of granularity providing relevant and interesting information for experts in different roles. The flexibility is however also a draw back as it can be difficult to determine an appropriate assessment level. As seen in the feedback there was no clear consensus of an optimal scope among respondents.

The assessment level also affects the amount of experts available which can result in massive swings in the results caused by individual opinions. A larger group would have been preferable but was not possible due to the limiting factor of target scope greatly affecting available experts with deep knowledge of the systems in question. While the results of the assessment are not statistically significant they still provide a good starting point for further analysis and reflection. This point is underlined by many feedback comments stating that the questionnaire gave a good structured basis for discussion.

Low overhead is a goal of the Renaissance process and this can be said to be true at least for the assessment phase based on the feedback comments. However, some uncertainty is caused by a low amount of written answers, which could be interpreted as participants not taking the time to deeply think through the questions. If a respondent quickly answers the questionnaire without deep thought it is naturally easy and fast. The questionnaire could perhaps be further improved by making questions requiring written answers unskippable in order to force participants to pause and formulate some kind of answer. However, the relevance of "forced" answers can be questioned as well. It can be argued that keeping the assessment phase as lightweight as possible is desired and a superficial view of the system is sufficient as further analysis is done regardless in the next phase of the Renaissance process. It is still important to keep in mind that the assessment results can not alone be used to make deductions about the system.

Low level Manufacturing portal assessment evaluation Feedback results for the lower level assessment were much in line with the higher level assessment. This is understandable as many of respondents were the same and the questionnaires themselves were constructed in a similar way even though the scope differs. The challenges of constructing questionnaires is still present and one comment in particular highlighted this well: “For many of the questions it was very hard to select an answer from a one-dimensional scale/quantity because the thinking/reasoning process would automatically start to intertwine other dimensions to the question”. This issue could perhaps be mitigated in the future with more specific questions and guiding descriptions and examples appended to the questions. Adding more questions with text answers could also help but then the analysis of the results becomes more time consuming and difficult. This seems like an issue that will persist to some degree in any case and has to be accepted as a limitation of this kind of questionnaire based system assessment. Including some data based metrics could also be considered to complement the questionnaire in the future but as no meaningful metrics could be defined by experts for this small case study it is unlikely. Some potential for relevant metrics was still identified in the planned version control system change, where a more modern solution would enable better tracking of for example high priority re-engineering targets based on how often changes are made to the source code.

The questionnaire does seem to bring some value in and of itself as several respondents commented that thinking about and answering the questions was useful and resulted in some “thinking out of the box”. It could even be beneficial to conduct these kinds of assessments from time to time to simply provide a structured way to examine different parts of the system.

Interviews This section discusses results of interviews conducted with experts evaluating the whole Renaissance process and the assessment questionnaires.

The interviews confirmed previous observations that metrics are difficult to determine and not currently realistically usable. With development the current work hour reporting tool and task management system could in theory provide interesting metrics. The upcoming version control change should also enable better metrics. If the Renaissance process is used in the future at Roima, adding metrics to the assessment phase in addition to expert opinions would further validate the results of the process. Calculating and interpreting metrics would on the other hand add overhead to the process, which can become an issue.

The Renaissance process was deemed to be a potential good addition to the way of work. Value in the process was also recognized in doing the assessments and discussing the results. This was especially highlighted in one respondent's comment: "I like it! Could we get this into everyday work? Things that should have been done have been brought up more systematically and transparently".

Most respondents thought that the Renaissance strategy was easy to understand which is positive. This was not however universal and at least one respondent found the process difficult to grasp. As the sample size was quite small it is possible that the perceived easiness of the strategy would change with more respondents. On the other hand many different roles and people with different skill sets were represented in the group of respondents which makes it likely that the results would be reasonably similar even with larger sample sizes.

The interviews revealed many positive views on the Renaissance strategy. As it is a formal and structured process it "makes sure that different options and aspects are considered" in contrast to a more informal way of handling development so far. This point was also underlined by several other answers that were formulated in a different way. Related to the structured process were also some concerns as a question was raised if the process introduces a risk of limiting thinking to only the options provided by the process. It could however be argued that thinking outside of the box is in no way limited by the process itself and that weighing pros and cons of pre-defined options for re-engineering at least makes sure that several different options have been considered.

Several respondents suggested that in order to be usable in practice the process has to be refined and include a broader group of people. Especially the customer viewpoint was brought up by several respondents as a potential area that would benefit from the process. "The process could help to better understand client needs" and enable better motivating or "pushing" features that are known to be important for Roima.

Renaissance was also thought to be usable for different systems in Roima and especially the incoming re-engineering effort for the desktop version of Lean System was highlighted as a potential use case.

Thoughts on respondents actually using the full process themselves was mixed. One respondent commented that they couldn't see themselves constructing questionnaires like the ones used in this case study. The same respondent was of the opinion that Renaissance would "have to be raised to some kind of strategic role in the organisation" in order to be used in practice as there wouldn't be enough resources available otherwise. Another respondent commented in a similar vein that "someone would have to really think about what is needed in order to use this process". In contrast, several respondents also commented that they could envision themselves being a part of constructing questionnaires and participate in the process in other ways than being respondents to questionnaires. The need for studying the process more closely in order to use the process fully was also brought up by several respondents. Based on the different answers it can be said that additional familiarization and allocation of resources are needed, and a separate deployment plan would be a requirement for taking Renaissance into use fully.

Challenges prioritizing new functionality and improving existing functionality have been recognized and could potentially be remedied with the Renaissance process. Also large challenges in handling configurability and identifying which parts should be static.

A point of some architectural decisions recommended by system architects being overruled from higher up was also brought up. This is an issue that Renaissance also could help with as a clearer relation between business priorities and technical issues is revealed through the system assessments from different view points.

6.4 Implications and lessons learned

This section discusses the observations of the experimental instantiation and the results of the re-engineering strategy evaluation providing answers to RQ3, RQ4 and RQ5.

The results from the lower level assessment provide answers to research question RQ3: How can a complex web application be re-engineered in order to support a new framework? Research question RQ3 is addressed by the Renaissance strategy's incremental evolution plans; restructuring and rearchitecturing. This is motivated by the results of the case study as it can be argued that Manufacturing Portal should be re-engineered to support react incrementally with the restructuring and rearchitecturing evolution plans as described earlier in section 6.2. In practice this means that the current AngularJS application can continue as the main framework providing the base functionalities like routing, credentials, session management etc. React support can be achieved by integration within the current application. Older LeanPortal (ASP.NET) portals can be migrated at a larger scale while existing LeanPortalSPA portals can be upgraded through smaller components. More specifically Manufacturing Portal could be largely left untouched as only the Texts and Documents component was deemed to be a high priority re-engineering target. Further assessments are needed in order to specify what other portals and components would be potential re-engineering candidates.

The results from evaluating Renaissance through the assessment questionnaire feedback and interviews, answer research questions RQ4 and RQ5. Research question RQ4: "Is the re-engineering strategy useful for solving different issues and providing added value?" is answered as follows. Renaissance is useful for different aspects like providing a structured framework of work in order to consider different solutions for re-engineering issues. Renaissance also adds value through additional insight into different aspects of the system merging business and technical views. In the case study interviews the consensus seemed to be that even just thinking about the assessment phase questions was useful and provided good structure for discussions. This is positive and the assessment of business and technical aspects could perhaps even be used separately from time to time without going through the whole Renaissance process each time.

Research question RQ5: "How easy to use is the re-engineering strategy and what resources are needed"? is answered as follows. Renaissance has proven to be a low overhead process, being able to be used with relatively few resources. Renaissance proved to be generally easy to use with a few caveats. Renaissance was viewed by some to be a little abstract and difficult to grasp, but considering that most respondents only got a brief introductory presentation of the strategy itself this can bee seen as natural. The ease of use was further solidified by many respondents being of the opinion that the strategy could be usable in practice, at least after further familiarization.

Software professionals make critical decisions with far reaching consequences alarmingly often based on pure intuition even on large re-engineering projects [24]. The outcome of such decision making obviously varies wildly depending largely on the key decision maker's hidden knowledge and even luck. This worry was also expressed in the interviews and one point expressed this issue well with the statement "massive business, difficult to manage with only a hunch". To reduce this kind of risk and achieve more transparency in decision making a controlled and systematic approach like Renaissance can be used.

The difficulties with the process are that it can be difficult to choose an appropriate assessment level and generate specific enough questions to gather relevant and valuable data. Adding metrics to the assessment would give more certainty of the results being correct and somewhat mitigate the large effect of how the questionnaire is constructed.

The process being flexible is a good thing and was also a key feature that encouraged to use Renaissance. However the flexibility also brings much responsibility as no meaningful results can be achieved without considerable knowledge of the system beforehand. Especially in cases where expert opinions are split and there perhaps is no clear re-engineering trigger, Renaissance does not provide much additional value. On the other hand if no clear cut re-engineering trigger exists there might not be any good reason to begin a re-engineering effort in the first place.

6.5 Limitations

This study has its limitations and can not account for possible new or other existing re-engineering strategies that might be better suited for Roima. A broader systematic mapping study would be needed to identify additional potential re-engineering strategies. This study is also not primarily comparing different strategies and as Renaissance was chosen partly with intuition and a focus on flexibility, the results of a re-engineering effort with a specialized strategy could be better suited for future re-engineering efforts. For example several re-engineering strategies exist that focus solely on cloud migration. It is not clear how a “jack of all trades”-strategy like Renaissance compares to strategies focused on specific problems. Does a generic strategy that is used often for different problems make it more efficient in the end, as stakeholders become familiar with the process, or is it worth it to learn a new process for every specific re-engineering issue? These limitations relate especially to research questions RQ1 and RQ2.

The results for research question RQ3 answering how a complex web application can be re-engineered in order to support a new framework is limited by the context of the study itself. Re-engineering to support React could be done in many different ways, of which a specific evolution strategy determined with the help of Renaissance in this study is only one option. Different kinds of frameworks also have differing requirements and would affect the answer of this question.

The study was also limited by time constraints and a smallish respondent group for the assessments and interviews. A larger group would have given more confidence in the validity of the results for research questions RQ4 and RQ5. Furthermore, the study did not deeply focus on how replicable the results are. By completing the same assessments with different groups of respondents and comparing the results this could have been addressed. Another limitation related to the evaluation of the strategy was that a recent organisational shift towards using SAFe (Scaled Agile Framework) [25] and it's model for Lean Portfolio Management (LPM) was not included in the scope of this study. The SAFe LPM framework, while not being a re-engineering strategy, has some confluence with Renaissance on an abstract level, where product development is analysed and assessed through different business aspects in order to set and prioritize tasks. Future research could investigate how Renaissance would work as a part of the larger SAFe LPM framework and discover potential synergies.

However, the limitations should not prevent generalizing the results and the study did not recognize any reasons why the results would not be applicable elsewhere. Renaissance can be concluded to be useful, lightweight and easy to use also for other small organisations working with complex legacy software systems.

7 Conclusions

Many different kinds of re-engineering strategies exist but a flexible and generic one is well suited for especially smaller organizations with limited resources dealing with varying and complex legacy systems. The Renaissance re-engineering strategy provides such an approach and has been shown by this study to be lightweight and flexible providing a process for managing and motivating decisions made when new unsupported requirements, architectural changes or migrations have to be tackled.

Prioritizing different parts of the system based on a combination of business and technical aspects has also been deemed valuable by software experts participating in the study. A process like Renaissance has also been shown by this study to have some intrinsic value through motivating structured ways of viewing and evaluating the system, providing good starting points for discussions and deeper understanding.

After completing the case study, where a part of the Renaissance process was applied at Roima, the organisation now has a better understanding of how future modernization needs could be better handled. Lessons learned from the case study were that assessing the system through questionnaires works well as a starting point but metrics would assist in validating the interpretations.

The interviews in the study indicated that Renaissance could also be used in other departments and systems in addition to the one examined in the case study, underlining the suitability of Renaissance for varying source and target systems. The study projects that the results are applicable for other organisations with similar characteristics to Roima, i.e., smaller companies developing complex software systems including well established legacy systems.

The most important limitations of this study includes the fact that the amount of different re-engineering strategies examined for the case study was small. Future studies would benefit from a broad systematic mapping study in order to discover other potential re-engineering strategies applicable for small organisations. Another limitation was a relatively small amount of respondents participating in the process and it's evaluation.

Based on findings of this study an aspect of Renaissance itself that would benefit of continued research in the future, is determining the scope of the system assessment level in an optimal way. Finally, comparing the effectiveness of re-engineering strategies focusing on solving specific modernization needs and re-engineering strategies that are generic in nature would be a good path for future research.

References

1 K. Bennett, "Legacy systems: coping with success," *IEEE Software*, vol. 12, no. 1, pp. 19–23, Jan 1995.

2 J. Bisbal, D. Lawless, , and J. Grimson, "Legacy information systems: issues and directions," *IEEE Software*, vol. 16, no. 5, pp. 103–111, Sep. 1999.

3 R. Khadka, B. V. Batlajery, A. M. Saeidi, S. Jansen, and J. Hage, "How do professionals perceive legacy systems and software modernization?" in *Proceedings of the 36th International Conference on Software Engineering.* ACM, 2014, pp. 36–47.

4 P. Kruchten, R. L. Nord, and I. Ozkaya, "Technical debt: From metaphor to theory and practice," *IEEE Software*, vol. 29, no. 6, pp. 18–21, Nov 2012.

5 E. J. Chikofsky and J. H. Cross, "Reverse engineering and design recovery: a taxonomy," *IEEE Software*, vol. 7, no. 1, pp. 13–17, Jan 1990.

6 C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "Devops," *Ieee Software*, vol. 33, no. 3, pp. 94–100, 2016.

7 T. Mens and T. Tourwe, “A survey of software refactoring,” *IEEE Transactions on Software Engineering*, vol. 30, no. 2, pp. 126–139, Feb 2004.

8 B. Althani and S. Khaddaj, “Systematic review of legacy system migration,” in *2017 16th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES)*, Oct 2017, pp. 154–157.

9 R. Khare and R. N. Taylor, “Extending the representational state transfer (rest) architectural style for decentralized systems,” in *Proceedings. 26th International Conference on Software Engineering.* IEEE, 2004, pp. 428–437.

10 “Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation),” *Official Journal of the European Union*, vol. L119, pp. 1–88, April 2016.

11 J. Bisbal, D. Lawless, Bing Wu, J. Grimson, V. Wade, R. Richardson, and D. O’Sullivan, “An overview of legacy information system migration,” in *Proceedings of Joint 4th International Computer Science Conference and 4th Asia Pacific Software Engineering Conference*, 1997, pp. 529–530.

12 V. Khusidman and W. Ulrich, “Architecture-driven modernization: Transforming the enterprise,” in *Seminar Software Analyse and Trasformation*, 2007, p. 7.

13 J. Ransom, I. Sommerville, and I. Warren, “A method for assessing legacy systems for evolution,” in *csmr.* IEEE, 1998, p. 128.

14 R. Rai, G. Sahoo, and S. Mehfuz, “Exploring the factors influencing the cloud computing adoption: a systematic study on cloud migration,” *SpringerPlus*, vol. 4, no. 1, p. 197, 2015.

15 R. H. Von Alan, S. T. March, J. Park, and S. Ram, “Design science in information systems research,” *MIS quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

16 E. Stroulia, M. El-Ramly, and P. Sorenson, “From legacy to web through interaction modeling,” in *International Conference on Software Maintenance, 2002. Proceedings.*, 2002, pp. 320–329.

17 J. Lavery, C. Boldyreff, B. Ling, and C. Allison, “Modelling the evolution of legacy systems to web-based systems,” *Journal of Software: Evolution and Process*, vol. 16, no. 1-2, pp. 5–30, 2004.

18 A. De Lucia, R. Francese, G. Scanniello, and G. Tortora, "Developing legacy system migration methods and tools for technology transfer," *Software: Practice and Experience*, vol. 38, no. 13, pp. 1333–1364, 2008.

19 A. Sanchez, N. Oliveira, L. S. Barbosa, and P. Henriques, "A perspective on architectural re-engineering," *Science of Computer Programming*, vol. 98, pp. 764–784, 2015.

20 A. Mesbah and A. van Deursen, "Migrating multi-page web applications to single-page ajax interfaces," in *11th European Conference on Software Maintenance and Reengineering (CSMR'07)*, March 2007, pp. 181–190.

21 A. Bianchi, D. Caivano, V. Marengo, and G. Visaggio, "Iterative reengineering of legacy systems," *IEEE Transactions on Software Engineering*, vol. 29, no. 3, pp. 225–241, March 2003.

22 H. M. Sneed, "Planning the reengineering of legacy systems," *IEEE software*, vol. 12, no. 1, pp. 24–34, 1995.

23 I. Warren and J. Ransom, "Renaissance: a method to support software system evolution," in *Computer Software and Applications Conference, 2002. COMPSAC 2002. Proceedings. 26th Annual International.* IEEE, 2002, pp. 415–420.

24 M.-M. Saarelainen, J. J. Ahonen, H. Lintinen, J. Koskinen, I. Kankaanpää, H. Sivula, P. Juutilainen, and T. Tilus, "Software modernization and replacement decision making in industry: A qualitative study," in *Proceedings of the 10th international conference on Evaluation and Assessment in Software Engineering.* BCS Learning & Development Ltd., 2006, pp. 12–21.

25 D. Leffingwell, *SAFe 4.5 Reference Guide: Scaled Agile Framework for Lean Enterprises.* Addison-Wesley Professional, 2018.

26 U. Hohenstein and P. Koka, "Enabling legacy applications for multi-tenancy without reengineering," pp. 284–308, 2016.

27 S. Jain and I. Chana, "Modernization of legacy systems: A generalised roadmap," in *Proceedings of the Sixth International Conference on Computer and Communication Technology 2015.* ACM, 2015, pp. 62–67.

28 E. J. Chikofsky and J. H. Cross, "Reverse engineering and design recovery: a taxonomy," *IEEE Software*, vol. 7, no. 1, pp. 13–17, Jan 1990.

29 S. Demeyer, S. Ducasse, and O. Nierstrasz, *Object-oriented reengineering patterns.* Elsevier, 2002.

30 H. C. Benestad, B. Anda, and E. Arisholm, "Understanding cost drivers of software evolution: a quantitative and qualitative investigation of change effort in two evolving software systems," *Empirical Software Engineering*, vol. 15, no. 2, pp. 166–203, 2010.

31 N. Brown, Y. Cai, Y. Guo, R. Kazman, M. Kim, P. Kruchten, E. Lim, A. MacCormack, R. Nord, I. Ozkaya, R. Sangwan, C. Seaman, K. Sullivan, and N. Zazworka, "Managing technical debt in software-reliant systems," in *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, ser. FoSER '10. New York, NY, USA: ACM, 2010, pp. 47–52.

Appendix 1. System Assesment Questionnaire First Round

System assessment questionnaire

Do not submit any personal or identifying data.

* Required

Description and goal of the questionnaire

In essence the point is to systematically map the current system portfolio and modernization needs. The questionnaire aims to follow the Renaissance method, a process for managing legacy system modernization. The goal of the questionnaire is to create a baseline understanding of different parts of the system. The assessment results will function as a starting point for creating an evolution plan.

The questionnaire is divided into three categories (Business quality, Technical quality and Organisational factors). The scope of this assessment is limited to three portals (Travel Expenses, Service, Manufacturing).

Try to answer all questions. If you do not have any opinion or knowledge about a question it can be left blank.

Don't get hung up on details, the idea is to create a general understanding of the system to support further analysis.

Role in the organisation

1. **Briefly describe your role** *

Business quality (Travel Expenses Portal)

Answer the questions based on the older LeanPortal version without taking into account the new React version which is in development.

2. **Is the system important to the organisation?**
Mark only one oval.

	1	2	3	4	
Not important	⬭	⬭	⬭	⬭	Critical

3. **Is the system tailored for different customers?**
Mark only one oval.

	1	2	3	4	
Completely standard	⬭	⬭	⬭	⬭	Everything is customized

4. **Estimate the monetary value of the system.**
Mark only one oval.

	1	2	3	4	
Low	⬭	⬭	⬭	⬭	High

5. **Any other business perspectives?**

Business quality (Service portal)

Answer the questions based on the LeanPortal functionality.

6. **Is the system important to the organisation?**
Mark only one oval.

	1	2	3	4	
Not important	⬭	⬭	⬭	⬭	Critical

7. **Is the system tailored for different customers?**
Mark only one oval.

	1	2	3	4	
Completely standard	⬭	⬭	⬭	⬭	Everything is customized

8. **Estimate the monetary value of the system.**
Mark only one oval.

	1	2	3	4	
Low	⬭	⬭	⬭	⬭	High

9. **Any other business perspectives?**

Business quality (Manufacturing Portal)

Answer the questions based on the LeanPortalSPA functionality.

10. **Is the system important to the organisation?**
Mark only one oval.

	1	2	3	4	
Not important	⬭	⬭	⬭	⬭	Critical

11. **Is the system tailored for different customers?**
Mark only one oval.

	1	2	3	4	
Completely standard	⬭	⬭	⬭	⬭	Everything is customized

12. **Estimate the monetary value of the system.**
Mark only one oval.

	1	2	3	4	
Low	⬭	⬭	⬭	⬭	High

13. **Any other business perspectives?**

Technical quality (Travel Expenses Portal)

Answer the questions based on the older LeanPortal version without taking into account the new React version which is in development.

14. **Is the system complex?**
Mark only one oval.

	1	2	3	4	
Simple	⬭	⬭	⬭	⬭	Complex

15. **How large is the system?**
Mark only one oval.

	1	2	3	4	
Small	⬭	⬭	⬭	⬭	Big

16. **Is the system error prone?**
Mark only one oval.

	1	2	3	4	
Error prone	⬭	⬭	⬭	⬭	Robust

17. **Is the system difficult to maintain?**
Mark only one oval.

	1	2	3	4	
Difficult	⬭	⬭	⬭	⬭	Easy

18. **Does the system have good performance?**
Mark only one oval.

	1	2	3	4	
Low performance	⬭	⬭	⬭	⬭	High performance

19. **Any other technical aspects?**

Technical quality (Service portal)

Answer the questions based on the LeanPortal functionality.

20. **Is the system complex?**
Mark only one oval.

	1	2	3	4	
Simple	⬭	⬭	⬭	⬭	Complex

21. **How large is the system?**
Mark only one oval.

	1	2	3	4	
Small	⬭	⬭	⬭	⬭	Big

22. **Is the system error prone?**
Mark only one oval.

	1	2	3	4	
Error prone	⬭	⬭	⬭	⬭	Robust

23. **Is the system difficult to maintain?**
Mark only one oval.

	1	2	3	4	
Difficult	⬭	⬭	⬭	⬭	Easy

24. **Does the system have good performance?**
Mark only one oval.

	1	2	3	4	
Low performance	⬭	⬭	⬭	⬭	High performance

25. **Any other technical aspects?**

Technical quality (Manufacturing portal)

Answer the questions based on the LeanPortalSPA functionality.

26. **Is the system complex?**
Mark only one oval.

	1	2	3	4	
Simple	⬭	⬭	⬭	⬭	Complex

27. **How large is the system?**
Mark only one oval.

	1	2	3	4	
Small	⬭	⬭	⬭	⬭	Big

28. **Is the system error prone?**
Mark only one oval.

	1	2	3	4	
Error prone	⬭	⬭	⬭	⬭	Robust

29. **Is the system difficult to maintain?**
Mark only one oval.

	1	2	3	4	
Difficult	⬭	⬭	⬭	⬭	Easy

30. **Does the system have good performance?**
Mark only one oval.

	1	2	3	4	
Low performance	⬭	⬭	⬭	⬭	High performance

31. **Any other technical aspects?**

Organisational factors (Travel Expenses Portal)

Answer the questions based on the older LeanPortal version without taking into account the new React version which is in development.

32. **Estimate acceptance to change**
Mark only one oval.

	1	2	3	4	
Resistant	⬭	⬭	⬭	⬭	Accepting

33. **How high is the work load for system experts?**
Mark only one oval.

	1	2	3	4	
Low	⬭	⬭	⬭	⬭	High

34. **Estimate the availability of experienced experts**
Mark only one oval.

	1	2	3	4	
Low	⬭	⬭	⬭	⬭	High

35. **Any other organisational aspects?**

Organisational factors (Service portal)

Answer the questions based on the LeanPortal functionality.

36. **Estimate acceptance to change**
Mark only one oval.

	1	2	3	4	
Resistant	⬭	⬭	⬭	⬭	Accepting

37. **How high is the work load for system experts?**
Mark only one oval.

	1	2	3	4	
Low	⬭	⬭	⬭	⬭	High

38. **Estimate the availability of experienced experts**
Mark only one oval.

	1	2	3	4	
Low	⬭	⬭	⬭	⬭	High

39. **Any other organisational aspects?**

Organisational factors (Manufacturing portal)

Answer the questions based on the LeanPortalSPA functionality.

40. **Estimate acceptance to change**
Mark only one oval.

	1	2	3	4	
Resistant	◯	◯	◯	◯	Accepting

41. **How high is the work load for system experts?**
Mark only one oval.

	1	2	3	4	
Low	◯	◯	◯	◯	High

42. **Estimate the availability of experienced experts**
Mark only one oval.

	1	2	3	4	
Low	◯	◯	◯	◯	High

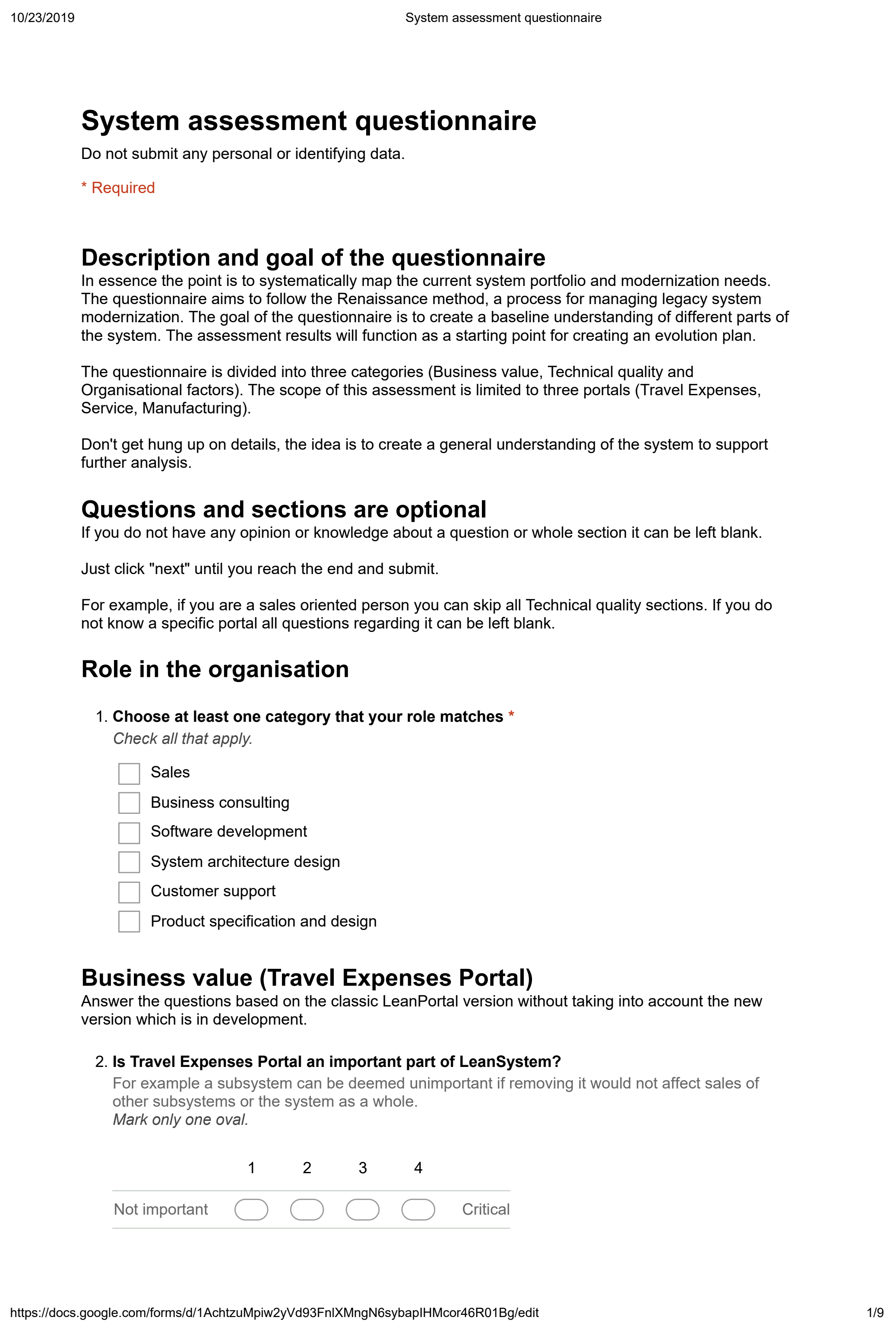
43. **Any other organisational aspects?**

Feedback

44. **Did you find thinking about and answering the questions useful?**

45. **Any general comments or thoughts?**

Appendix 2. System Assesment Questionnaire Second Round



System assessment questionnaire

Do not submit any personal or identifying data.

* Required

Description and goal of the questionnaire

In essence the point is to systematically map the current system portfolio and modernization needs. The questionnaire aims to follow the Renaissance method, a process for managing legacy system modernization. The goal of the questionnaire is to create a baseline understanding of different parts of the system. The assessment results will function as a starting point for creating an evolution plan.

The questionnaire is divided into three categories (Business value, Technical quality and Organisational factors). The scope of this assessment is limited to three portals (Travel Expenses, Service, Manufacturing).

Don't get hung up on details, the idea is to create a general understanding of the system to support further analysis.

Questions and sections are optional

If you do not have any opinion or knowledge about a question or whole section it can be left blank.

Just click "next" until you reach the end and submit.

For example, if you are a sales oriented person you can skip all Technical quality sections. If you do not know a specific portal all questions regarding it can be left blank.

Role in the organisation

1. **Choose at least one category that your role matches** *

   Check all that apply.

   ☐ Sales
   ☐ Business consulting
   ☐ Software development
   ☐ System architecture design
   ☐ Customer support
   ☐ Product specification and design

Business value (Travel Expenses Portal)

Answer the questions based on the classic LeanPortal version without taking into account the new version which is in development.

2. **Is Travel Expenses Portal an important part of LeanSystem?**

   For example a subsystem can be deemed unimportant if removing it would not affect sales of other subsystems or the system as a whole.

   Mark only one oval.

	1	2	3	4	
Not important	◯	◯	◯	◯	Critical

3. **Is Travel Expenses Portal tailored for different customers?**
Mark only one oval.

	1	2	3	4	
Completely standard	◯	◯	◯	◯	Everything is customized

4. **Estimate the monetary value of Travel Expenses Portal.**
Take into account both direct and indirect value.
Mark only one oval.

	1	2	3	4	
Low	◯	◯	◯	◯	High

5. **Are there any other business perspectives that are important for understanding Travel Expenses Portal?**
For example the organisation having a new target market, new future business requirements or the system being a loss leader ("Sisäänheittotuote"). Changes in the market/competition.

Business value (Service Portal)

Answer the questions based on the classic LeanPortal functionality.

6. **Is Service Portal an important part of LeanSystem?**
For example a subsystem can be deemed unimportant if removing it would not affect sales of other subsystems or the system as a whole.
Mark only one oval.

	1	2	3	4	
Not important	◯	◯	◯	◯	Critical

7. **Is Service Portal tailored for different customers?**
Mark only one oval.

	1	2	3	4	
Completely standard	◯	◯	◯	◯	Everything is customized

8. **Estimate the monetary value of Service Portal.**
Take into account both direct and indirect value.
Mark only one oval.

	1	2	3	4	
Low	◯	◯	◯	◯	High

9. **Are there any other business perspectives that are important for understanding Service Portal?**
For example the organisation having a new target market, new future business requirements or the system being a loss leader ("Sisäänheittotuote"). Changes in the market/competition.

Business value (Manufacturing Portal)

Answer the questions based on the LeanPortalSPA functionality.

10. **Is Manufacturing Portal an important part of LeanSystem?**
For example a subsystem can be deemed unimportant if removing it would not affect sales of other subsystems or the system as a whole.
Mark only one oval.

	1	2	3	4	
Not important	◯	◯	◯	◯	Critical

11. **Is Manufacturing Portal tailored for different customers?**
Mark only one oval.

	1	2	3	4	
Completely standard	◯	◯	◯	◯	Everything is customized

12. **Estimate the monetary value of Manufacturing Portal.**
Take into account both direct and indirect value.
Mark only one oval.

	1	2	3	4	
Low	◯	◯	◯	◯	High

13. **Are there any other business perspectives that are important for understanding Manufacturing Portal?**
For example the organisation having a new target market, new future business requirements or the system being a loss leader ("Sisäänheittotuote"). Changes in the market/competition.

Technical quality (Travel Expenses Portal)

Answer the questions based on the older LeanPortal version without taking into account the new React version which is in development.

14. **Is Travel Expenses Portal complex?**
Amount of functionality and configurability.
Mark only one oval.

	1	2	3	4	
Simple	⬭	⬭	⬭	⬭	Complex

15. **How large a part of LeanSystem is Travel Expenses Portal?**
Relative to all other portals.
Mark only one oval.

	1	2	3	4	
Small	⬭	⬭	⬭	⬭	Big

16. **Is Travel Expenses Portal error prone?**
Mark only one oval.

	1	2	3	4	
Error prone	⬭	⬭	⬭	⬭	Robust

17. **Is Travel Expenses Portal difficult to maintain?**
Effort required to implement changes or new features.
Mark only one oval.

	1	2	3	4	
Difficult	⬭	⬭	⬭	⬭	Easy

18. **Does Travel Expenses Portal have good performance?**
Mark only one oval.

	1	2	3	4	
Low performance	⬭	⬭	⬭	⬭	High performance

19. **Are there any other technical aspects that are important for understanding Travel Expenses Portal?**
For example end of life frameworks, third party components or hardware issues.

Technical quality (Service portal)

Answer the questions based on the LeanPortal functionality.

20. **Is Service Portal complex?**
Amount of functionality and configurability.
Mark only one oval.

	1	2	3	4	
Simple	⬭	⬭	⬭	⬭	Complex

21. **How large a part of LeanSystem is Service Portal?**
Relative to other portals.
Mark only one oval.

	1	2	3	4	
Small	⬭	⬭	⬭	⬭	Big

22. **Is Service Portal error prone?**
Mark only one oval.

	1	2	3	4	
Error prone	⬭	⬭	⬭	⬭	Robust

23. **Is Service Portal difficult to maintain?**
Effort required to implement changes or new features.
Mark only one oval.

	1	2	3	4	
Difficult	⬭	⬭	⬭	⬭	Easy

24. **Does Service Portal have good performance?**
Mark only one oval.

	1	2	3	4	
Low performance	⬭	⬭	⬭	⬭	High performance

25. **Are there any other technical aspects that are important for understanding Service Portal?**
For example end of life frameworks, third party components or hardware issues.

Technical quality (Manufacturing portal)

Answer the questions based on the LeanPortalSPA functionality.

26. **Is Manufacturing Portal complex?**
Amount of functionality and configurability.
Mark only one oval.

	1	2	3	4	
Simple	⬭	⬭	⬭	⬭	Complex

27. **How large a part of LeanSystem is Manufacturing Portal?**
Relative to other portals.
Mark only one oval.

	1	2	3	4	
Small	⬭	⬭	⬭	⬭	Big

28. **Is Manufacturing Portal error prone?**
Mark only one oval.

	1	2	3	4	
Error prone	◯	◯	◯	◯	Robust

29. **Is Manufacturing Portal difficult to maintain?**
Effort required to implement changes or new features.
Mark only one oval.

	1	2	3	4	
Difficult	◯	◯	◯	◯	Easy

30. **Does Manufacturing Portal have good performance?**
Mark only one oval.

	1	2	3	4	
Low performance	◯	◯	◯	◯	High performance

31. **Are there any other technical aspects that are important for understanding Service Portal?**
For example end of life frameworks, third party components or hardware issues.

Organisational factors (Travel Expenses Portal)

Answer the questions based on the older LeanPortal version without taking into account the new React version which is in development.

32. **Estimate acceptance to change among experts working with Travel Expenses Portal.**
Interest to adapt new technologies, change work processes etc.
Mark only one oval.

	1	2	3	4	
Resistant	◯	◯	◯	◯	Accepting

33. **How high is the work load for experts working with Travel Expenses Portal?**
Especially customer work.
Mark only one oval.

	1	2	3	4	
Low	◯	◯	◯	◯	High

34. **Estimate the availability of technical experts who have experience working with Travel Expenses Portal.**
Especially experts with deep understanding of the architecture and code structure.
Mark only one oval.

	1	2	3	4	
Low	⬭	⬭	⬭	⬭	High

35. **Estimate the availability of business specialists who have experience working with Travel Expenses Portal.**
Especially specialists with deep understanding of the business logic.
Mark only one oval.

	1	2	3	4	
Low	⬭	⬭	⬭	⬭	High

36. **Are there any other organisational aspects that affect Travel Expenses Portal?**
Mergers and acquisitions, recruitment challenges, retirements etc.

Organisational factors (Service portal)

Answer the questions based on the LeanPortal functionality.

37. **Estimate acceptance to change among experts working with Service Portal.**
Interest to adapt new technologies, change work processes etc.
Mark only one oval.

	1	2	3	4	
Resistant	⬭	⬭	⬭	⬭	Accepting

38. **How high is the work load for experts working with Service Portal?**
Especially customer work.
Mark only one oval.

	1	2	3	4	
Low	⬭	⬭	⬭	⬭	High

39. **Estimate the availability of technical experts who have experience working with Service Portal.**
Especially experts with deep understanding of the architecture and code structure.
Mark only one oval.

	1	2	3	4	
Low	⬭	⬭	⬭	⬭	High

40. **Estimate the availability of business specialists who have experience working with Service Portal.**
Especially specialists with deep understanding of the business logic.
Mark only one oval.

	1	2	3	4	
Low	⬭	⬭	⬭	⬭	High

41. **Are there any other organisational aspects that affect Service Portal?**
Mergers and acquisitions, recruitment challenges, retirements etc.

____

Organisational factors (Manufacturing portal)

Answer the questions based on the LeanPortalSPA functionality.

42. **Estimate acceptance to change among experts working with Manufacturing Portal.**
Interest to adapt new technologies, change work processes etc.
Mark only one oval.

	1	2	3	4	
Resistant	⬭	⬭	⬭	⬭	Accepting

43. **How high is the work load for experts working with Manufacturing Portal?**
Especially customer work.
Mark only one oval.

	1	2	3	4	
Low	⬭	⬭	⬭	⬭	High

44. **Estimate the availability of technical experts who have experience working with Manufacturing Portal.**
Especially experts with deep understanding of the architecture and code structure.
Mark only one oval.

	1	2	3	4	
Low	⬭	⬭	⬭	⬭	High

45. **Estimate the availability of business specialists who have experience working with Manufacturing Portal.**
Especially specialists with deep understanding of the business logic.
Mark only one oval.

	1	2	3	4	
Low	⬭	⬭	⬭	⬭	High

46. **Are there any other organisational aspects that affect Manufacturing Portal?**
Mergers and acquisitions, recruitment challenges, retirements etc.

Feedback

47. **Did you find thinking about and answering the questions useful, in what way?**

48. **Any general comments or thoughts about the questionnaire?**

Powered by
Google Forms

Appendix 3. Manufacturing Questionnaire

Manufacturing portal assessment questionnaire

Do not submit any personal or identifying data.

* Required

Description and goal of the questionnaire

In essence the point is to systematically map different components of Manufacturing portal in order to prioritize future development. The questionnaire aims to follow the Renaissance method, a process for managing legacy system modernization. The goal of the questionnaire is to create a deeper understanding of the main components of the assessed system. The assessment results will function as a starting point for creating an evolution plan.

The questionnaire is divided into three categories (Business value, Technical quality and Organisational factors). The scope of this assessment is limited to the Manufacturing portal.

Questions and sections are optional

If you do not have any opinion or knowledge about a question or whole section it can be left blank.

Just click "next" until you reach the end and submit.

For example, if you are a sales oriented person you can skip all Technical sections.

Role in the organisation

1. **Choose at least one category that your role matches** *

   Check all that apply.

   - ☐ Sales
   - ☐ Business consulting
   - ☐ Software development
   - ☐ System architecture design
   - ☐ Customer support
   - ☐ Product specification and design

Business aspects I

The following questions are mainly focused on the LeanPortalSPA Resource task list view (kuormitusryhmän työlista) and some related components in . Some general questions regarding the whole Manufacturing portal are also included.

2. **What is an important attribute customers consider when buying Manufacturing portal?**

   Mark only one oval per row.

	Small factor	Neutral factor	Large factor
Usability	○	○	○
Functionality	○	○	○
Configurability	○	○	○
Amount of features	○	○	○

3. **What is the main need that is solved for a customer using Manufacturing portal?**

4. **Rate the amount of customer tailoring in the different components of Manufacturing portal.**
Consider the latest standard product version.
Mark only one oval per row.

	Low customization	Medium customization	High customization	Very high customization
Task cards				
Quality failures				
Serial numbers				
Materials				
Texts and documents				

5. **Estimate what causes most customer tailoring needs for Manufacturing portal?**
Mark only one oval per row.

	Small cause	Normal cause	Large cause
UI-changes			
Using different features			
Customizing functionality			
Exotic business processes			

6. **Estimate where more monetary value has been generated; selling the standard product version or selling the tailor made versions of Manufacturing portal?**
Cases like Sabriscan/Raumaster vs. Moventas/Sandvik.
Mark only one oval.

	1	2	3	4	5	
Standard product versions.						Tailor made versions

7. **Does Manufacturing portal need less or more configurability to generate value in the future?**
Mark only one oval.

	1	2	3	4	5	
Less						More

Business aspects II

The following questions are mainly focused on the LeanPortalSPA Resource task list view (kuormitusryhmän työlista) and some related components. Some general questions regarding the whole Manufacturing portal are also included.

8. **Rate the importance of the following components for customers.**
Take into account all functionality related to the component.
Mark only one oval per row.

	Low importance	Medium importance	High importance	Very high importance
Task cards				
Quality failures				
Serial numbers				
Materials				
Texts and documents				

9. **Evaluate what the task card object should be in the future in terms of catering to customer needs.**
Currently a lot of functionality in Manufacturing portal is based on the chosen task card which can contain a resource task (wor operation), checklist or work element. Should all use cases be supported in the future?
Mark only one oval per row.

	Deprecated in the future.	Normal use case	Should be prioritized
Resource tasks			
Checklists			
Work elements			

10. **Is there an unimplemented feature that would greatly improve Manufacturing portal?**

Technical aspects (Structural)

The following questions are mainly focused on the Resource task list view (kuormitusryhmän työlista) and some related components. Some general questions regarding the whole Manufacturing portal are also included.

11. **Which component requires most maintenance?**
Most changes or new features.
Mark only one oval per row.

	Low maintenance	Medium maintenance	High maintenance	Very High maintenance
Task cards				
Quality failures				
Serial numbers				
Materials				
Texts and documents				

12. **Which component is most error prone?**
Mark only one oval per row.

	Robust	Mostly robust	Error prone	Very error prone
Task cards				
Quality failures				
Serial numbers				
Materials				
Texts and documents				

13. **Estimate how much following aspects contribute to bugs**
Mark only one oval per row.

	Small contribution	Medium contribution	High contribution
Programming mistakes			
Configuration errors			
Lack of testing			

14. **Rate the complexity of the following components.**
Mark only one oval per row.

	Low complexity	Medium complexity	High complexity	Very high complexity
Task cards	⬭	⬭	⬭	⬭
Quality failures	⬭	⬭	⬭	⬭
Serial numbers	⬭	⬭	⬭	⬭
Materials	⬭	⬭	⬭	⬭
Texts and documents	⬭	⬭	⬭	⬭

15. **How much complexity is caused by configurability?**
Handling of different form settings, enumerations, web.config values etc.
Mark only one oval.

	1	2	3	4	5	
No extra complexity	⬭	⬭	⬭	⬭	⬭	Most complexity is caused by configurability

16. **Rate the performance of each component**
Mark only one oval per row.

	Low performance	Medium performance	High performance	Very high performance
Task cards	⬭	⬭	⬭	⬭
Quality failures	⬭	⬭	⬭	⬭
Serial numbers	⬭	⬭	⬭	⬭
Materials	⬭	⬭	⬭	⬭
Texts and documents	⬭	⬭	⬭	⬭

17. **Estimate what affects performance more; configuration overhead or the framework?**
Mark only one oval.

	1	2	3	4	5	
Configuration overhead	⬭	⬭	⬭	⬭	⬭	Framework

18. **Does configurability help with issues related to following aspects of development?**
Mark only one oval per row.

	No effect	Makes it easier	Critical
Version control	⬭	⬭	⬭
Lacking specifications	⬭	⬭	⬭
Installation and delivery	⬭	⬭	⬭

19. **How could the configurability be better handled?**

Technical aspects (Environmental)

20. **How much dependency is there between the front and back end?**
Changes in front end causing changes in back end and vice versa.
Mark only one oval.

	1	2	3	4	5	
Low dependency	◯	◯	◯	◯	◯	High dependency

21. **Should the goal be to achieve a completely stand alone front end in the future?**
Mark only one oval.

◯ No
◯ Yes
◯ Maybe

22. **Does Breeze bring important functionality related to data queries made in Manufacturing?**
Mark only one oval.

	1	2	3	4	5	
Not Important	◯	◯	◯	◯	◯	Very important

23. **What do you think should be used in an ideal scenario for data queries?**
Continue with Breeze, move to new hot libraries like GraphQL? Create a REST-api, something else?

Organisational factors (React migration)

This section is intended for developers.

24. **How familiar are you with React?**
Mark only one oval.

	1	2	3	4	5	
No idea	◯	◯	◯	◯	◯	I'm a React guru

25. **Are you interested in React?**
Mark only one oval.

	1	2	3	4	5	
Not interested	◯	◯	◯	◯	◯	Very interested

26. **Taking into account normal ongoing customer work load, would it be realistic for you to start exclusively using React for all new development?**
New development in Devel version.
Mark only one oval.

	1	2	3	4	5	
Not enough time	◯	◯	◯	◯	◯	Totally doable

Feedback

27. **Did you find thinking about and answering the questions useful, in what way?**

28. **Any general comments or thoughts about the questionnaire?**