Modifying software systems to improve user adoption

FE Marx

orcid.org/ 0000-0002-7192-8047

Dissertation accepted in fulfilment of the requirements for the degree *Master of Engineering in Computer and Electronic Engineering* at the North West University

Supervisor: Dr JC Vosloo

Graduation: December 2021

Student number: 25880527

ACKNOWLEDGEMENTS

First of all, I would like to thank my fianceé Simoné Jansen van Vuuren for your love, support, encouragement and all of the cups of coffee you made me. I could not have done this without you.

I would also like to thank my mentor Sybrand van Niekerk and my friend Stephan Taljaard for all the help and words of encouragement. Your guidance and support is much appreciated.

To my study leader Jan Vosloo, thank you for your valuable input and guidance throughout this academic year.

I would also like to thank my parents Nico and Liesbeth Marx for giving me the opportunity to complete my engineering degree and now this study. Your love and support is much appreciated.

To my friend Coenraad Mouton, I would like to thank you for all of the late night conversations and discussions that helped me to stay focused on the end goal of completing my master's degree.

Lastly, I would like to thank Prof Eddie Matthews and ETA Operations (Pty) Ltd for providing me with the opportunity and support to complete this study.

Title: Modifying software systems to improve user adoption

Student: Frans Marx

Supervisor: Dr Jan Vosloo

Keywords: software, user adoption, technology acceptance model, reporting system.

ABSTRACT

Industry 4.0 has driven the digitisation of large scale industrial and mining operations. This introduced the burden of large volumes of data to process. Data from industrial systems is usually processed by specialised systems and software. Numerous software systems exist to process this data and generate reports; however, users may be biased towards working with inefficient software. This leads to the problem of purpose-built software being underused.

This study aims to provide a methodology to increase user adoption of software by addressing the factors affecting user acceptance of software released in a production environment. The Technology Acceptance Model (TAM) states that the Perceived Usefulness (PU) and Perceived Ease of Use (PEOU) are the main factors affecting user acceptance. Existing studies were investigated, and it was found that few studies use TAM to improve software adoption. There is also no general methodology to obtain information about the current PU and PEOU of a system.

Secondary factors regarding these two main factors were researched through literature. A methodology was developed to investigate the state of these secondary factors which uses a scoring and evaluation system. This relies on existing users to give feedback by the means of a questionnaire regarding underused systems in their current state. The factors from the ISO 25010 standard for software quality were selected as the basis for the questionnaires. Key factors are selected from the results and improvements to each factor are suggested. The system is evaluated again after the improvements are completed.

The proposed methodology was applied on a case study of an industrial reporting system. The baseline for the system performance was obtained through feedback from existing users of the system using the questionnaire that was created. Most of the factors received a good score, however the low usability score was a clear outlier.

Using the proposed improvement methods, a new user interface was developed in order to increase usability. The new user interface was a web-based system implemented using Microsoft ASP .NET MVC, and proper access control to the system was implemented. Previously, report setup was done via Microsoft Excel. Reliability increased, as the new user interface helps users to provide the correct input by giving the user selectable values or defining what type of value is expected, thereby reducing the number of errors experienced by the system.

The updated system was released to the users. After a sufficient amount of time, the user acceptance rate was then compared with the initial user acceptance rate. The amount of users increased dramatically after the new user interface was released and the overall user acceptance increased from 30% to 70%. Qualitative results were obtained via a second round of user feedback that evaluated the state of the system after the improvements. The results showed an overall average improvement of 21.2% across all factors.

The improvements in both the user acceptance percentage as well as the improvement in the average user ratings of the software factor confirm that the methodology was effective in increasing user adoption of software systems and all of the objectives were fulfilled. It was found, as suggested by literature, that increasing the PEOU could lead to an improve PU.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

Greek symbol	Description	Unit
α	Cronbach's alpha	-

LIST OF ABBREVIATIONS

IoT	Internet of Things
TRA	Theory of Reasoned Action
TPB	Theory of Planned Behaviour
TAM	Technology Acceptance Model
TAM2	Extended Technology Acceptance Model
VAM	Value-based Adoption Model
UTAUT	Unified Theory of Acceptance and Use of Technology
BI	Behavioural Intention
A	Attitude
SN	Subjective Norm
PBC	Perceived Behavioural Control
PU	Perceived Usefulness
PEOU	Perceived Ease of Use

CHAPTER 1: INTRODUCTION

1.1 Preamble

The rise of industry 4.0 has driven the digitisation of large scale industrial and mining operations. This introduced the burden of large volumes of data that needs processing. Data from industrial systems is usually processed by specialised systems and software. Numerous software systems exist to process this data and generate reports; however, users may be biased towards working with non-optimal software. Due to various reasons, this leads to purpose-built software being underused.

This study aims to provide a method to increase user adoption of software by addressing the factors affecting user acceptance from a software engineering perspective. A comprehensive summary of technology acceptance theories is done, followed by a discussion of software quality factors. Next, a brief overview of various software development methodologies is given. Lastly, the state of the art is investigated through a literature study of existing studies. The shortcomings regarding existing studies are discussed, followed by a discussion on how these shortcomings will be addressed.

To increase the readability of the thesis, reference is made to ‘software systems’ to refer to software systems in general, as well as the system being focused on.

1.2 Background

Digitisation of industrial operations

The phrase industry 4.0 has become a topic of much discussion as industry focus starts to shift to the digitisation of operations and increased usage of interconnected electronic systems [1]. However, with all the ongoing development and discussions, it is easy to lose track of the origins and meaning of the concept.

Industry 4.0 refers to the ongoing fourth industrial revolution. The concept was introduced by the German government in 2011 and refers to the increasing focus on digitisation of production facilities and increased usage of electronics and software in industrial operations. According to Lee et al., the dawn of the fourth industrial revolution can be attributed to the increased use of cyber-physical systems, the Internet of Things (IoT) and advanced software services [2], [3].

Examples of the benefits that the industry 4.0 concept brings can be found in the increasing digitisation occurring in the mining industry. Deep level mine shafts can be several kilometres underneath the earth and can stretch for kilometres. Before digital sensors and interconnected systems, environmental readings had to be taken manually. Interconnected systems that push sensor readings to a central platform for analysis save countless man-hours and speeds up decision-making so that operations managers can respond to change more effectively. Automated control systems could also be implemented to reduce resource usage where possible to improve efficiency [4], [5].

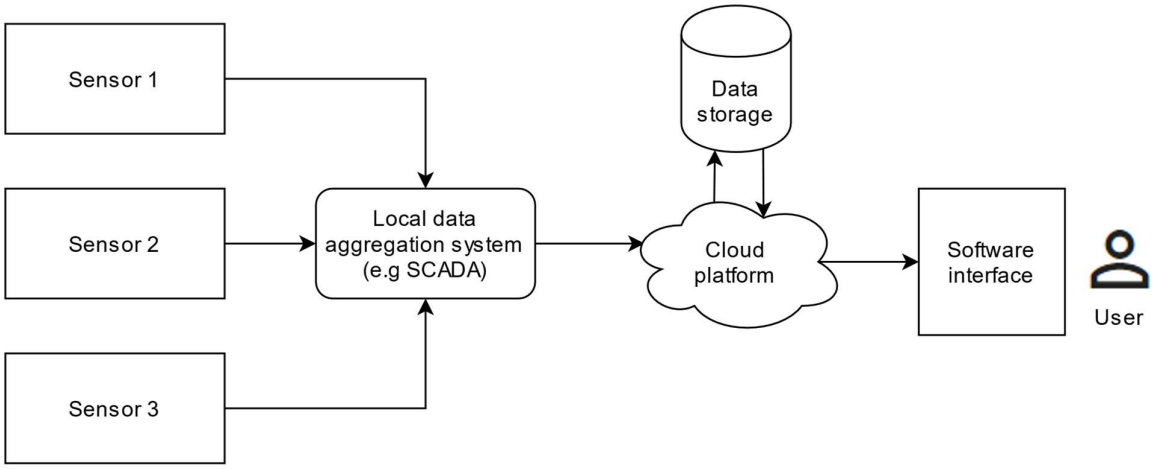


Figure 1: Data flow in IoT applications (adapted from [6])

Figure 1 shows an example of an Internet of Things (IoT) enabled system used in industrial or mining applications. The sensor data is aggregated and sent to a cloud platform which stores the data in a database and provides access to a user via a software or web user interface. In some cases, the system can relay information back to the equipment to react to the current conditions.

Another example can be found in the manufacturing industry. Smart factories powered by IoT enabled systems are rising in popularity. For example, smart production lines could improve efficiency and provide real-time data that allow for instantaneous feedback and decisions. Automated workflows could increase the yield, uptime and quality while simultaneously reducing operational costs [1], [7].

Digitisation makes the acquisition and linking of raw data sources easier and with the widespread availability of inexpensive, accurate sensors, the process is relatively inexpensive. Sensors can be installed and linked to a network and provide a constant stream of data to a central cloud-hosted system. The data can be aggregated into the desired format to generate reports. The

reports could be used to gain an overview of key factors and quickly identify problems. Systems and reports could also be used to monitor data to improve resource usage and implement cost-saving strategies [5], [8].

Data processing software

The result of the additional electronics, software systems and digitisation of existing processes is that more data is being generated. The data needs to be aggregated and analysed to be useful. A data analytics system can be used to analyse the data. Data analytics systems are systems that enable users to analyse data, learn from the analysis and use the knowledge and insight to optimise business operations [9]. Data analytics systems perform several functions to process input data to produce meaningful information as an output [10].

Because of the large amounts of data that industrial organisations and companies generate that need to be processed and analysed, companies have started to offer large scale solutions for data processing to the manufacturing and mining industry called Business Intelligence (BI) systems. BI systems can be defined as systems that can analyse and present information to users to enhance decision-making capabilities [11], [12]. However, these systems face challenges in the form of user adoption.

Software adoption challenges

The success of software relies heavily on satisfaction of system users [11]. If users are not satisfied with the capabilities and results that are offered by a system, they may end up not using it. Performance gains are often hampered by the unwillingness of some users to accept and utilise certain systems with which they are either not familiar with or see no value in using [13].

Most users have their own personal experience with certain software that can process data and generate reports. This means that users have a personal preference towards using a piece of

software. This could result in a variety of software being used inside a company instead of a single standardised system, which leads to differences in report formatting and inconsistent results[1] [14].

People that use software which is not optimal for a specific task may get results that are not accurate, or the process may take longer than necessary, thereby decreasing work efficiency. For example, using purpose-built BI systems that handle data processing and reporting could increase efficiency. If suboptimal software is used instead for the same purpose, efficiency is lost on different levels within a company [15].

Importance of technology acceptance

User acceptance of software is important in three areas, which is the perspective of a manager as a user, a manager of the development process and manager of an organisation [16].

Managers are an important component of an organisation, and need to be as efficient as possible. Information systems could improve managerial efficiency by improving information processing, communication and decision-making activities. Thus, managers must use all the technology available to them [16].

Organisation managers are responsible for keeping their employees as efficient as possible. An increasing amount of work is done using software systems. A manager must ensure that the employees use the software that best fits their needs, and as a result, the manager could gain valuable information from software user acceptance studies. Software systems cannot improve the organisational performance if they are not used to their full potential by professionals [16], [17]. Resistance towards adoption of software tools could lead to reduced overall organisational performance [18].

Development managers could benefit immensely from software user acceptance research that helps to better understand the key determinants of software user acceptance of the system in question. When managing the development process of a new system, the user acceptance rate of the software system being developed should be considered. Wasted resources could cause the cost of software development to increase, potentially affecting the profitability of the software

[1] D. Ende and B. Groenveld, “Using standards to increase the value of reporting.” https://www2.deloitte.com/content/dam/Deloitte/nl/Documents/audit/deloitte-nl-audit-xbrl-ceo-today-using-standards-to-increase-the-value-of-reporting.pdf (accessed May 19, 2020).

being developed. It is estimated that between 30% and 80% of software cost could be attributed to maintenance or improvements made after the initial release [19], [20].

The needs of users may differ from the original specifications that the project was based on. The information gained could be used to understand where a certain system lacks usefulness or ease of use so that resources can be allocated accordingly [19]. As a result, more time can be spent on areas that are important to the user, and less development time is wasted. The user benefits from this as the software could conform more closely to their needs and the user experience also improves [16].

1.3 Background summary

Industry 4.0 provides clear benefits for industrial organisations such as mines and factories. To take advantage of the benefits, large amounts of data are processed and analysed [8]. If the data is processed by suboptimal software systems, the benefits of industry 4.0 decrease. The increased efficiency gained by quickly gaining information from interconnected systems is nullified by the time spent processing data or by providing suboptimal information from inaccurate results [15].

On a management level, if work is done inefficiently, the company loses value as resources are not being managed efficiently, and decision-making could be delayed. On a user level, if different software is used instead of the organisational standard, reports generated by the system may not conform to the organisational standard. Moreover, data sources may also be processed differently or inaccurately, lowering the consistency of the information provided to clients and managers [16], [17].

Finally, from a development perspective, suboptimal usage of software results in wasted effort and resources. By studying the user adoption factors, resources can be focused on improving the shortcomings from a user perspective and increase overall software utilisation which benefits the organisation using the software as well as its clients [19], [20].

The next step is to investigate and compare user acceptance theories and models. The place of software user acceptance research will be investigated in popular software development methodologies. Existing studies on user acceptance will be evaluated, and the insights gained will be used to develop a method that analyses the current state of user acceptance factors of a system, suggests improvements and tests the effectiveness of the improvements on user acceptance.

1.4 Need for the study

The background above indicates that technology acceptance is an important factor. Low acceptance causes a decrease in resources and efficiency. Software systems need to be improved with the goal of improving user acceptance.

The need for the study is to show that acceptance can be improved by modifying the software system accordingly. Thus, the need is to develop a general method of improving software acceptance.

The following objectives will be met throughout this study:

1. Investigate theories and methods to analyse and improve the user acceptance.
2. Develop a general method to improve user acceptance of an underutilised software system.
3. Implementation of the method on a case study and validating the effectiveness of the methodology.
4. Conclude on the effectiveness of the methodology to improve user acceptance, comment on the findings and provide recommendations for future work.

1.5 Document overview

The rest of this thesis will be focused on investigating and solving the problem identified in chapter 1 and will be structured as follows:

Chapter 1: Introduction

Chapter 1 provides a background of the problem of user acceptance as well as literature aimed at solving this problem. The concept of industry 4.0 is briefly discussed, as well as the advantages it brings. The necessity of data in interconnected systems and the software systems that process the data is highlighted. Lastly, the importance of high acceptance rates for software systems is discussed.

Chapter 2: Literature study

Chapter 2 investigates the current state of literature that can help to solve the problem. Technology acceptance theories are investigated, evaluated and compared. The best model to assess user acceptance of software from a software engineering perspective is selected. Factors that affect software acceptance are discussed, and an overview of software design methodologies is given.

Existing studies that use the selected acceptance model are investigated and an overview of the state of the art is given. Findings and shortcomings identified from the studies are discussed, followed by a brief problem statement and the need for the study that is identified from the literature.

Chapter 3: Methodology

Chapter 2 provides an overview of the methodology developed to improve user acceptance of a software system. The selection of an underused software system is briefly discussed, followed by clear definitions of the software quality factors that are analysed in the methodology. Next, the grouping of software factors is discussed. A summary of the theory behind development of a Likert scale questionnaire is given, followed by the questions pertaining to each subfactor. Distribution of the questionnaire and procedure for analysis of the results is outlined, followed by methods from literature to improve the software aspects related to each factor. Next, an example of results that can be expected is given. Finally, the validation process for the effectiveness of the methodology is given.

Chapter 4: Results and Discussion

Chapter 3 validates the effectiveness of the developed methodology by the means of a case study and ensures that the objectives of the study are fulfilled. The software system used as case study is introduced, and the reasons why it is suitable for the methodology is discussed. The first set of results from the questionnaire is analysed and thoroughly discussed. From the first set of results, the factors that perform the worst is highlighted and the improvements made regarding each factor is showcased, followed by a second set of results. The findings from the case study are discussed, and the effectiveness of the methodology is analysed.

Chapter 5: Conclusion

Chapter 4 concludes the study by summarising the work done, commenting on the case study results and drawing a conclusion regarding the effectiveness of the developed methodology. Shortcomings of the current study are identified and recommendations for future work are provided.

CHAPTER 2: LITERATURE STUDY

2.1 Preamble

In the previous chapter, the benefits of industry 4.0 and interconnected systems were established. Software adoption challenges were discussed, and the importance of software user acceptance was highlighted from a managerial, user and software development perspective.

In this chapter, a thorough evaluation of current literature will be done. Technology acceptance theories will be investigated, evaluated and compared. Technology acceptance theories are defined as theories that attempt to explain the factors that affect acceptance of technology. The best model to assess user acceptance of software from a software engineering perspective will be selected, and existing studies that utilise the selected model will be evaluated. Factors that affect software user acceptance will also be discussed, and an overview of software design methodologies will be given.

2.2 Technology acceptance theories

The rapid development of information systems and technologies has prompted the development of theories surrounding the factors that affect the acceptance of these technologies.

Numerous models and theories have been developed over the years that attempt to explain the acceptance of new and old technologies. The theories have their origins in the psychological and sociological research fields but were expanded to model the acceptance of computers and software systems. These theories aim to better understand, predict and improve the user acceptance of advanced software systems. Most of the theories are centred around information system acceptance, however, they can be adapted to understand and predict the acceptance and use of any technology [13], [21], [22].

According to Taherdoost [21], acceptance, in general, can be defined as: “The positive decision to use an innovation and an antagonism to the term refusal”.

Thus, technology acceptance refers to the decision by users to use a particular technology [21].

This section will investigate the most predominant models and theories related to technology acceptance. The following models were investigated:

- Theory of Reasoned Action (TRA) [23]
- Theory of Planned Behaviour (TPB) [24]
- Technology Acceptance Model (TAM) [16]
- Extended Technology Acceptance Model (TAM2) [25]
- Unified Theory of Use and Acceptance of Technology (UTUAT) [26]

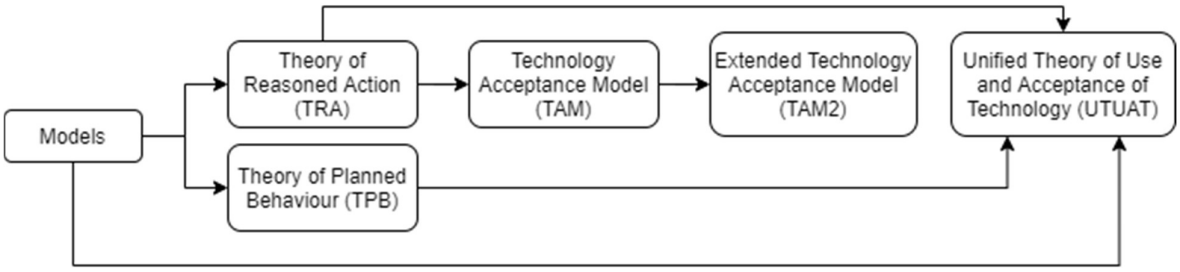


Figure 2: Technology acceptance theories (adapted from [21])

The relationship between the models can be seen in Figure 2. The Theory of Reasoned Action (TRA) forms the basis of these models. The Technology Acceptance Model (TAM) and the Theory of Planned Behaviour both developed from TRA independently. The Extended Technology Acceptance Model (TAM2) is a continuation of TAM. The Unified Theory of Use and Acceptance of Technology (UTUAT) is a combination of all of the theories developed earlier.

Theory of Reasoned Action (TRA)

The model that forms the basis of these theories is the Theory of Reasoned Action (TRA) which was introduced by Fischbein and Azjen [24]. TRA attempts to determine the factors that affect consciously intended behaviour. The theory is well supported by empirical evidence [27]. This makes it one of the most frequently cited and most influential models used to predict behavioural intention and behaviour in a variety of fields, including technology acceptance [28].

According to the model, the person's performance of behaviour is determined by the person's intention to perform a specific action. In turn, attitude and subjective norm determine the behavioural intention where the relative weights are estimated using regression. The attitude refers to the person's "favourableness" towards behaviour or action, whereas the Subjective Norm refers to the perceived social influence or pressure from others that certain behaviour is acceptable or even encouraged [17].

The external variables are assumed to influence either attitude or subjective norm. Thus, the following formula signifies the behavioural intention:

$$BI = A + SN \quad (1)$$

Where:

- BI is the Behavioural Intention
- A is the Attitude towards the behaviour
- SN is the Subjective Norm

Figure 3 represents a visual representation of TRA:

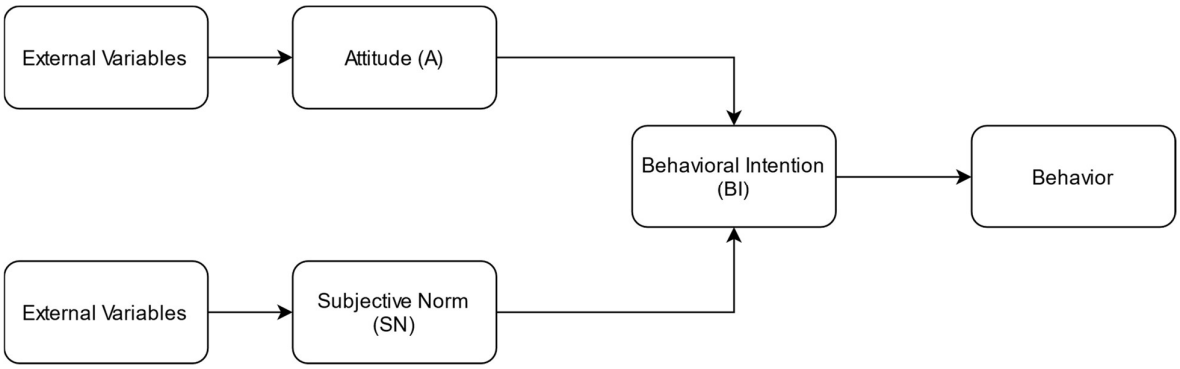


Figure 3: Theory of Reasoned Action [23]

Technology usage is one of the many fields where TRA is used to predict behaviour and usage. For example, it was used as the basis to study consumer attitudes towards mobile advertising [29], intention to use online stock trading [30], computer-based word processing adoption [17] and user acceptance of internet banking [31].

Theory of Planned Behaviour (TPB)

The Theory of Planned Behaviour (TPB) is an extension of The Theory of Reasoned Action (TRA). The author of TRA identified certain goal-directed behaviours over which an individual has limited volitional control. Volitional control refers to the choices or actions an individual makes through one's own will and decisions.

The external and internal factors that could influence volitional control were identified, and a behaviour-goal relation is established and defined. TRA was then adapted to explain goal-

directed behaviour over which the individual has limited volitional control. The new theory is called TPB and is different from TRA in the sense that it considers the perceived as well as actual control of the behaviour [27], [28], [32].

The basic constructs that TPB is modelled on are the same as TRA, with the addition of the Perceived Behavioural Control (PBC). Thus, the basic equation that describes Behavioural Intention (BI) is as follows [27]:

$$BI = A + SN + PBC \tag{2}$$

where:

- BI is the Behavioural Intention
- A is the Attitude towards the behaviour
- SN is the Subjective Norm
- PBC is the Perceived Behavioural Control

Figure 4 represents TPB visually:

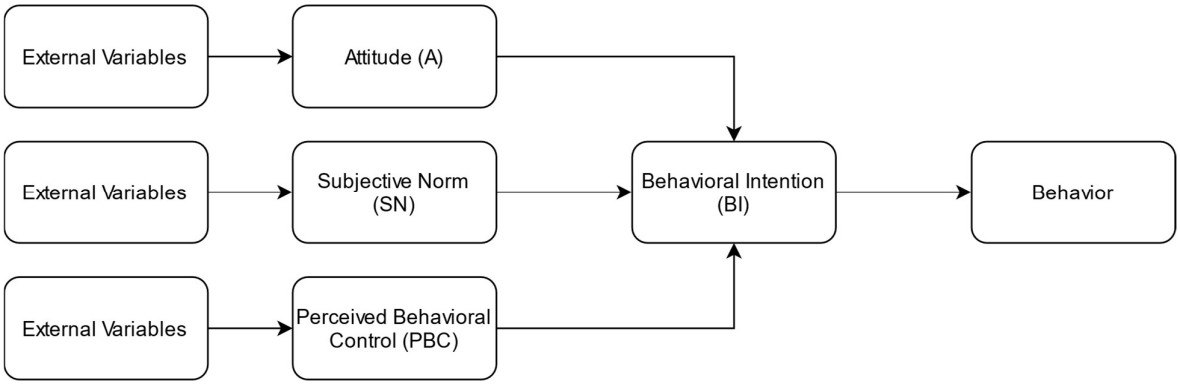


Figure 4: Theory of Planned Behaviour [27]

It is mostly used in analysing and predicting everyday behaviours and not used frequently when looking at user acceptance behaviours. However, it was used alongside the Technology Acceptance Model (TAM) to investigate and predict the intention to adopt online learning technologies amongst students. It was found that both theories could predict the intention to adopt e-learning technologies well, but TAM was more robust in this particular case [33], [34].

An extended version was used to investigate the use of technology by teachers. It was found that the attitude towards computers has the largest impact on the intention to use amongst teachers, followed by perceived behaviour control [35].

Technology Acceptance Model (TAM)

The Theory of Reasoned Action was adapted and extended by Davis in an attempt to model the acceptance of information technology [36]. Subjective norm is removed from the model, which leaves only the attitude as the determining factor. Attitude is modelled using two factors, namely the Perceived Ease of Use (PEOU) and Perceived Usefulness (PU). The new model was called the Technology Acceptance Model (TAM) and has since become one of the most used models in analysing and predicting technology acceptance [13], [21].

Davis [16] defines PU as the following: “The degree to which a user believes that using a particular system would enhance his or her job performance”.

A system high in PU is a system for which a user believes that there is a positive use-performance relationship. If a user has high PU of a system, it means they realise that the system would bring a lot of benefits when using it, such as increasing work efficiency, accuracy, reliability or scalability. Because organisations and companies generally incentivise good job performance through some form of compensation like raises and bonuses, people tend to opt for the systems which could provide them with the capability to raise their job performance, thereby reinforcing the notion that PU is an important factor in predicting user acceptance of software [13], [22], [37]–[39].

The definition of ease is “freedom from difficulty or great effort”. Davis [16] defines PEOU as the following: “The degree to which a person believes that using a system would be free of effort”.

Even if a potential user has high PU of a certain piece of software, there is no guarantee that they will utilise the system if the system is too complicated to use or learn. If the amount of effort required to use a system outweighs the benefits, a user may opt to utilise a different system instead. PEOU affects the PU of a system as well as the Behavioural Intention [13], [38], [40].

Thus, the relation to Behavioural Intention is given by the following formula:

$$BI = PU + PEOU \quad (3)$$

Where:

- BI is the Behavioural Intention to Use
- PU is the Perceived Usefulness
- PEOU is the Perceived Ease of Use

The behavioural intention to use a system has been proven to be a reliable predictor of actual software use in numerous software user acceptance case studies. In both studies that the original model was tested on, PU was found to have a greater correlation to usage behaviour than PEOU. The analysis also suggested that PEOU influences PU, which was later verified in subsequent studies [16], [19].

External factors influence these two factors and differ between use cases. The external variables are usually hypothesised, tested and verified while using the model, but factors from other models or prior research can also be used, as demonstrated by F. Lin et al. [39].

The validity of the model has been tested numerous times over the years and found to be more than adequate for analysing and predicting technology acceptance. It was validated by Subramanian [41] on two further case studies, as well as Adams et al. [42], both of which found high levels of correlation between PEOU and PU on the intention to use. Adams pointed out that some special cases do exist where the PEO or PU may not be optimal but the system in question is the only option available for the task, thus the actual usage figures may be high regardless.

Additional examples are as follows: it was used to do research on the acceptance factors for business intelligence tools [43], develop a better electronic health record system [37] and study adoption of interactive media systems by radiologists [44].

Compared to the Theory of Reasoned Action (TRA) and Theory of Planned Behaviour (TPB), the Technology Acceptance Model (TAM) was found to be a simpler model and can be generally applied to any system, whereas TRA and TPB need to be tailored to each specific system. The ease of use also makes TAM a more attractive option than either TRA or TPB [45].

Extended Technology Acceptance Model (TAM2)

An extended version of the Technology Acceptance Model **(**TAM) was developed by Venkatesh and Davis [25]. The new theory aimed to find factors influencing TAM's Perceived Usefulness (PU) and Usage Intention, and to understand how these determinants affect change with increasing user experience of a system. The new model is called The Extended Technology Acceptance model or TAM2 [25].

TAM2 suggests a few factors that may influence the PU of a system:

- Subjective norm
- Experience
- Image
- Job relevance
- Output quality
- Result demonstrability
- Voluntariness

First is the subjective norm, which is defined as "a person's perception that most people who are important to him think he should perform the behaviour or not" [25]. This was included as a direct determinant of Behavioural Intention in the Theory of Planned Behaviour (TPB) and was included because of the rationale that people may perform behaviour they don't necessarily agree with, but still perform the behaviour because of social influences [25].

The experience that a user has with a system is also added as a direct antecedent of PU, as well as the overall image of the system. Job relevance, output quality and result demonstrability of the system are also theorised to influence PU. Finally, voluntariness is defined as "the extent to which potential adopters perceive the adoption decision to be non-mandatory". The relation of Perceived Ease of Use (PEOU) and PU to Behavioural Intention (BI) is the same as the Technology Acceptance Model (TAM), but with the addition of the abovementioned factors to increase the predictive power of PEOU [25].

Examples of TAM2 include the use by Gellerstedt et al. [46] to analyse the adoption of ICT systems in schools by teachers [46], and the use of traceability systems in agri-food supply chains [47].

Unified Theory of Acceptance and Use of Technology (UTAUT)

Venkatesh and Morris [48] studied the similarities and differences between a variety of models previously used in the context of information systems to develop a unified model. UTAUT identifies four key factors ("Performance Expectancy, Effort Expectancy, Social Influence and Facilitating Conditions") and four moderators ("Age, Gender, Experience and Voluntariness") that relate to Behavioural Intention [26], [48].

Figure 5 illustrates the relationships between all the above-mentioned factors [48]:

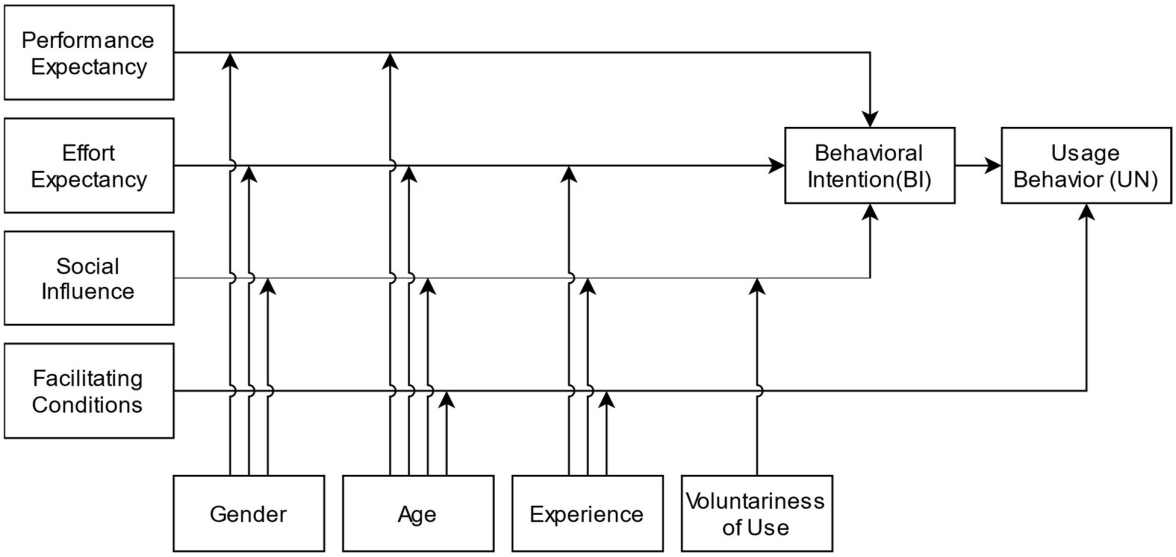


Figure 5: Unified Theory of Acceptance and Use of Technology [48]

Examples where UTAUT was used to investigate technology acceptance is the adoption of internet banking [36], mobile shopping services [49] and online crowdfunding [50].

However, most studies utilising UTAUT employ only a subset of the factors, as stated by Venkatesh et al. [51]. The study is also often extended by adding additional constructs to fit the situational needs, for example, Baishya and Samalia used an extended UTAUT to investigate the adoption of smartphones [52].

Summary of acceptance theories

The Theory of Reasoned Action (TRA) and Theory of Planned Behaviour (TPB) was found to be effective at predicting the acceptance of software, however, the subjective norm aspect was found to have a varying amount of effects on the behavioural intention to use, thus introducing variance between studies. Furthermore, the social aspect of software usage cannot be controlled by software developers, thus eliminating TRA and TPB as possible options for a general method to improve user acceptance. This is cemented by the fact that very few software-oriented studies used TPB and TRA compared to the TAM when investigating user acceptance.

The Unified Theory of Acceptance and Use of Technology (UTAUT) and the Extended Technology Acceptance Model (TAM2) aims to solve some of the limitations of TAM by introducing various factors that aim to explain the variance in use caused by social factors. However, the social factors cannot be improved by a software engineering team, and as such neither model is ideal for improving user acceptance from a development perspective.

This is supported by the fact that all the studies where UTAUT or TAM2 is used, the information was only used to gain insight and was not utilised to improve the user acceptance of the system in question. In most studies, the factors from the original model are either modified or excluded from the study, further showing that the models can only be applied to specific circumstances.

Most suitable theory

The Technology Acceptance Model (TAM) was found to be the best model to base a general method for software improvement. It was by far the most frequently used out of all the theories investigated, despite not being the oldest, and is backed up by plenty of empirical results from technology acceptance studies. It is easy to implement on any software system, and results could be gathered quickly without an excessive waiting period. Because there is no social aspect involved, the variance that different personality types may bring is eliminated and is suitable for being used in organisation-wide studies.

External factors are used to gauge the Perceived Ease of Use (PEOU) and Perceived Usefulness (PU) which can differ between use cases, thus making TAM suitable for a general method to improve user acceptance of software systems if a general approach to identifying factors relating to PEOU and PU is taken.

2.3 Technology acceptance factors

The previous subsection discussed technology acceptance models, and it was found that the Technology Acceptance Model (TAM) by Davis [16] is best suited for analysing user acceptance acceptance of software in general without looking at social or organisational factors. This section will focus on the specific factors of a Business Intelligence (BI) system that can be used in TAM.

Software quality factors

Different factors are used in software user acceptance studies, depending on the application. These factors will be investigated.

This study will focus on factors that could be improved upon for a data analytics system from a software development perspective, thus social and organisational factors will not be looked at. Software quality factors can be improved by a development team and will be investigated.

According to Verma et al. [18] the main factors affecting BI software is information quality and system quality. These factors are encompassed by software quality and were verified to significantly impact the attitude towards BI software. Verma et al. found that system quality and information quality affects the user acceptance of data analytics systems, thus the information and system quality will be a key focus of this study [18].

Software quality models are essential for identifying deficiencies and limitations of software that can affect user acceptance. In this section, the ISO 25010 software quality model [53] will be discussed that can be applied to any software system, including data analytics software.

Improved software quality has a direct effect on user acceptance. Users are more likely to use and trust a well-designed system with quality characteristics such as the following [54]:

- Improved efficiency
- Improved productivity
- Reduced errors
- Reduced training needed
- Accessible functionality
- Easy to learn and use

Software quality encompasses both system and information quality. Quality is a subjective term, and as such there are many different definitions of software quality, usually relating to excellence or fitness for the purpose for which it is intended. Another definition is the conformance of software to specifications. According to the ISO 25010 standard, quality is defined as “a set of features and characteristics of a product that bears on its ability to satisfy the stated or implied needs” [55].

For the purpose of this study, software quality will be defined as the degree to which the software complies with the specifications based on functional requirements and specifications. Thus, the quality of a software system can be measured by how closely it fulfils the needs of those who use it for the purpose it was built for [56]–[58].. This fits in perfectly with the objectives of this study. It provides a general overview of all the important factors hen analysing system and information quality.

It is critical that software quality is evaluated using widely accepted metrics, thus the ISO 25010 standard for software quality will be used. ISO 25010 is part of the International Organization for Standardization’s (ISO) Systems and Software Quality Requirements and Evaluation (SQuaRE) series of standards and consists of several criteria for the assessment of software quality.

According to ISO, the goal with ISO 25010 was to cover two main processes: Software quality requirements specifications and system and software quality evaluation. The purpose was to assist developers and users with developing and acquiring software systems that meet certain quality standards. ISO 25010 is a revised version of ISO 9126 [54], [55], [59].

ISO 25010 specifies the following main software quality factors [55]:

- Functional suitability
- Performance efficiency
- Compatibility
- Usability
- Reliability
- Security
- Maintainability
- Portability

Below follows a short description of each factor, followed by an example of usage.

Functional suitability

Functional suitability refers to “the degree to which a system provides functions that meet the stated and implied needs when used under specified conditions” [57].

The functionality of a system is one of the key reasons why people choose a specific software system, as demonstrated by Rajković et al. where it was used as a key factor to improve user acceptance [37].

Performance efficiency

Performance efficiency refers to “the performance relative to the number of resources utilised under stated conditions”[57]. Resources could include the software itself, other software, hardware and network resources. This may also include not only the performance of the system but also the required amount of resources [60].

The system performance efficiency could have an impact on the acceptance due to hardware requirements being lower if the system is more efficient. If efficiency is high, the system could also process data quicker, which improves work efficiency and leads to increased user acceptance [18].

Compatibility

Compatibility refers to “the degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions while sharing the same hardware or software environment” [57]. In other words, the compatibility factor refers to how well the system interacts with other systems [60].

Compatibility with other systems is important for situations where information needs to exchange between systems. This is an important aspect of information quality, as the input and output data provided needs to be accurate and reliable to be used by other applications. Poor quality of exchanged data may lead to adverse effects on decision-making. For example, incorrect temperature readings in [18].

Usability

Usability forms a major part of the quality of a software system. It refers to “the degree to which a system can be used by specified users to achieve specific goals with effectiveness, efficiency and satisfaction” [57]. It also refers to the capability of the software to be understood and learned, and also how attractive the software is [60], [61].

Usability is often a key part of a software assessment, and essential for guaranteeing the quality of the software. Usability analysis could lead to improvements to ease of use and improvements to the efficiency of the user when using the system. Making a system easier to use leads to faster turnaround time for work that the user needs to perform on the system. A system with high usability is more likely to convince users of the benefits it brings, and thus keeps users from switching to other systems [62], [63].

Reliability

Reliability of a product refers to “the degree to which a system, product or component performs specified functions under specified conditions for a specified period” [57]. It can also be defined as the ability of software to maintain operational capacity when used under certain specified conditions [61].

Reliability is another key part of the PU of a system. The amount of testing that forms a part of the software development process is a testament to the importance of good reliability, and the effect that it has on user acceptance. Poor software reliability may lead to suboptimal outcomes, wasted development resources, low user acceptance rates and even lawsuits [64].

Security

Security refers to “the degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorisation” [57]. This applies to the data that is stored and accessed, as well as data being transmitted over a network. It can also be defined as “the idea of engineering software so that it continues to function correctly under malicious attacks” [65].

Security and access control are important parts of enterprise quality software. Access to confidential data should only be access by staff members authorised to do so. Different access levels are also important, for example, complete access, view only, data input only and access to reporting and analysis tools [62].

Maintainability

Maintainability of software refers to “the degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers” [57]. Maintainability of the software is the concern of the development team behind the system, and as such it has no impact on the PEOU or PU of a system from a user’s perspective. Thus, it will not be included in the research questionnaire included in this study.

Portability

Portability is defined as “the degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another” [57]. This could refer to the capability of the system to facilitate porting of the system to a different platform from the maintainer’s perspective, or the accessibility of the system on different platforms from a user’s perspective. The latter definition will be used in this study [60]. The portability of software ties directly into improving software to increase acceptance [66].

Except for maintainability, all the factors and subfactors included in ISO 25010 [53], [55] will be used in the method for improving the user adoption of software systems. The ISO 25010 standard provides a good basis to analyse the user base’s PU and PEOU. The system quality and information quality factors used in previous studies, are covered by the subcharacteristics of the factors mentioned above, and the subcharacteristics can be used to form a research questionnaire.

2.4 Software user acceptance testing in different development methodologies

The previous subsection discussed the factors that are analysed when investigating the acceptance of data analytics software. This section identifies the role that acceptance testing plays in various software development methodologies. This is important as improvements to a system need to occur within the standard development cycle.

The software development methodology must be defined to understand the role that software user acceptance studies could play in software development and acceptance.

According to Van der Bank [67], a Software Development Methodology (SDM) can be defined as “frameworks or processes according to which a software project can be developed”. These

methodologies describe detailed phases according to which the development process can be divided. Each phase focuses on and optimises a specific aspect of the process. A software development methodology is often referred to as a software development life cycle [67].

Popular SDMs are [68]:

- Waterfall
- Incremental
- V-Shaped
- Rapid Application Development (RAD)
- Spiral
- Agile

In order to understand when user acceptance testing needs to take place within each SDM, these methodologies need to be investigated.

Waterfall

The waterfall methodology is one of the oldest and most commonly used SDMs [69]. The waterfall SDM follows a linear process as demonstrated by Figure 6 [70]:

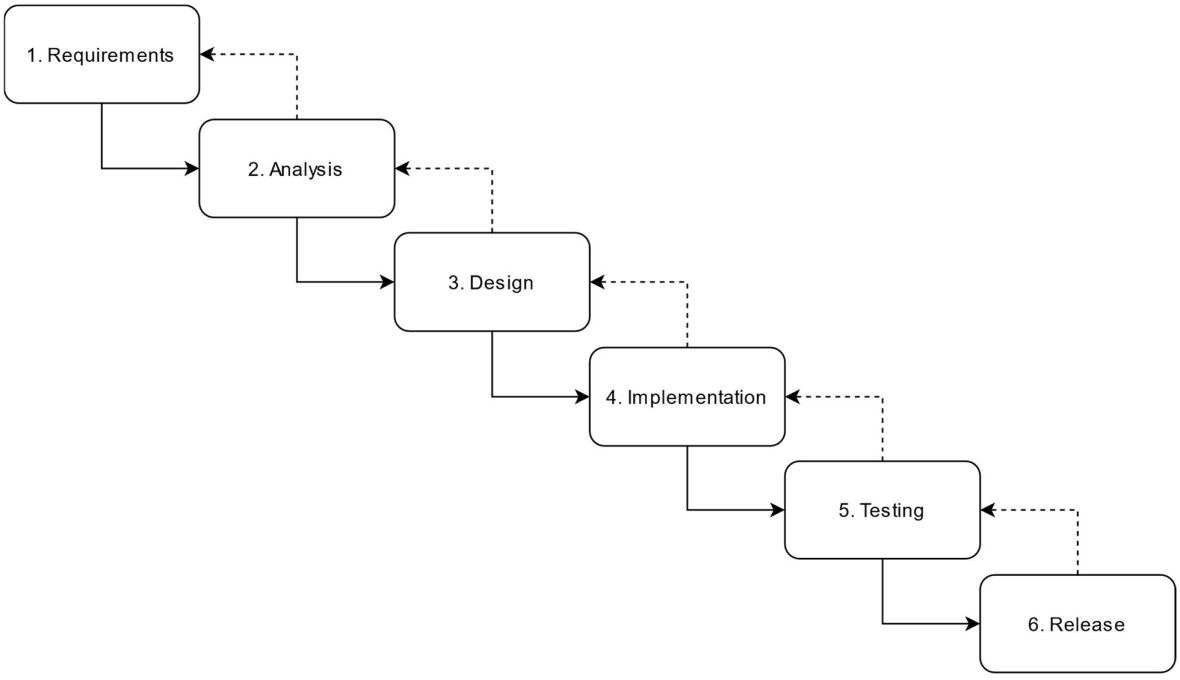


Figure 6: Waterfall methodology [71]

The following phases are part of the waterfall SDM:

1. **Requirements:** The user requirements are gathered
2. **Analysis:** The requirements are analysed and converted to software specifications
3. **Design:** A design of the system is formulated that conforms to the specifications
4. **Implementation:** The system is built according to the design.
5. **Testing:** The functional parts of the implemented system are tested and verified.
6. **Release:** The system is released to the users and maintenance is done when needed.

The most noticeable aspect of the waterfall SDM is that the next phase cannot begin until a previous phase has been completed. The requirements need to be clearly defined and well understood. If the requirements change mid-cycle, the entire process begins from the beginning, which could delay the release of the finished product. As a result, the waterfall SDM is often not preferred in modern software development [68], [71].

User acceptance testing takes place before initial software release during the testing phase.

Incremental software development

The incremental SDM focuses on the incremental construction of software projects via a series of iterations. The iterations consist of the phases shown in Figure 7:

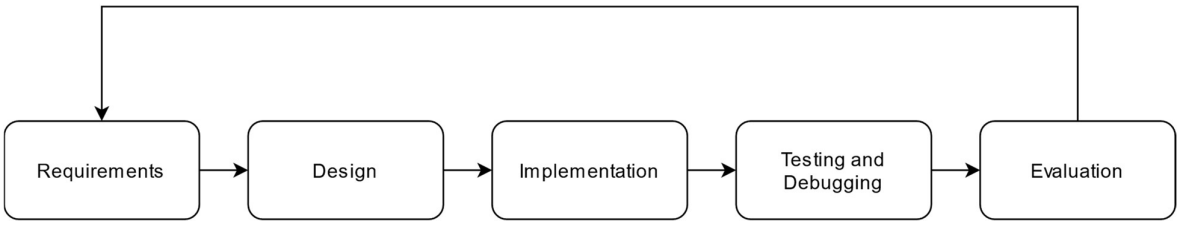


Figure 7: Iterative software development [69]

Incremental SDM consists of smaller feature drives that follow a similar set of phases than the waterfall SDM. The software system that is being developed is split into several smaller sections, each with its own set of requirements or specifications. The process is followed for each of these sections, and when the functionality has been completed and evaluated, the process starts from the beginning for a new piece of functionality that adds to the existing functionality. Multiple software modules may be worked on concurrently, with each module following its development cycle independently [69].

The following phases occur during incremental software development [69]:

1. **Requirements:** The user requirements are gathered.
2. **Design:** A design of the system is formulated that conforms to the specifications
3. **Implementation:** The system is built according to the design.
4. **Testing and debugging:** The functional parts of the implemented system are tested and verified.
5. **Evaluation:** The system is released to the users and maintenance is done when needed.

User acceptance testing forms part of the testing and debugging phase before the initial version of the product is released.

V-Shaped methodology

The V-shaped methodology is an extension of the waterfall SDM. The SDM focuses heavily on testing, and as a result, each development stage is matched with a testing stage [72].

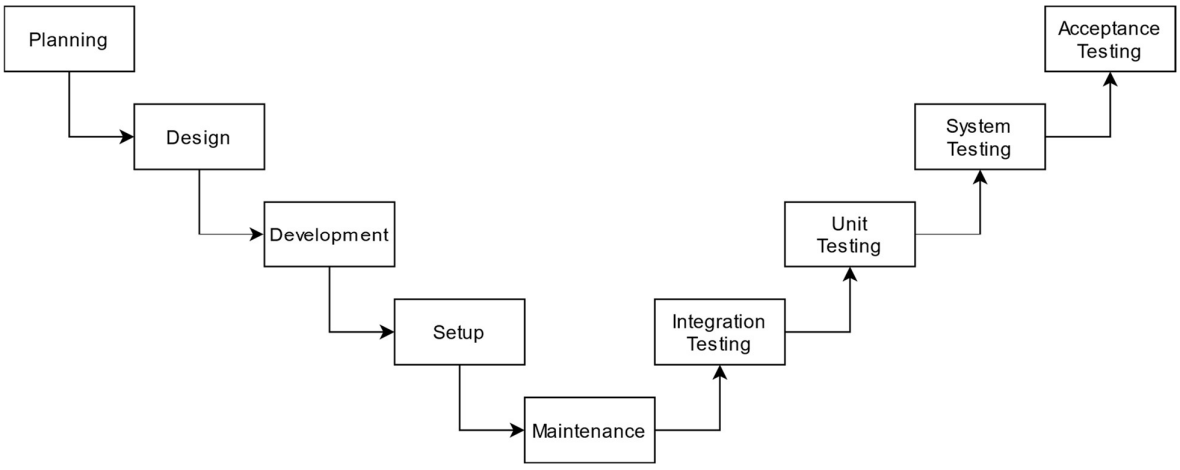


Figure 8: V-Shaped methodology [72]

User acceptance testing takes place after general system testing at the end of the process before the system is released.

Rapid Application Development (RAD)

Rapid application development aims to provide much faster development and higher quality systems than traditional methodologies. Emphasis is placed on development rather than meticulous planning. Development cycles are handled in time boxed stages during which multiple modules are worked on in parallel. When all of the modules are completed, the final integration is done, and the product is delivered [72], [73].

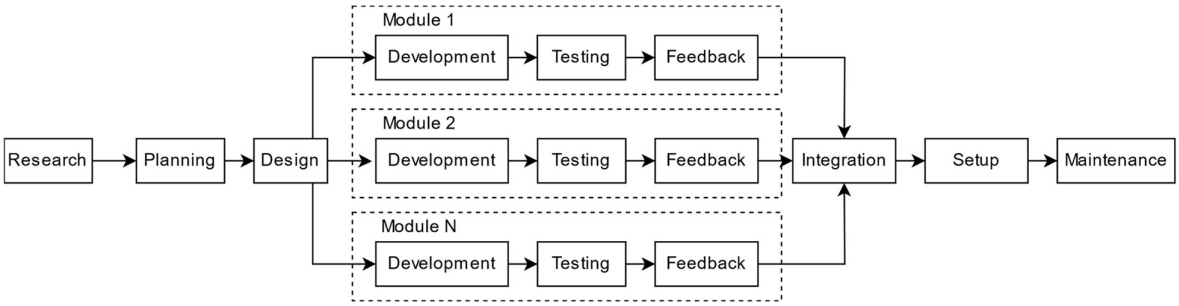


Figure 9: Rapid Application Development [72]

User acceptance testing takes place after the initial product is complete, before the system is released.

Spiral methodology

The spiral SDM focuses on identifying objectives and comparing different solutions. The spiral SDM has four main phases [72]:

1. **Planning:** Research is done regarding requirements and objectives. The functional structure of the system is planned.
2. **Risk Analysis:** A design is drawn up and a prototype is created. The prototype is assessed and analysed.
3. **Development:** The system is developed and tested.
4. **Evaluation:** Feedback is garnered from the client and the spiral process repeats if required.

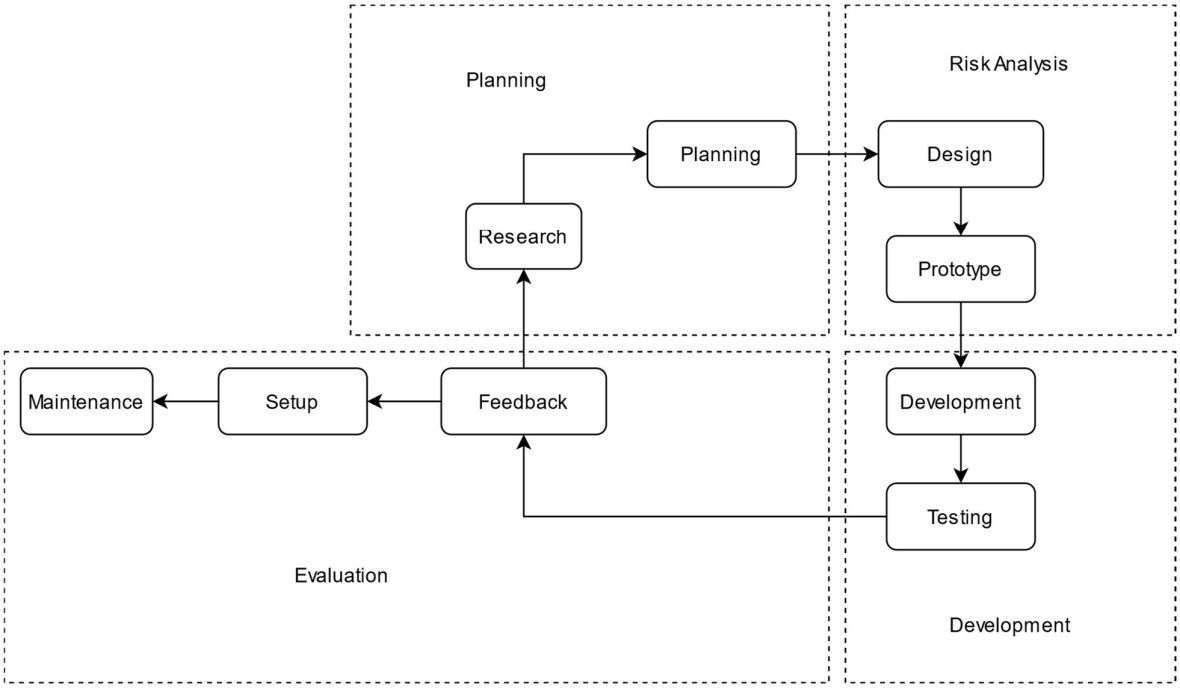


Figure 10: Spiral development methodology [72]

User acceptance testing is done during the evaluation phase, before the system is released.

Agile software development

Agile software development is not an SDM on its own, but rather refers to a collection of SDMs and practices based on the values of the Agile Manifesto. Examples include Scrum, Kanban and Extreme Programming [71]. A figure illustrating the process is not provided because of the wide variety of SDMs that fall under this category.

The manifesto encourages iterative development based on four values [74]:

- **"Individuals and interactions** over processes and tools" – Work is done in small, self-organising teams.
- **"Working software** over comprehensive documentation "– Working software that can be showcased to clients is more important than extensive documentation.
- **"Customer collaboration** over contract negotiation " – Agile relies on customer involvement throughout the development, as the requirements may be unclear at the beginning.
- **"Responding to change** over following a plan" – The development teams should be quick to react to changing requirements and feedback to ensure timely delivery of the final product.

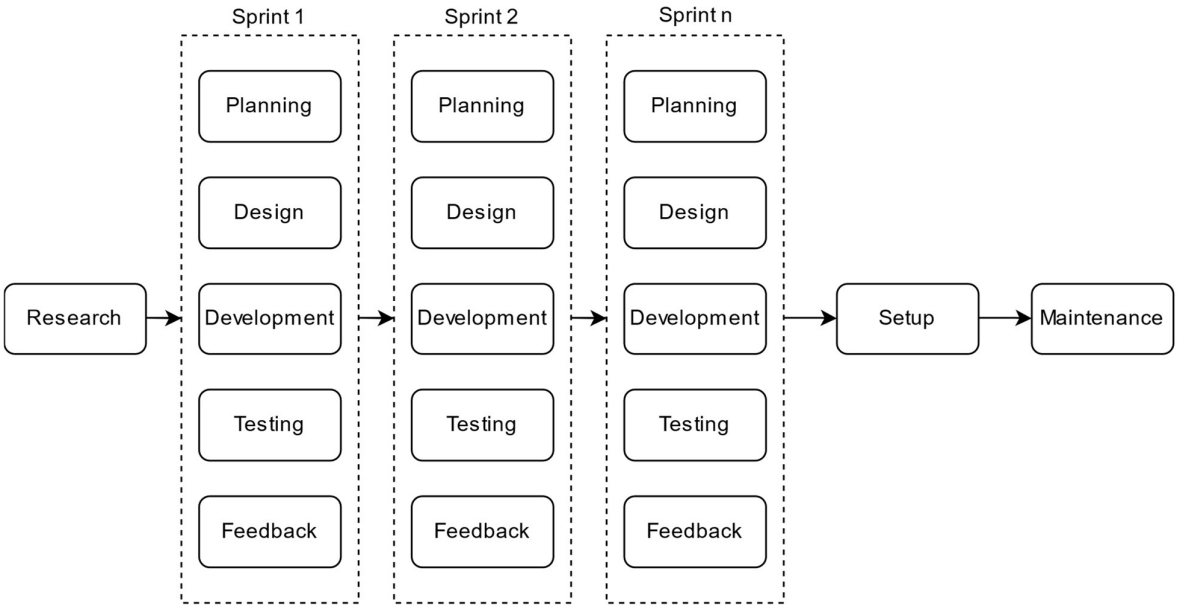


Figure 11: Agile development methodology – Scrum [72]

Figure 11 shows the Scrum design SDM. The software is developed in phases called "sprints", with each sprint typically lasting 2-3 weeks. The process starts with gathering requirements, planning, design, implementation and testing. After each sprint, an iteration of the product is completed which could be demonstrated and tested. The cycle continues until all the product requirements are completed [71].

User acceptance testing can take place after the initial product is complete and ready to be tested, after which another cycle of sprints addresses the shortcomings.

Software development methodology summary

Each Software Development Methodology (SDM) is divided into separate phases. User acceptance testing should generally be done as soon as a demonstrable product is ready. The changes could then be implemented according to the SDM in use.

2.5 State of the art: existing studies

The previous section discussed the role that acceptance testing plays in different software development methodologies. This section will investigate existing studies on technology acceptance and summarise the findings.

Research was done on previous studies that utilise the Technology Acceptance Model (TAM) to analyse or improve user adoption of technology. Most studies only focus on investigating the levels of Perceived Ease of Use (PEOU) and Perceived Usefulness (PU) and obtaining the relationships between the factors that influence them. The external factors differ between studies and are assigned based on the application.

14 studies that relate to the topic was investigated and compared. The factors that were utilised were focused on, as well as the number of respondents in the study. The case study is briefly discussed, and it was noted whether the study only did research on the current state of user adoption and perception of a system or if the results from the user adoption research were used to improve the system with the aim of increasing user adoption.

Perceived usefulness, perceived ease of use, and user acceptance of information technology

Davis [13] established the link between PEOU and PU on the behavioural intention to use the system. Two separate studies were done regarding software adoption. Two separate case studies were performed, involving 112 and 40 respondents respectively. In both cases, the findings were only analysed and not used to improve the user adoption.

Both studies successfully validated the impact of PEOU and PU on the behavioural intention to use the software. Self-reported usage figures were correlated with the PEOU and PU. The impact

of PU on usage was higher than that of PEOU. There is also evidence that PEOU influences the PU [16].

The following factors were tested:

- Perceived Ease of Use
- Perceived Usefulness
- Job effectiveness
- Productivity
- System importance
- Effort
- Learnability

User Acceptance of Computer Technology: A Comparison of Two Theoretical Models

Davis et al. [17] compared the Technology Acceptance Model to the Theory of Reasoned Action. The case study involved 107 respondents. The findings were only analysed and not used to improve the user adoption.

The conclusion from the study is that actual use could be predicted from the usage intentions. PU is a major determinant of people's intentions to use computer software, and the PEOU is a significant secondary factor.

Only PEOU and PU usefulness were tested.

An extension of the technology acceptance model in the big data analytics system implementation environment

Verma et al. [18] analysed the user acceptance factors of big data analytics tools. Characteristics of big data analytics tools were found to have a significant impact on PEOU and PU. The case study involved 150 respondents. The findings were only analysed and not used to improve the user adoption.

PU and PEOU influence the behavioural intention to use. System quality and information quality was found to be significant factors that impacted the PEOU and PU of big data analytics systems. Questions related to reliability and accuracy was used to assess the perceived information quality.

The perceived system quality was assessed using questions relating to accessibility, performance and flexibility.

The following factors were tested:

- Perceived Ease of Use
- Perceived Usefulness
- System quality
- Information quality
- Perceived benefits

Consumer adoption versus rejection decisions in seemingly similar service innovations: The case of the Internet and mobile banking

Laukkanen [75] investigated the possible factors that affect internet and mobile banking applications. The case study involved 1736 respondents. The findings were only analysed and not used to improve the user adoption.

The studies were split amongst adopters and non-adopters. The value barrier was found to be the most dominant barrier in all studies. An interesting conclusion is that the findings support the view that the system type will affect the user's adoption decision.

The following factors were tested:

- Perceived Ease of Use
- Perceived Usefulness
- Usage
- Value
- Risk
- Tradition
- Image

Checking the potential shift to perceived usefulness: The analysis of users' response to the updated electronic health record core features

Rajković et al. [37] used the information gained from a software user acceptance study to improve the acceptance rates of an electronic health record system by improving upon the areas which negatively affected the acceptance. The case study involved 195 respondents. The knowledge

gained was used to implement improvements on the system with the aim of increasing user acceptance.

The study focused on the user adoption of administrative software used by doctors and other medical personnel. An electronic health record system was used in the case study. Feedback on the current state of the system was garnered, and the information from the results was used to implement improvements on the system. User acceptance was monitored throughout the process.

Medical practitioners were more willing to use the system after the redesign and improved features. Thus, improving the PEOU and PU increased the intention to use as well as actual use figures. The increase in acceptance rate varied between departments, and the overall increase was 20%.

The following factors were tested:

- Perceived Ease of Use
- Perceived Usefulness
- Perceived Ease of Use impact on Perceived Usefulness
- Time
- Job relevance
- Standardisation
- System improvement
- Usage frequency
- Functionality

It was found that improving the PEOU of a system can convince more users to attempt to make use of the system and realise the benefits, thereby improving the PU of a system. This leads to an improvement in user adoption [37].

The technology acceptance model (TAM): A meta-analytic structural equation modelling approach to explaining teachers' adoption of digital technology in education

Scherer et al. [76] used TAM to explain the adoption of digital technology in education by teachers. The case study involved 34 357 respondents. The findings were analysed and not used to improve the user adoption.

Subjective norm positively impacted PEOU and PU. The study confirms the role of computer self-efficacy on PEOU and may represent a usage barrier. Facilitating conditions showed positive relations to PEOU and PU.

The following factors were tested:

- Perceived Ease of Use
- Perceived Usefulness
- Subjective Norm
- Computer self-efficacy
- Facilitating conditions

Technology acceptance model for the use of information technology in universities

Un Jan and Contreras [77] studied the acceptance of information technology in universities in two separate studies, involving 48 and 41 respondents respectively. In both cases, the findings were only analysed and not used to improve the user adoption.

Both studies involved engineering students. It was found that the PU was more important than PEOU as the students could overcome the limitations of poor ease of use as engineering students are generally proficient at operating computers.

The following factors were tested:

- Perceived Ease of Use
- Perceived Usefulness
- Subjective norm
- Compatibility

Factors influencing the adoption of Enterprise Social Software in Australia

Antonius et al. [78] studied the adoption of enterprise social software. The case study involved 300 respondents and the findings were not used to improve the user adoption.

Organisational factors and organisational culture were found to be the least influential factors. Individual factors influenced PU the most.

The following factors were tested:

- Perceived Ease of Use

- Perceived Usefulness
- Individual factors
- Organisational factors
- Task complexity
- Organisational culture
- Knowledge strategy

Assessing citizen adoption of e-Government initiatives in Gambia: A validation of the technology acceptance model in information systems success

Lin et al. [39] investigated the adoption of electronic government initiatives. 146 respondents participated in the case study and the findings were not used to improve the user adoption.

The link between behavioural intention to use and actual use was established. Information quality was found to have a strong influence on PU. In this case, the system was of poor quality, thus the system quality did not correlate strongly to PU.

The following factors were tested:

- Perceived Ease of Use
- Perceived Usefulness
- System quality
- Information quality

Exploring underutilisation of videophones in hospice settings

Day et al. [79] explored the user acceptance of videophone technology. The case study involved 14 staff members that were individually interviewed. The findings from the interviews were used to improve the user acceptance of the technology in question.

It was found that TAM provides a good framework for analysing and improving user acceptance. The main cause for a low acceptance rate, in this case, was a poor PEOU.

The following factors were tested:

- Perceived Ease of Use
- Perceived Usefulness
- Attitude
- Technical quality

- Usage barriers

Technology acceptance theories and factors influencing artificial Intelligence-based intelligent products

Sohn and Kwon [22] investigated the factors affecting the use and acceptance of artificial intelligence-based product. The case study involved 378 respondents. The findings were not used to improve the user adoption.

No external factors were used in assessing the software with regards to TAM, the research items from the original study [16] were used as-is. Various factors from TPB, UTAUT and the VAM was used alongside the factors from TAM.

The following factors were tested:

- Perceived Ease of Use
- Perceived Usefulness
- Attitude
- Subjective norm
- Perceived behavioural control
- Performance expectancy
- Effort expectancy
- Enjoyment
- Perceived fee
- Technicality
- Perceived value
- Intention to use
- Intention to purchase

Leading to Intention: The Role of Attitude in Relation to Technology Acceptance Model in E-Learning

Hussein [80] used the Technology Acceptance Model to analyse the impact of attitude on the acceptance of e-learning software. The impact of attitude was found to be significant, as most people already had knowledge of e-learning software and have a good PEOU.

The following factors were tested:

- Perceived Ease of Use
- Perceived Usefulness
- Attitude

An empirical investigation into the utilisation-based information technology success model: Integrating task-performance and social influence perspective

Kim et al. [81] investigated information technology success. The study does not utilise TAM, however, it further confirms the relationship between system performance expectancy and acceptance through utilisation figures. Furthermore, it does not apply the knowledge gained to improve the utilisation.

The following factors were tested:

- Performance expectancy
- Social influence
- User satisfaction
- IT utilisation

Technology Acceptance Model for Business Intelligence Systems: Preliminary Research

Bach et al. [43] investigated TAM with respect to business intelligence systems. The study provides a good theoretical basis for a model for estimating the user adoption of a potential business intelligence system that can be implemented. However, the model is more focused on the management of the implemented system, which may be out of the development team's control.

The following factors were tested:

- Perceived Usefulness
- Perceived Ease of Implementation
- Information Quality
- Implementation (System quality)
- Project management
- Change management

- Knowledge sharing

Discussion of existing studies

Table 1: State of the art overview

	Technology Acceptance Model		Quality		Implementation	
Source	Perceived Usefulness	Perceived Ease of Use	System Quality	Information Quality	Measure	Improve
Davis [13]	■	■			■	
Davis et al. [17]	■	■			■	
Verma et al. [18]	■	■	■	■	■	
Laukkanen [75]	■	■			■	
Rajković et al. [37]	■	■	■		■	■
Scherer et al. [76]	■	■			■	
Un Jan, Contreras [77]	■	■			■	
Antonius et al. [78]	■	■			■	
Lin et al. [39]	■	■	■	■	■	
Day et al. [79]	■	■	■		■	■
Sohn, Kwon [22]	■	■			■	
Hussein [80]	■	■			■	
Kim et al. [81]					■	
Bach et al. [43]	■	■			■	

Table 1 shows a summary of the state of the art for the application of TAM in research studies. Very few of the studies use the information gained through the research to improve the user adoption of the system in question. Thus, there is an opportunity to develop a methodology for improving user adoption by improving the key areas where user acceptance is influenced negatively. This can either be done constantly throughout the development process or at the end during the maintenance phase.

After comparing existing studies that utilise the Technology Acceptance Model, it was confirmed that Perceived Ease of Use and Perceived Usefulness play a role in software user acceptance. The external factors vary between studies according to the specific needs of the study, and there is no consistent method for choosing the factors to analyse from an engineering perspective. This is considered a strength of TAM, as it enables the user to extend the model with external factors related to the use case [39]. A software quality model containing factors that can be applied to TAM will be utilised.

The next aspect to investigate is the factors that are tested in each study. The external factors differ between studies utilising TAM [39]. For a study related to Business Intelligence (BI) or data analytics software, the system quality and information quality was investigated. Verma et al found that system and information quality could have an influence on user adoption of a BI system [18].

The first factor that was investigated is related to system quality [18]. According to Seddon and Kiew , “System Quality is concerned with whether or not there are "bugs" in the system, the consistency of the user interface, ease of use, response rates in interactive systems, documentation, and sometimes, quality and maintainability of the program code”. This is especially relevant to software development as it can be directly improved by a development team and was found to have a significant impact on software user acceptance [18], [39].

The second factor that occurs frequently in studies related to user acceptance of software is information quality [18], [82]. Information quality is closely related to system quality and refers to the quality of the system output in the form of processed results and reports, as well as the information that the system uses as an input to produce results. This can also refer to the accuracy, reliability and completeness of the results. Information quality plays a big role in the user acceptance of data analytics and will be included in the study [18], [83].

There are also very few cases where the information gained is applied to improve the specific areas where the software is found to be lacking. By using the insights from user acceptance

research, the system can be improved with the overall goal of improving user adoption of the software system over a certain period, as demonstrated by Rajković et al [37].

The following section summarises the background and literature, followed by a problem statement.

2.6 Summary and problem statement

Industry 4.0 depends on the efficient processing of large amounts of data. Data analytics systems are the primary method of processing data for the extraction of information.

Some users prefer to use suboptimal software because of personal preference, or because of low Perceived Usefulness (PU) or low Perceived Ease of Use (PEOU). As a result, user acceptance of the software system is affected.

User acceptance of software is important, as a lack of software usage or the use of suboptimal software results in the following:

- Loss of efficiency on a management level.
- Efficiency is lost on an employee level.
- Wasted development time and resources.

All the above-mentioned factors could negatively affect profitability of an organisation and should be prevented by ensuring optimal user acceptance of the software system in question. A high acceptance rate of data analytics software along with the proper use and utilisation is important in maximising the potential benefits that data analytics software brings.

Technology acceptance was investigated and various models for analysing technology acceptance were compared. The Technology Acceptance Model (TAM) was found to be the best model for a general methodology to improve user acceptance of software. TAM provides the opportunity to analyse a system using factors selected by the researcher and could be applied in a general methodology to improve user acceptance of software systems. It compares favourably to similar models, as evidenced by the myriad of literature related to TAM.

An overview of existing studies that utilise TAM was provided. It was found that the acceptance rates of software can be analysed by the Technology Acceptance Model, but very few studies use the information gained through analysis to improve the shortcomings of the software to increase user adoption. As a result, there is an opportunity to develop a general methodology to increase user acceptance by using the principles outlined by TAM.

Software factors that could affect user acceptance were investigated and discussed.

From the summary above, the following problem statement can be derived:

User acceptance of software may be affected by personal preference or poorly designed software. Using suboptimal software instead may affect the efficiency of a workforce or decrease the profitability of a company as resources spent developing the software is wasted. User acceptance of software need to be improved by identifying and improving the factors that negatively affect user acceptance.

CHAPTER 3: METHODOLOGY

3.1 Preamble

The literature discussed in the previous chapter highlighted the importance of user acceptance of software systems, especially within an organisation. TAM was discussed, as well as the impact that system and information quality have on software adoption. Existing literature on user acceptance was studied, and it was found that very few studies implemented improvements based on the information gained through user acceptance feedback.

In this chapter, the findings from the literature review are used to develop an improved methodology to increase user adoption of software systems by increasing factors related to PU and PEOU of the system.

3.2 Methodology overview

The objective of the methodology is to provide a method to improve user acceptance of software systems by analysing and improving factors related to PU and PEOU.

Each step of the methodology will be given in this overview, followed by the section that the step is discussed in.

Step 1: Software system identification

The purpose of the methodology is to improve the user acceptance of an underused system; thus, a system needs to be selected. The current acceptance rate of the system should be noted, as this will form the baseline against which improvements will be measured (discussed in section 3.3).

Step 2: Selection of factors to analyse

The software factors that will be analysed are then selected. The factors from ISO 25010 will be used to investigate different aspects of system and information quality [18] (discussed in the second half of section 3.3).

Step 3 and 4: Questionnaire development and distribution

The groups of factors are then used to form a research instrument in the form of a user survey. Questions relating to each factor will be asked, and the results will be aggregated to form a general overview of the users' perception of the software system (discussed in section 3.4). Before the results of the first set of surveys are interpreted, the current number of users will be noted [37].

Step 5: Analyse the results of the questionnaire

Cronbach's alpha will be used to check the consistency of the results [84] (discussed in section 3.5). After verifying that the data is consistent, the results from each user will be combined using simple scoring (discussed in section 3.5), so that a general overview of the factors related to PU and PEOU of the system is obtained [85].

Step 6: Improve software factors based on results

The information gained from the results will be used to suggest a course of action to improve the software in the areas where it is lacking (discussed in section 3.6). Specific software aspects will be focused on to improve the user's PU and PEOU of the system [37].

After the improvements have been completed, a few months will be allowed to pass to allow potential users to become active users. The number of active users after the improvements will be compared against the original number, and the user acceptance rate from both will be compared [37] (discussed in section 3.7).

Step 7: Re-evaluate the software acceptance

After a sufficient period has passed, the same research questionnaire will be sent out to all of the current users (discussed in section 3.7). The results will be analysed, compared and discussed (discussed in section 3.7). The number of users will also be compared to the original amount. A conclusion regarding the effectiveness of the methodology will be drawn from the results [18], [37].

Step 8: Validate the results

Validate that the methodology was effective in increasing user acceptance. This is done by looking at usage numbers, as well as comparing the original factor scores with the post-improvement results.

3.3 Software and factor selection

Selection of software to improve

The first step in the methodology is to identify a software system with poor user adoption. Depending on the Software Development Methodology used to develop the system, the software could be in a finished state where it is underutilised, or in the prototype or early testing phase of the development cycle.

Consider the following system as an example: A system with an excellent user interface and good functionality suffers from low user acceptance, and the development team wants to investigate the cause. In this example it will be assumed that the system only has ten active users, but has 30 potential users. The current user acceptance rate will be obtained by dividing the amount of active users by the amount of potential users. In the example, the user acceptance rate is 33.3%

Software quality factors

As discussed in section 2.3, the software factors from the ISO 25010 model for software quality will be used to form the software characterisation model. The main factors are the following [53]:

- Functional suitability
- Performance efficiency
- Compatibility
- Usability
- Reliability
- Security
- Maintainability
- Portability

These eight factors can be divided into several subfactors which can be seen in section 2.3 [59]:

To get accurate feedback from the users, the questions must be properly phrased. If the questions are vague or ambiguous, the results may become skewed. To this end, the factors must be clearly defined and understood before the list of questions can be created. The following definitions are sourced from the official literature for ISO 25010. Where unclear, the definitions are expanded to provide a clear understanding from the perspective of the user [57].

Functional Suitability

Functional suitability refers to “the degree to which a system provides functions that meet the stated and implied needs when used under specified conditions.” Table 2 shows the subfactors associated with functional suitability [57]:

Table 2: Subfactors related to functional suitability

Subfactor	Definition
Functional completeness	Refers to “the degree to which the functionality covers all the required tasks, and the ability to achieve the outcome desired by the user”.
Functional correctness	Refers to “the correctness or accuracy of the results provided by the system”.
Functional appropriateness	Refers to “the degree to which the system facilitates the completion of specified tasks”.

Performance Efficiency

Performance efficiency refers to “the performance relative to the number of resources utilised under stated conditions.” Table 3 shows the subfactors associated with performance efficieny [57]:

Table 3: Subfactors related to performance efficiency

Subfactor	Definition
Time behaviour	Characterises response times for a given input (button press, action performed etc.).
Resource utilisation	Characterises resources used, i.e. memory, CPU, disk and network usage. This can also include time and human resources.
Capacity	“The degree to which the maximum limits meet the requirements of the user”.

Compatibility

Compatibility refers to “the degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions while sharing the same hardware or software environment.” Table 4 shows the subfactors associated with compatibility [57]:

Table 4: Subfactors related to compatibility

Subfactor	Definition
Co-existence	"The degree to which the system can perform its functions correctly in the presence of other systems while sharing the same resources".
Interoperability	"The degree to which the system can correctly interchange information with other systems and utilise this information correctly".

Usability

Usability refers to "the degree to which a system can be used by specified users to achieve specific goals with effectiveness, efficiency and satisfaction". It also refers to the capability of the software to be understood and learned. Table 5 shows the subfactors associated with functional usability [57]:

Table 5: Subfactors related to usability

Subfactor	Definition
Appropriateness recognisability	"The degree to which a user can recognise that the system is appropriate for their needs".
Learnability	Learning effort for different users, i.e. novice, expert, casual etc.
Operability	"The ability of the software to be easily operated by a given user in a given environment".
User error protection	"The degree to which a system protects users against making errors".
User interface aesthetics	Refers to how aesthetically pleasing the software is, as well as the general layout.
Accessibility	"The degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use".

Reliability

Reliability of a product refers to "the degree to which a system, product or component performs specified functions under specified conditions for a specified period". It is divided into the following Table 6 contains the subfactors associated with reliability [57]:

Table 6: Subfactors related to reliability

Subfactor	Definition
Maturity	This concerns the frequency of failure of the software.
Availability	"The degree to which a system, product or component is operational and accessible when required for use. Can refer to the percentage amount of uptime".
Fault tolerance	"The ability of the software to reduce system errors or withstand component failure".
Recoverability	"The degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system".

Security

Security refers to "the degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorisation." Table 7 shows the subfactors associated with security [57]:

Table 7: Subfactors related to security

Subfactor	Definition
Confidentiality	"The degree to which a product or system ensures that data are accessible only to those authorised to have access".
Integrity	"The degree to which a system, product or component prevents unauthorised access to, or modification of, computer programs or data".
Non-repudiation	"The degree to which actions or events can be proven to have taken place so that the events or actions cannot be repudiated later". Thus, the traceability of actions, and the inability to remove these traces.
Accountability	"The degree to which the actions of a user entity can be traced uniquely to the user".
Authenticity	"The degree to which the identity of a subject or resource can be proved to be the one claimed".

Portability

Portability is “the degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.” Table 8 shows the subfactors associated with portability [57]:

Table 8: Subfactors related to portability

Subfactor	Definition
Adaptability	“The degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments”.
Installability	“The degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment”.
Replaceability	“The degree to which a product can replace another specified software product for the same purpose in the same environment”.

Factor grouping

TAM suggests two major factors that affect acceptance: Perceived Ease of Use (PEOU) and Perceived Usefulness (PU) [13]. In the studies that analyse technology adoption, factors are investigated that could either be classified as relating to PU or relating to PEOU. Before the factors that will be investigated could be classified into one of these groups, there needs to be clear definitions for the requirements of factors that pertain to either group.

As per the definition of PU in chapter 1, PU is the degree to which a potential user believes that a piece of technology would enhance productivity. Thus, any software characteristic that will improve the quality of the work that the user can do with the system will be classified under PU.

The definition of PEOU refers to “the degree to which a person believes that using a particular system would be free of effort” [13]. Features that make the system easier or more intuitive to use would thus fall under PEOU.

Each factor, along with its subfactors were categorised based on the impact on PEOU and PU according to the requirements stated above.

Functionality was included in the PU group, as the core functionality and capabilities of a system play a considerable role in the PU of a system. The reliability of a system is also included here as

a system that is robust and stable to use also influences the PU. Compatibility also falls under this category as data exchange between systems could be very useful. Lastly, security could be useful in controlling access to information and functionality.

The following factors are related to PU:

- Functional suitability
- Reliability
- Compatibility
- Security

The first factor to be included in the PEOU group is usability. The usability factor refers to the ease of use of a given function and incorporates subfactors like learnability, operability and user error protection to gauge usability. The user interface and accessibility features of a system also contribute greatly towards PEOU. Performance efficiency and general system responsiveness contribute towards PEOU. Lastly, portability of the system and the ability to access the system from a variety of platforms also affects PEOU.

The factors related to PEOU are:

- Usability
- Performance efficiency
- Portability

The relationship between user acceptance and all of the subfactors are illustrated in Figure 12:

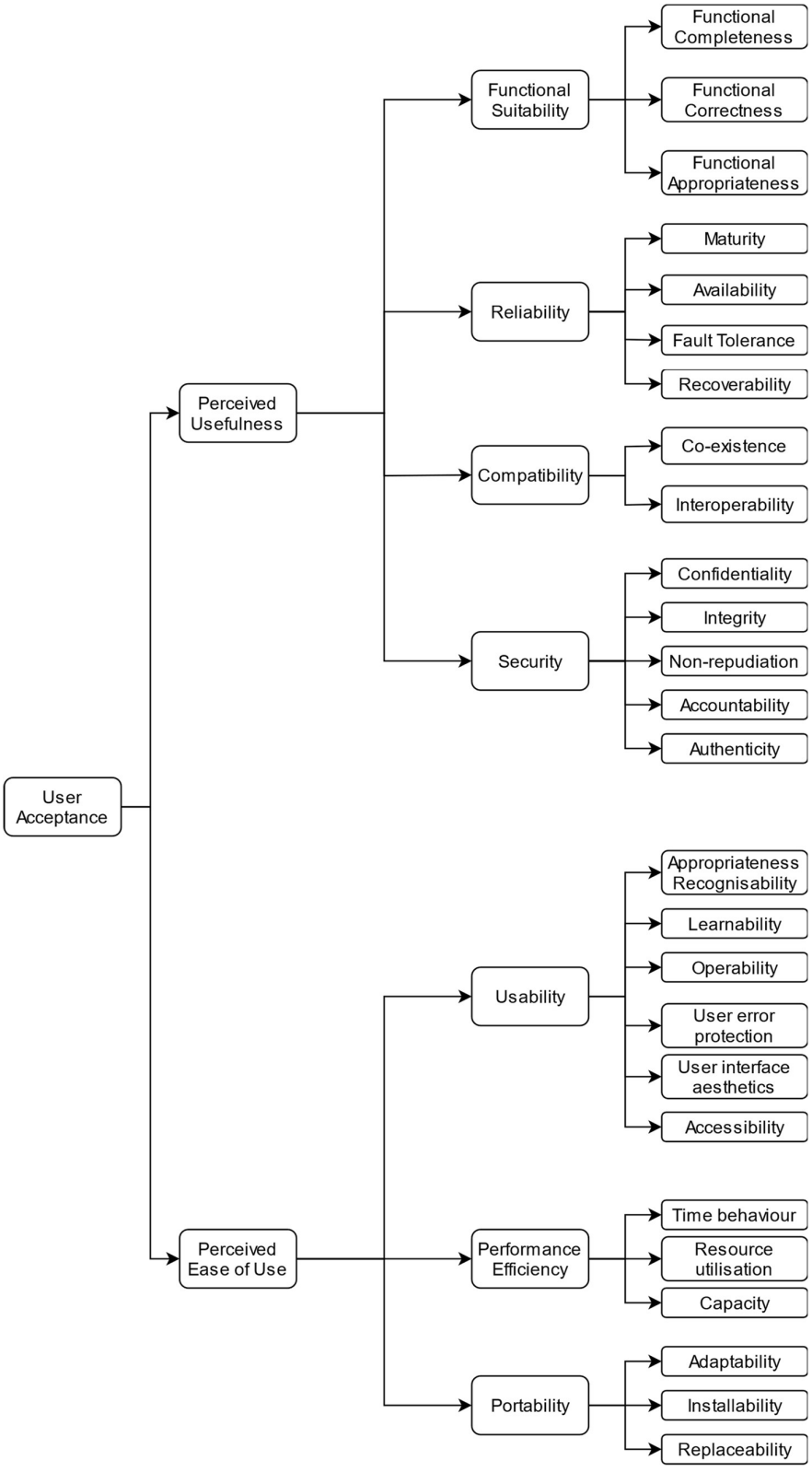


Figure 12: Mind map of subfactors

3.4 Development and distribution of questionnaire

Questionnaire theory

The next step is to develop a questionnaire based on the factors selected in the previous step of the methodology. It will be used to obtain information about the current state of the system.

Likert scale

The Likert scale method will be used in the survey. Likert scales are the most widely used psychometric scale in research involving user surveys. It provides the following benefits [86]:

- It is easy to construct and modify a Likert scale.
- Numerical results obtained via the Likert scale can be directly used for statistical inference.
- Results obtained via the Likert scale generally have good reliability.

Positive bias

When developing Likert scale items, it is important to consider the wording bias direction for each question. Questions with a positive wording bias refer to questions that prompt the reader's agreement towards something that the system in question does well. An example of a positive bias question is: "The system is **easy** to use" [16].

The opposite of the above question would be to rephrase the question with regards to negative bias. The result would be: "The system is **not** easy to use", or "The system is **hard** to use" [16].

Questions regarding similar factors should always be asked with the same bias to avoid skewing the results and affecting the variance and correlation between questions.

Question construction

A research questionnaire will be developed that consists of items adapted from the 25 subfactors selected in the previous step. The questionnaire will be used to garner feedback from current users about the state of the system in its current form. Respondents that have existing experience with the system of at least a few hours will be selected. The insights gained will be used to prioritise factors which will be improved to increase user adoption

A 6-point Likert Scale will be utilised in the questionnaire. Each question should be asked as a declarative statement, to which a respondent may answer with his level of agreement. There are five possible answers to each question. The answers will later be translated to a rating from one to five. In this case, an answer of “Strongly Disagree” would be a score of one, and “Strongly Agree” would correlate to a score of five. Table 11 in section 3.5 shows the score mapping for all of the answers in more detail.

The survey will be split into two sections. Table 9 shows the questions related to Perceived Usefulness (PU) that will be used in the research questionnaire:

Table 9: Survey questions related to PU

Factor	Subfactor	Question
Functional suitability	Functional completeness	The system provides all the required functionality.
	Functional correctness	The system produces the correct results, and the results are accurate.
	Functional appropriateness	The system provides the correct functionality to complete the specified task.
Reliability	Maturity	The system does not crash or fail frequently.
	Availability	The system is always available.
	Fault tolerance	The system handles errors gracefully.
	Recoverability	Work can easily be recovered in the event of a crash.
Compatibility	Co-existence	The system can be used alongside other software with no issues.
	Interoperability	The system can exchange accurate information with other systems and use this information correctly.
Security	Confidentiality	The work completed on this system is only accessible by users with authorisation.
	Integrity	The system prevents unauthorised access and modification of data.
	Non-repudiation	Actions taken while working on the system are traceable and these traces cannot be removed without authorisation.

	Accountability	Actions can be traced to a unique user or entity.
	Authenticity	The source of the data provided by this system can be proven.

The questions for each factor related to Perceived Ease of Use (PEOU) are listed in Table 10:

Table 10: Survey questions related to PEOU

Factor	Subfactor	Question
Usability	**Appropriateness recognisability**	This system is easy to understand and intuitive to use.
	Learnability	It is easy to learn how to use the system.
	Operability	This system is easy to use to get the intended results.
	User error protection	The system helps the user to provide the correct input.
	User interface aesthetics	The system is well organised and nice to look at.
	Accessibility	The system can be used by people with a wide variety of skill levels.
Performance efficiency	**Time behaviour**	The system responds quickly to any input.
	Resource utilisation	The system uses resources (e.g. CPU, RAM, Network) efficiently.
	Capacity	The system can handle large workloads or many concurrent users without issues.
Portability	**Adaptability**	The system works well on different platforms and devices.
	Installability	The system is easy to install or access.
	Replaceability	The system can replace similar systems in the same environment for the same purpose.

The validity of the questionnaire could be validated using face validity, which refers to verifying that the questions are suitable to represent each subfactor.

The response rate of this study should ideally be above 50% of the current users.

Distribution of questionnaire

The next step is to distribute the questionnaire created in the previous step of the methodology to users. The questionnaire should be distributed only to users that have prior experience with the system. However, if users that do not have experience with the system are asked to provide feedback, they should be proficient with using computers to be eligible to provide feedback. Users will be eligible if they have a Bachelors degree in a technical field such as engineering or information technology [13].

The process for requesting user feedback is as follows: The questionnaire is sent to users in the form of a Google Forms online form, followed by a waiting period of a week. After a week has passed, the response rate of the questionnaire is evaluated. If the response rate is below 50% of all users, the questionnaire will be sent again, followed by another week's waiting period.

The results are compiled automatically using Google Forms, where the results are available for download in the form of a comma-separated value(.csv) file format. The downloadable .csv file containing all results provides a convenient method for accessing the results to analyse the data. The results can either be analysed manually using Excel formulas, or automated using a scripting language suited for data analysis, such as Python and R.

Ethics

The ethics requirements of the North-West University Engineering Research Ethics Committee (NWU-ENG-REC) will be followed. More information can be found online[2].

The author of this paper has successfully completed the ethics training provided by the NWU-ENG-REC, and has received ethics clearance for the research questionnaire to be provided to participants.

[2] http://engineering.nwu.ac.za/engineering-research-ethics-committee/about-nwu-eng-rec

3.5 Result analysis

Categorical answer to interval mapping

To convert the categorical results obtained by the Likert scale data to numeric results, an ordinal to interval scale conversion is needed [87].

Table 11 shows the scores allocated to each answer level:

Table 11: Likert scale score map

Answer	Score
Disagree completely	1
Disagree	2
Neutral	3
Agree	4
Agree completely	5

Where a respondent leaves out an answer, the respondent's answer to the questions regarding the factor in question will not form a part of the statistic but reported separately. Where a respondent has not provided a valid answer for half or more of the questions referring to a specific factor, the remaining answers for that factor by the respondent will be ignored when calculating statistics.

Variance

The variance is calculated as follows:

$$\sigma^2 = \frac{\sum_{i=1}^{k}(x_i - \mu)}{k - 1} \quad (4)$$

where:

- k represents the number of items

- x_i represents the item
- $\boldsymbol{\mu}$ represents the mean of all items

Cronbach's Alpha

Cronbach's alpha will be used to check the correlation of results. This method is defined as the "internal consistency" of data, and it quantifies the degree of correlation of a set of test items. It can also be referred to as the extent to which the measurement is a consistent measure of a concept. It will be used to test the consistency of questions between subfactors[3].

The formula for Cronbach's alpha is given as follows [84]:

$$\alpha = \frac{k}{k-1}\left(1 - \frac{\sum_{i=1}^{k}\sigma_{y_i}^2}{\sigma_x^2}\right) \tag{5}$$

where:

- k represents the number of items
- $\sum_{i=1}^{k}\sigma_{y_i}^2$ represents the sum of the variance of each of the item results
- σ_x^2 represents the variance of the total scores of each item in the test added together

The result is given as:

$$\alpha \leq 1 \tag{6}$$

An alpha value closer to one signifies stronger correlation between data items. However, values very close to one, such as values over 0.9, may indicate redundant questions in the research questionnaire regarding the factor in question, meaning that more than one question is present that refers to the same factor.

Researchers are divided by the question of what constitutes an acceptable alpha value. Some argue that 0.7 and above is preferred, while others accept a minimum value of 0.45 as acceptable.

[3] "Using and Interpreting Cronbach's Alpha | University of Virginia Library Research Data Services + Sciences." https://data.library.virginia.edu/using-and-interpreting-cronbachs-alpha/ (accessed Apr. 07, 2020).

The alpha value may also be low as a result of a low amount of responses compared to the amount of questions or a low number of items describing a specific factor. In that case, other statistics could be used to gain information about the answers. It could also be argued that adding more items to the survey referring to the specific factor may improve the alpha value [29], [88], [89].

For this study, a value of 0.45 will be considered acceptable. If a lower consistency value is obtained, other statistics of the data in question will be investigated to see if the information gained provides meaningful information.

Aggregation score of each factor

By measuring the individual factors, a total measure of software quality and thus PEOU and PU can be extracted. Two popular methods of the aggregation of software quality scores into a single score is simple scoring, as well as weighted scoring [85].

Simple scoring is the simplest method of the two. The overall score for a quality factor is given by the mean of the individual subfactor scores. This assumes that each individual subfactor has the same weight and is the easiest to implement without advanced statistical methods like factor analysis [85].

Weighted scoring is another option and requires assigning a relative weight to each subfactor. This allows the researcher to weight each subfactor according to how important they consider them to be. However, using subjective measures may influence the accuracy of the results [85].

For a general method to measure the users' perception of the system, it will be assumed that each subfactor and factor carry equal weights, thus the simple scoring method will be used. This is the simplest method to gain an overview of the current state of the system.

Example:

Consider the example, given in Table 12, of the scores given by users for the subfactors of the usability group:

Table 12: Example results for usability

Subfactor	User 1	User 2	User 3	Mean
Appropriateness	3.3	3.4	3.8	3.5
Learnability	4.0	4.1	3.9	4.0
Operability	3.9	4.1	4.3	4.1
User error protection	4.2	4.3	4.7	4.3
User interface aesthetics	4.5	4.3	4.4	4.4
Total score for Usability				4.06

Calculating the mean of the scores in column four of Table 12 yields a score of 4.06 for usability. This is repeated for all of the answers for all factors.

The same method will be used to gain an overview of the current state Perceived Ease of Use (PEOU*)* and Perceived Usefulness (PU) of the system. The mean of all the factors within the two groups established earlier in the thesis will be used as the mean score for PU and PEOU and will give a good indication of how the users perceive the system.

Returning to the use of an example system, a possible result from user feedback can yield the results show in Table 13:

Table 13: Example of questionnaire results

Factor	Average initial user rating
Functional Suitability	4.1
Reliability	2.75
Compatibility	3.8
Security	3.54
Usability	4.15

Performance Efficiency	3.566667
Portability	3.5

By using the factor groupings discussed earlier, the mean of the factors related to PU is 3.55 and the mean of the factors related to PEOU is 3.73. Thus, the users have a higher overall PU than PEOU.

The Cronbach's Alpha value can be calculated for all of the results from each user per subfactor. For example, if the amount of questions is 3, the sum of variances is 2.2 and the variance of the sum of all scores is 5.04 then the Cronbach's Alpha value would be the following:

$$\alpha = \frac{k}{k-1}\left(1 - \frac{\sum_{i=1}^{k}\sigma_{y_i}^2}{\sigma_x^2}\right) = \frac{3}{3-1}\left(1 - \frac{2.2}{5.04}\right) = 0.84 \quad (7)$$

This would indicate good correlation between users, and the results can be accepted.

3.6 Improvement of software

Factor selection

The factor that has the lowest average rating will be the primary focus of the improvements. This will have the most significant impact on Perceived Usefulness (PU) and Perceived Ease of Use (PEOU).

Where possible, some of the subfactors from other factors that received extremely low scores will be improved as well. For example, when improving the usability by modifying the user interface, additional security could be implemented through the addition of user access control to the interface. Similarly, when improving the compatibility of the system, effort could be made to properly optimise some of the code, thereby also improving performance efficiency.

Summary of improvement methods

The following methods could be utilised to improve the lowest scoring factor, after which the updated system will be released to the users. After the updated system is released, a training session could be held to inform the potential users about the updated capabilities and improvements made to the system.

It is important to note that focusing on improving a single factor could lead to a ripple effect that improves all factors. This is because factors are not completely isolated, and many factors are inter-related. For example, according to Avouris, functionality, reliability and efficiency may be affected by usability. This also correlates with the notion provided by Rajković et al. that an improvement in PEOU may affect PU [37], [63].

Methods to improve each factor were researched and are summarised below. The methods need to be feasible within the current constraints of the software system being improved, and need to be discussed with technical personnel or stakeholders where applicable.

Functional suitability

Enhancing the functionality of software systems is sometimes necessary to better comply with the needs of the customer. Customers frequently experience that 20% or more of the required functionality is missing from the software, and as such the development team often need to expand the available functionality [90]. The alternatives include modifying business practices and using additional software, which could waste valuable time and resources on the client's end. Specifications could be altered or expanded due to feature creeping, further driving the need to improve the functionality [90].

A method for software functionality improvement is to extract features from user feedback. The features that could improve users' perception of the software is analysed and a score is given to each feature. The features that will have the biggest impact will be prioritised. The features are then implemented, refined and optimised to best suit the needs of the users, followed by monitoring of user feedback to verify that the functionality has improved [91].

Feature request could also be monitored for popular requests, followed by prioritisation of the most important missing features. The new feature or enhancement to existing features will then be added, thereby improving the functional completeness and functional correctness of the software towards a specific purpose. This could expand the target audience or improve the experience of current customers [90].

Reliability

Software reliability is different from hardware reliability in the sense that software cannot experience wear or degradation over time. The software does not fail unless the software itself contains flaws. Thus, the primary method of ensuring software reliability should be to ensure that the software system itself does not contain flaws.

During the testing phase of software development, reliability tests such as integration and unit tests should be implemented to help identify issues before they occur in a production environment. Critical failures are identified during testing, and the failure data is analysed. The data could be used to improve the reliability of the system before release [92], [93].

Improving the reliability of existing software starts with analysing the current state of software reliability. An important measure of reliability is asymptotic measurement metrics. Chen's method, as well as Pham's method of failure rate analysis, can be used to assess reliability [94]. Reliability could be improved by improving fault-detection capabilities, implementing a process restart feature and reducing the software failure rate. Process restarts may reset the memory usage of software systems with inadequate memory management, and can improve reliability [94].

Compatibility

Ensuring compatibility between components within a software system as well as between different systems is a complex task and requires proper testing alongside the development. Software compatibility testing is a quality assurance task that helps to ensure that multi-component systems work correctly across different combinations of versions or configurations.

The following tools and techniques are used to improve compatibilities between software [95]:

- **The use of interconnection standards:** Standard application data interface definitions such as APIs that ensure that the correct data is sent and received and that the data is in the correct format.
- **Configuration management tools:** Enables changes and deployments to be faster and more predictable.
- **Service-oriented architectures:** Services are provided to other applications and components through communication protocols. Services can communicate directly to each other or through a central service that controls the flow of information.

- **Middleware frameworks:** Frameworks that convert information to a common standard through the use of APIs to facilitate data exchange.

Security

Due to agile business strategies, security is usually the last aspect to check when an application is developed and can often be overlooked or ignored due to time constraints. However, security in software systems should be planned and designed in every stage of the development process to ensure optimal returns in terms of cost and resources [96].

DevOps is a set of practices that combine the software development process (Dev) with IT operations (Ops) to increase an organisation's ability to deliver services as efficiently as possible. When security is the main focus of a DevOps team, it is referred to as DevSecOps. Through DevSecOps, processes and tools could be put in place to automate workflows that will perform automated security testing and help identify flaws in the system. The flaws could be identified much quicker than by manual inspection, and with good accuracy. This ensures more secure and stable software [96].

Usability

Usability can be improved by first setting a goal for overall usability, then evaluating the key aspects of usability. The mean values of the ratings should be compared against the target that was set earlier.

It is important to realise that the usability of a system is not a single one-dimensional aspect of a system, but rather a combination of factors such as operability, learnability, user interface aesthetics and more. As a result, it is important to focus on multiple aspects when improving the usability of a system.

Nielsen suggests the following focal points for increasing the usability of a system [97]:

- **Use simple and natural wording:** The user interface should never display information that is not needed or is irrelevant to the current task. Information should appear in a natural, logic order.

- **Use familiar language:** Dialogue should be expressed in familiar phrases and concepts.

- **Minimise memory load:** The layout should be as simple as possible while providing all functionality. Users should not have to remember information between sections. Instructions should be visible or easily retrievable.

- **Consistency:** Users should not have to wonder whether different words or actions mean the same thing.

- **Feedback:** The system should keep users informed about what is going on.

- **Clearly marked exits:** Users often choose actions unintentionally and need a clear way to go back to the previous state.

- **Shortcuts:** Advanced methods for expert users that could speed up navigation and executing actions such as keyboard shortcuts may speed up workflow dramatically.

- **Good error messages:** When an error occurs, the user should be notified in plain language about the problem and the system should suggest an appropriate solution.

- **Prevent errors:** An even better practice is to not have errors in the first place by designing the system carefully.

- **Help and documentation:** Documentation should be focused on the user's task, easily accessible and easy to search.

Performance Efficiency

Software performance is a pervasive quality influenced by lots of underlying factors, ranging from the design architecture, quality of the code and environment in which the code is running [98].

There are two general approaches to improve the performance of a software system. The first is a measurement-based approach, followed by a model-based approach. The measurement-based approach involves rigorous testing, diagnosis of performance bottlenecks and tuning the system to improve performance efficiency. This usually takes place late in the development cycle and is the most common approach to performance improvement [98].

The model-based approach relies on the development of performance models very early in the development cycle. These models are used to obtain quantitative data which could be analysed to suggest improvements to the underlying system architecture and design [98].

The software performance improvement process may include some or all of the following activities

[98]:

- **Identify possible concerns:** Qualitative analysis of factors affecting performance negatively.
- **Define and analyse requirements:** Define the required performance level of the system components.
- **Predict performance:** Analyse the architecture and design's predicted performance by modelling the system in different scenarios.
- **Performance testing:** Test the system under different scenarios.
- **Maintenance and system changes:** Improve the system as required.
- **Total system analysis:** Compare the planned system to the final system's performance.

Portability

Portability usually requires extra planning ahead of the initial design to decide where the system will need to be used. As a result, resources are rarely dedicated to achieving good portability. Johansson et al. attribute this to the lack of documented portability strategies and problems [99].

The three most important points for improving portability is the following [99]:

- **Control the user interfaces:** User interfaces vary between platforms and form factors, and as a result, the user interface design is very important. The underlying logic could be easily ported across platforms, but the user interface needs to adapt and still provide the same level of functionality in each case.
- **Isolate dependencies:** Components will often need conversion to be ported across platforms, and as a result, will need to be isolated from each other and the user interface where possible.
- **Think portable:** If portability may add value to the system, it should be kept in mind when designing the system. Portability should be considered as early as the requirements phase, throughout the design, testing and implementation phases.

Web-based software systems also provide a good mix of portability and functionality and could be developed as standard web apps, web-based hybrid mobile apps as well as progressive web

apps.

Modern web apps are developed with portability in mind and are hosted on remote servers, served via standard protocols such as HTTPS. The widespread compatibility of the Blink (Google) and WebKit (Apple) rendering engines provide excellent cross-platform portability, while common languages such as HTML and JavaScript provide fast development turnover while ensuring widespread compatibility [100].

Example results:

Continuing with the example used throughout chapter 2 , the following scenario demonstrates a typical case where the methodology is used to improve user acceptance of a system and shows the results that the methodology is expected to provide.

Information about the current state of the system was obtained by providing existing users with the questionnaires developed.

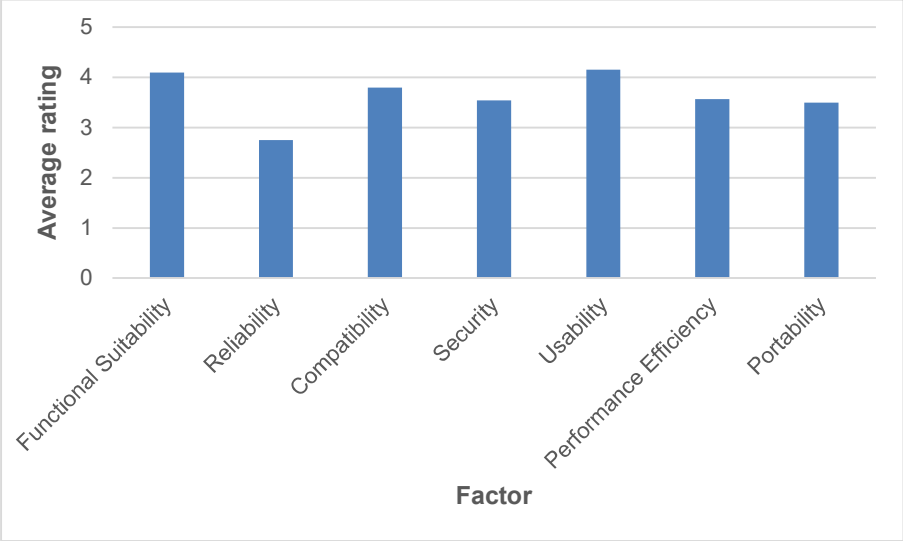


Figure 13: Example of expected initial results

Figure 13 shows an example of results that were shown as an example in Table 13. All of the factors received good scores except for reliability. Reliability achieved a much lower score than all of the other factors and could be the reason for the low user acceptance figures.

By using the methods discussed earlier, reliability can be improved. It may occur that other factors are also improved while focusing on reliability improvements, as the factors are not completely isolated from each other.

After improving the reliability of the system, the same questionnaire will be distributed to users to obtained information about the updated system. A drastic improvement in the average user rating of reliability can be expected, but other factors may also see slight improvements.

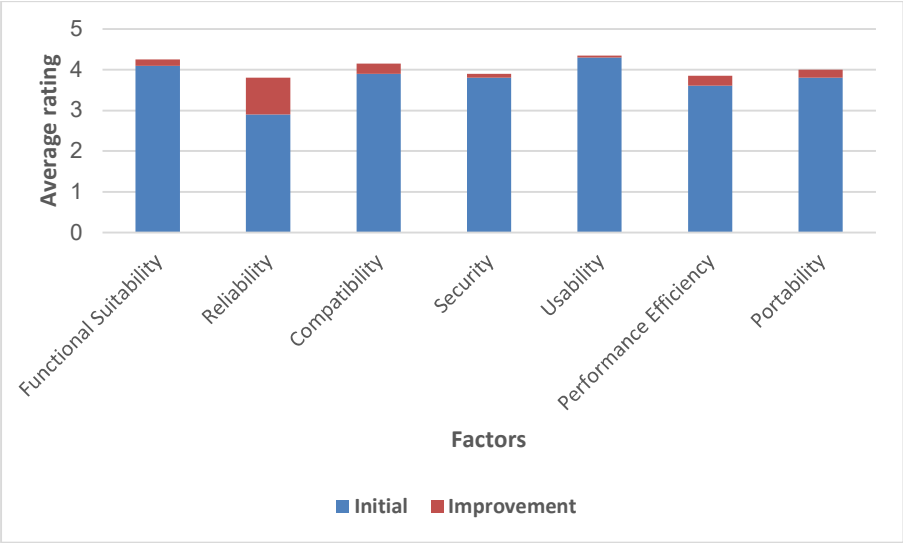


Figure 14: Example of expected results after improvements

Figure 14 shows the expected results after the improvements were made. Reliability was expected to show a massive increase, while the other factors only showed minor increases. A gradual increase in the number of users is expected over time as users started to use and understand the system.

It is worth noting that in this case, an improvement to a factor related to Perceived Usefulness (PU) was demonstrated, and as such a large improvement to factors related to Perceived Ease of Use (PEOU) is not expected. As discussed earlier, an improvement to factors related to PEOU may result in a substantial increase in factors related to the PU as well.

3.7 Validation of results

The next step in the methodology is to validate the effectiveness of the improvements made in the previous step. The following steps summarise the process of validation:

1. Comparison of overall usage
    a. Overall user count
    b. User acceptance rate
    c. Other usage statistics where available.
2. Re-evaluation of the system using the same questionnaire.

Overall usage comparison

The first method for validating the effectiveness of the improvements is to monitor the number of active system users. The objective of the methodology is to increase the user adoption of a software system by increasing the Perceived Usefulness (PU) and Perceived Ease of Use (PEOU). If the number of users has increased substantially over the initial amount, it is a good indication that the methodology was successful.

The overall acceptance rate could again be calculated by dividing the number of current users by the number of potential users. The acceptance rate can be compared to the initial acceptance rate of the system to validate improvement.

However, overall active usage time is often not a good measure of improvement, as improvements to the efficiency of the system may result in decreased time spent using the system. Thus, quantitative statistics need to be evaluated where available, such as the number of reports or calculations done on the system in a specific timeframe. The quantitative statistic being evaluated should be compared to the equivalent statistic that was recorded at the beginning of the process.

Re-evaluation of user acceptance through questionnaire

After implementing the improvements based on the first questionnaire's results, enough time needs to be given to allow new and returning users to adopt the updated system. This could range from a few weeks to a few months.

When enough time has passed, the questionnaire from the first survey is redistributed to active users. The second set of results needs to be processed in the same way as the first set of results to obtain statistics that can be directly compared to the first set of results.

The expected result for the second survey is an increase in the average score for each factor that was a part of the improvement process. However, it is possible that if a large number of new users are present, the overall average scores for some factors may not change drastically. New users may not have as much experience with the system as older users and may have a lower PEOU of the system. As the PEOU increases, it has the potential to increase the PU over time, as noted by Rajković et al [37].

The last step in the process is to analyse the results of the validation methods, discuss the outcome and comment on the effectiveness of the methodology.

3.8 Summary of methdology

The methodology consists of the following steps:

1. **Software system identification**
   Identify an underutilised software system and note the current acceptance rate.
2. **Selection of factors to analyse**
   Select factors to analyse based on prior research and group the factors according to their impact on Perceived Ease of Use (PEOU) and Perceived Usefulness (PU).
3. **Development of questionnaire**
   Use the factors selected in step two to create a questionnaire that will be used to gain insight into the current state of the system.
4. **Questionnaire distribution**
   Distribute the questionnaire amongst current users of the system and wait for responses within an acceptable window of time.
5. **Analyse the results of the questionnaire**
   Combine the overall ratings given to each software factor from each response to gain an overview of the current state of the system.
6. **Improvement of software based on research findings**
   Based on the findings in step 5, implement improvements in the key areas that received a low overall rating. Redistribute the software and notify current and potential users of the improvements.

7. **Re-evalute the software**

   Based on the feedback from the second round of questionnaires, evaluate the improvement and conclude on the effectiveness of the methodology.

8. **Validation of results**

   Two methods will be used to validate the effectiveness of the methodology:

   - **Re-evaluation of user acceptance:** After an acceptable period has passed, compare the latest acceptance rate to the acceptance rate at the start of the process (step 1). Distribute the same questionnaire used in step four again to the current users, and compare the results to the initial results.
   - **Overall usage comparison:** The second method of validation is to directly compare the number of users compared to the number of users before the improved version of the software has been distributed. This is the primary method of validation

This chapter summarised the methodology for improving user acceptance of software systems. The next chapter discusses the implementation of the methodology on a case study and evaluates the effectiveness thereof.

CHAPTER 4: RESULTS AND DISCUSSION

4.1 Preamble

The previous chapter discussed a method to improve user acceptance of software by improving factors related to the Perceived Ease of Use (PEOU) and Perceived Usefulness (PU). The method for obtaining and processing results was given, as well as the validation methods for the methodology.

This chapter discusses the results from implementing the method on a case study. An overview of the case study is given, followed by the process followed to collect the initial results as stated in the methodology. The results from the first set of user feedback are displayed and analysed, followed by the selection of factors to improve. A detailed summary of the improvements is given, followed by a second set of results that were collected four months after the improved system was released.

4.2 Case study background

A case study was used to validate the methodology developed in chapter 2. The software system used is a reporting and data analytics system developed in-house by the company that adopted the system to replace a makeshift older system. The new system will be referred to as New Reporting System (NRS)

NRS consisted of two components, namely a web user interface and report templates. The report templates are set up in Microsoft Excel spreadsheets. The web user interface could be used to set up automatic scheduling of tasks such as automated data processing and report generation. Reports are automatically sent to a predefined list of email addresses and are not sent if the data quality is insufficient when generating the report.

System overview

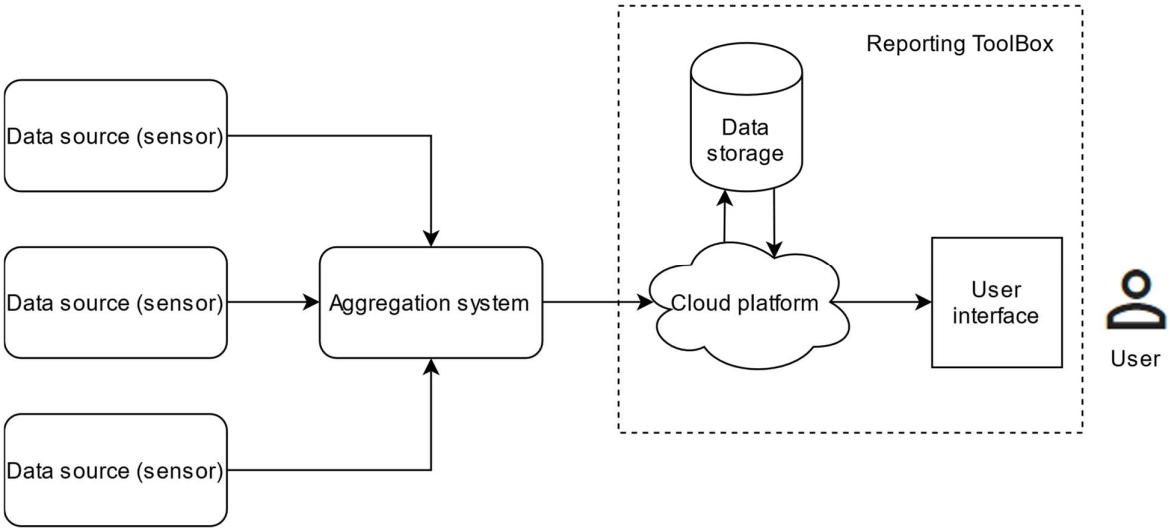


Figure 15: New Reporting System Overview

Figure 15 shows an overview of the basic flow of information in New Reporting System (NRS). NRS is hosted on a cloud-based server, which provides superior accessibility and performance over a local server [101]. The core of the system is based on the Python programming language for its functionality, flexibility and ease of use. The Pandas library of data analysis functionality is used to process the data and provide accurate results. The LaTeX typesetting system is used to produce high-quality reports with consistent layouts and formatting.

Various sources provide data to the system as illustrated in Figure 15. The user sets up a template for a report or data processing task, which is then processed by the cloud-based system. NRS can query data from multiple cloud databases, process the data accordingly and provide accurate results from large datasets. The results are returned in the form of a report or stored in the database for future use.

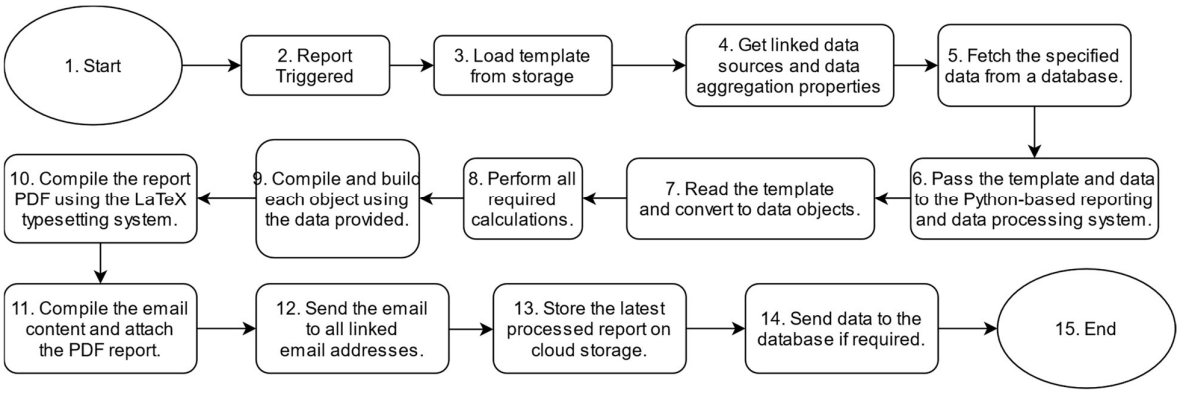


Figure 16: NRS - Report processing data flow

Figure 16 shows the process of generating a report using NRS. Steps 1-6 are executed by a web-based system, while steps 7-15 are performed by a Python-based reporting and data processing system. The web system fetches the template and data from a MySQL database hosted in a cloud provider as well as a cloud-based file system, then passes it to NRS's data processing system. The template is processed and the report is compiled. An email is generated, then sent to the required email addresses. Finally, the completed report is stored and data is sent back to the database if required.

Report template setup

Due to the advanced nature of the system, the setup procedure is quite complex.

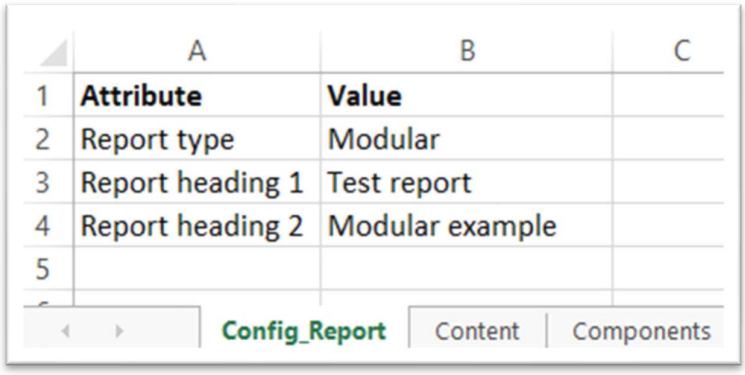
Report templates are set up in a Microsoft Excel file (.xlsm). The template is divided into nine main sections spread between tabs in the Excel file:

- **Configure report**
- **Content**
- **Components**
- **Export**
- **Figures**
- **Tables**
- **Graphs**
- **Calculations**

The basic concept for each section is the same. The settings and properties are spread between four columns:

- **ID:** The ID column denotes the ID of the item being set up. The attribute denotes the name of the item. This is used by the internal system to identify each item.
- **Attribute:** The attribute is a predefined value which is handled accordingly by the system when the report is processed. An example is the width of a graph or the name of a table.
- **Parameter:** The parameter is a certain subproperty of each attribute that could be set. In most cases this will default to “Value”, but it could also be attribute specific items such as the type of calculation or data series.
- **Value:** The value assigned to the specified parameter of each attribute.

Information about all of the possible attributes, parameters and values can be found in the NRS documentation.

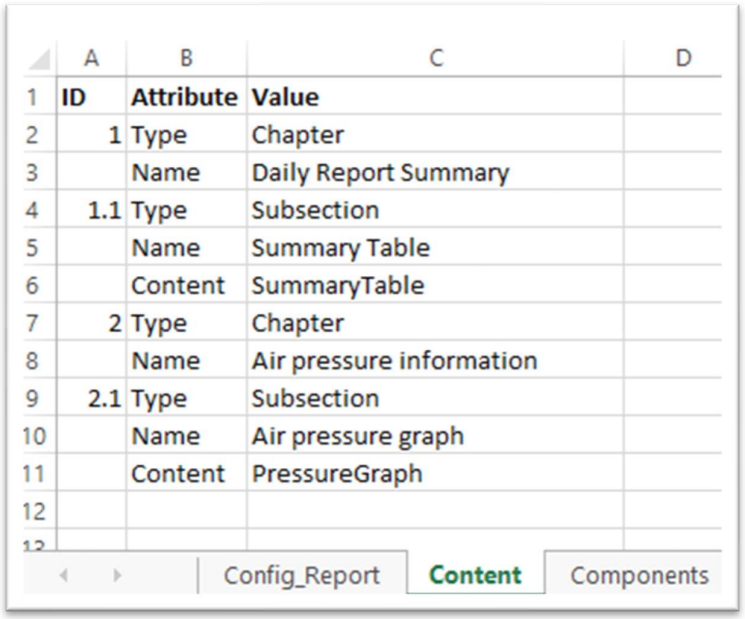


	A	B	C
1	**Attribute**	**Value**	
2	Report type	Modular	
3	Report heading 1	Test report	
4	Report heading 2	Modular example	
5			

Config_Report | Content | Components

Figure 17: NRS - Template setup (Configure report)

Figure 17 showcases the most basic section of a report setup. The ID and parameter fields are omitted, as the ID is not required in this section and the default parameter of “Value” is used in every case. In this example, the attribute “Report type” is assigned a value of “Modular”. The attribute “Report heading 1” and “Report heading 2” is assigned the values of “Test report” and “Modular example” respectively.



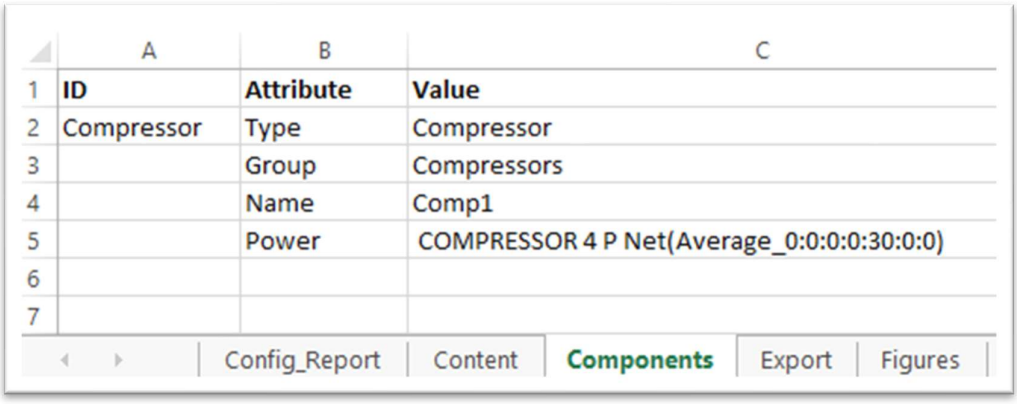
	A	B	C	D
1	**ID**	**Attribute**	**Value**	
2	1	Type	Chapter	
3		Name	Daily Report Summary	
4	1.1	Type	Subsection	
5		Name	Summary Table	
6		Content	SummaryTable	
7	2	Type	Chapter	
8		Name	Air pressure information	
9	2.1	Type	Subsection	
10		Name	Air pressure graph	
11		Content	PressureGraph	
12				

Config_Report | Content | Components

Figure 18: NRS - Template setup (Content)

Figure 18 illustrates the format of the content tab. The content tab of the report dictates the layout of chapters, subsections and content items displayed in the report, such as tables and graphs. The ID column is used to assign an ID property where required. The attribute column dictates the

type of attribute being configured, and the value column assigns a value of the relevant attribute. The parameter column is omitted as the default parameter of “Value” is used in every case.

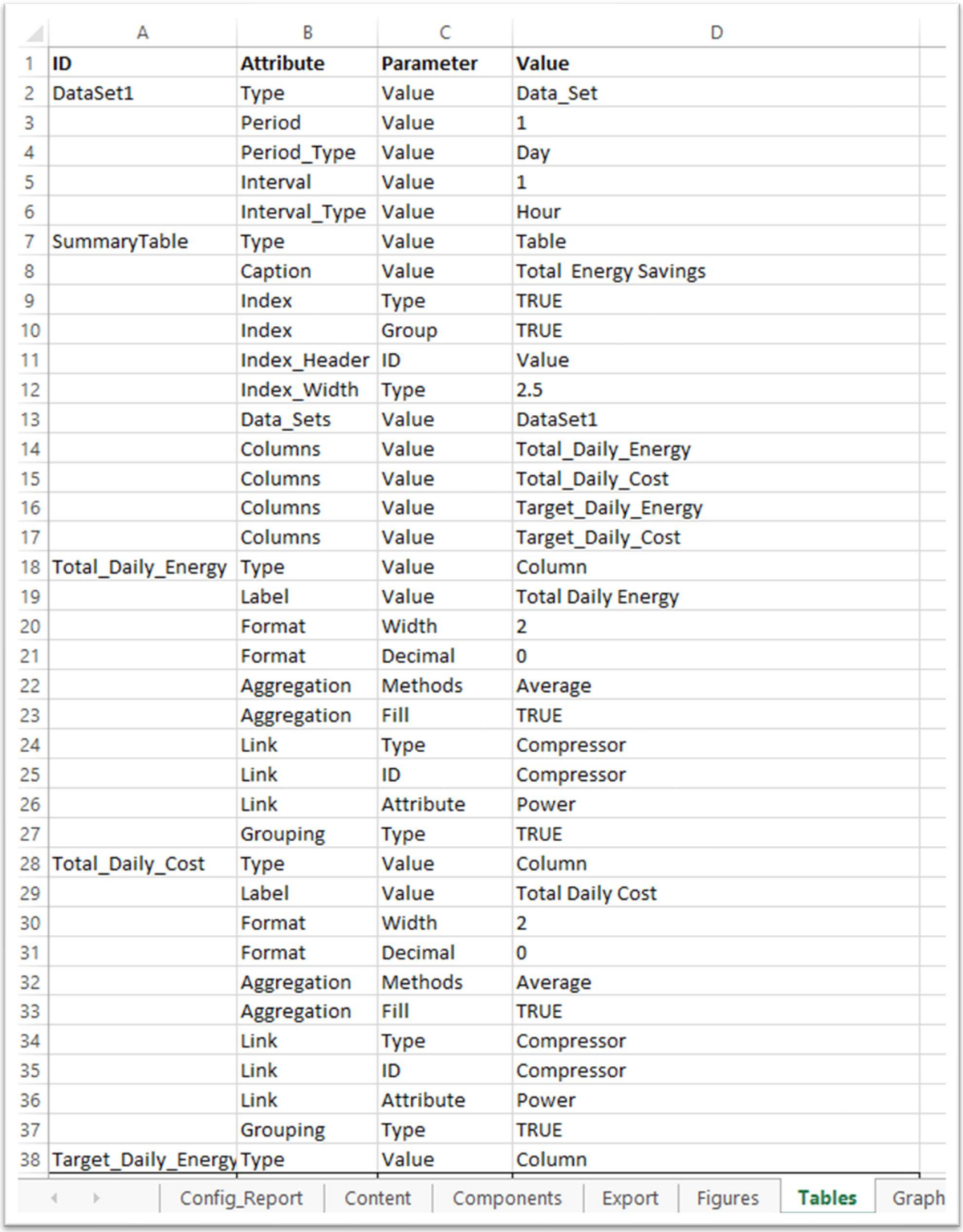


	A	B	C
1	**ID**	**Attribute**	**Value**
2	Compressor	Type	Compressor
3		Group	Compressors
4		Name	Comp1
5		Power	COMPRESSOR 4 P Net(Average_0:0:0:0:30:0:0)
6			
7			

Config_Report | Content | **Components** | Export | Figures

Figure 19: NRS - Template setup (Components)

The components are set up in a similar manner to content items. The ID is assigned, a few attributes are created and each attribute is assigned a value. The “Power” attribute is assigned to the name of a data source along with the required data aggregation format.



	A	B	C	D
1	**ID**	**Attribute**	**Parameter**	**Value**
2	DataSet1	Type	Value	Data_Set
3		Period	Value	1
4		Period_Type	Value	Day
5		Interval	Value	1
6		Interval_Type	Value	Hour
7	SummaryTable	Type	Value	Table
8		Caption	Value	Total Energy Savings
9		Index	Type	TRUE
10		Index	Group	TRUE
11		Index_Header	ID	Value
12		Index_Width	Type	2.5
13		Data_Sets	Value	DataSet1
14		Columns	Value	Total_Daily_Energy
15		Columns	Value	Total_Daily_Cost
16		Columns	Value	Target_Daily_Energy
17		Columns	Value	Target_Daily_Cost
18	Total_Daily_Energy	Type	Value	Column
19		Label	Value	Total Daily Energy
20		Format	Width	2
21		Format	Decimal	0
22		Aggregation	Methods	Average
23		Aggregation	Fill	TRUE
24		Link	Type	Compressor
25		Link	ID	Compressor
26		Link	Attribute	Power
27		Grouping	Type	TRUE
28	Total_Daily_Cost	Type	Value	Column
29		Label	Value	Total Daily Cost
30		Format	Width	2
31		Format	Decimal	0
32		Aggregation	Methods	Average
33		Aggregation	Fill	TRUE
34		Link	Type	Compressor
35		Link	ID	Compressor
36		Link	Attribute	Power
37		Grouping	Type	TRUE
38	Target_Daily_Energy	Type	Value	Column

Config_Report | Content | Components | Export | Figures | **Tables** | Graph

Figure 20: NRS - Template setup (Objects)

Figure 20 shows the format of setting up objects. These objects are tables, graphs, figures, calculations and export object. Similar to components, the ID is assigned, a few attributes are created for each object and the parameter used is also specified. A value is then specified to the parameter. An example of different parameters being assigned values under the same attribute is the “Link” attribute under the “Total_Daily_Energy” object.

The files are uploaded using the web user interface, which in turn stores it using a cloud storage solution.

User interface

The NRS configuration user interface is located on a web system that serves a variety of functions for NRS and other systems, and is accessed through any browser. The web-based report management user interface is used to upload the templates, link recipients and trigger the reports. Various report parameters such as trigger frequency and the report email details can also be set.

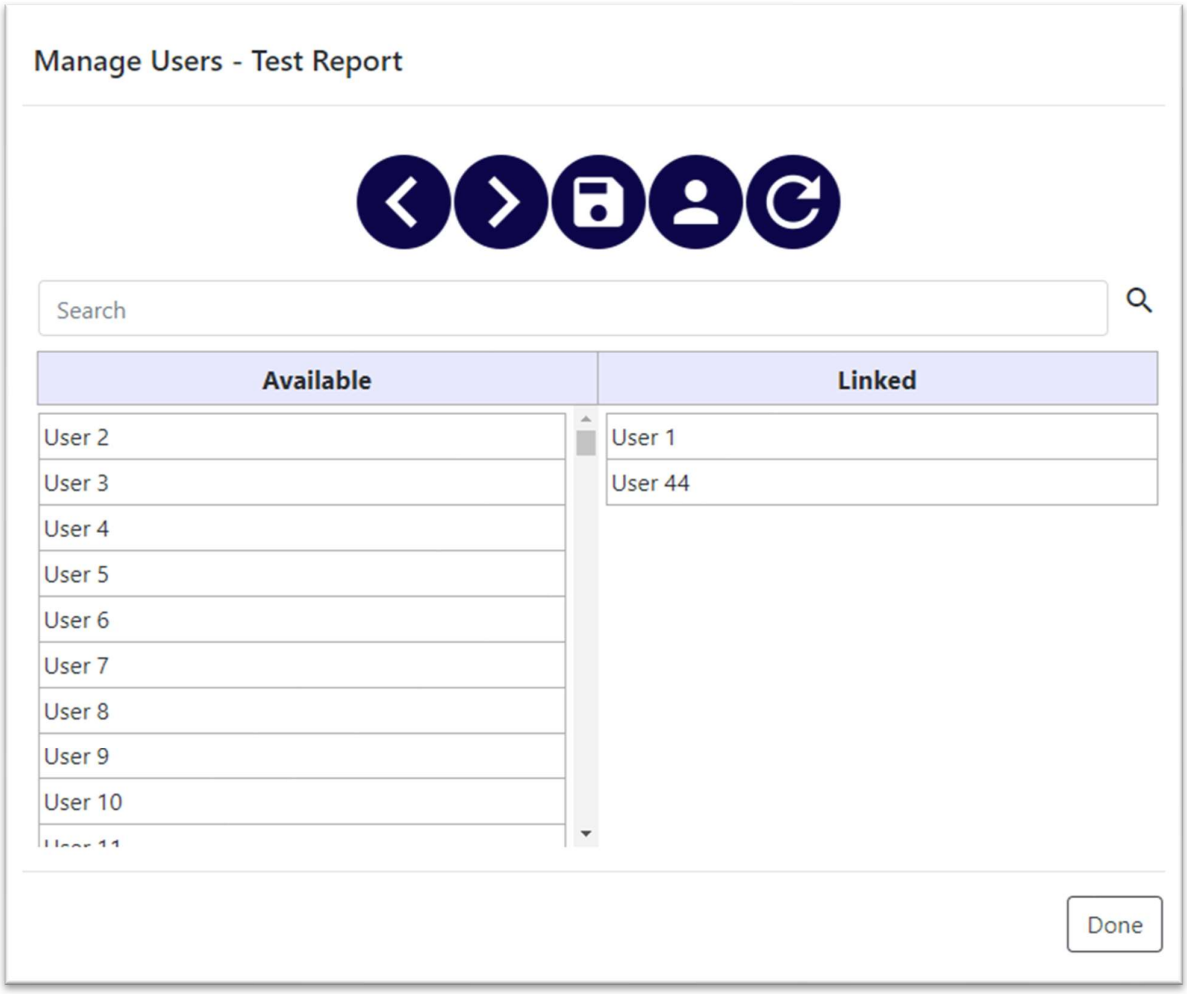


Figure 21: NRS - Linked Users

Figure 21 shows the “linked users” user interface. Users that need to receive the report can be be linked using this interface. There is only a basic form of access control by limiting the user’s access to the part of the system that manages reports. Users with access to this part of the system can access and edit any report belonging to the client or organisation they are currently working on.

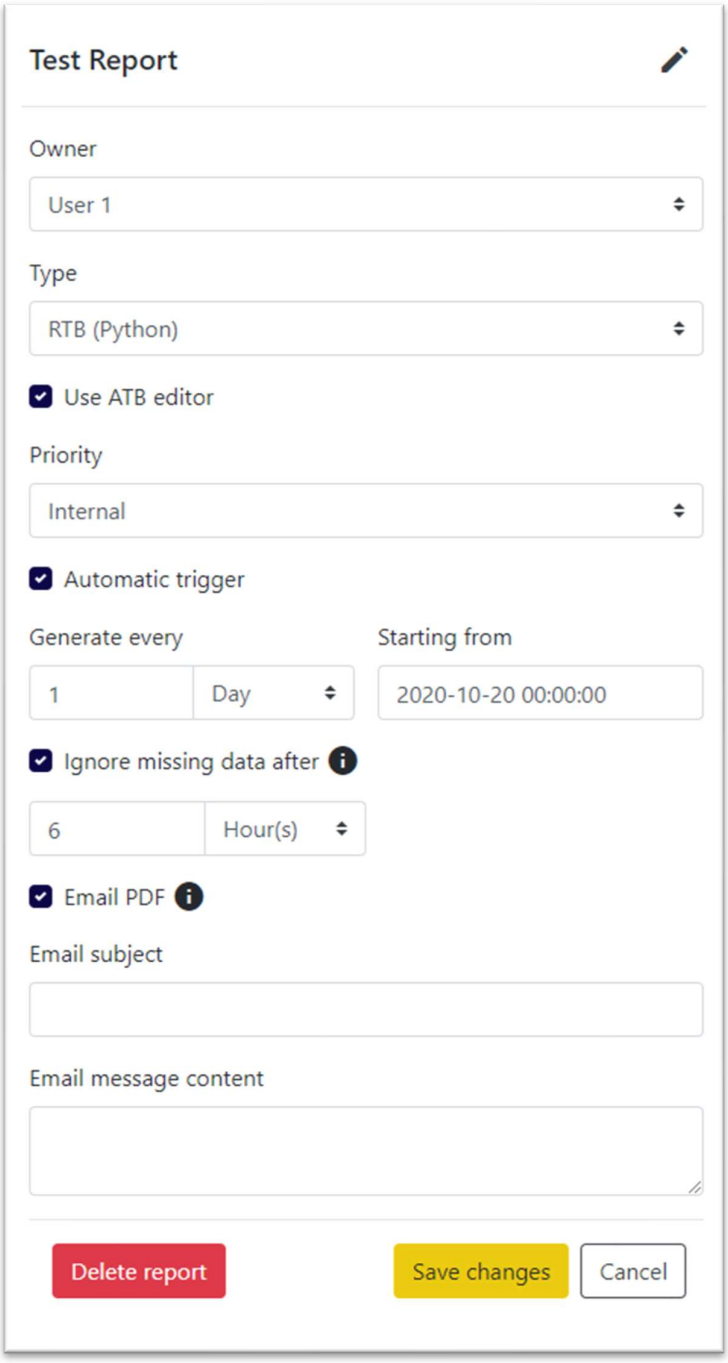


Figure 22: NRS - Manage report

Figure 22 shows the manage report user interface. The report owner can be set, as well as the type of report. Automatic triggering of the report can be set up to generate and send the report to the linked users at the specified times and dates. The email content and subject header can be manually set.

The templates for setting up data processing tasks and reports needed to be set up using Excel and uploaded to a web user interface.

User adoption problem

Despite the advanced features and capabilities, the user adoption of NRS was only 8.33%, with only 5 out of 60 potential users actively using the system during January 2019. This was determined through asking potential users directly if they were actively using the system as no system existed to track user activity. Various training and introduction presentations were given by the company to all active and potential users with no significant increase to user adoption.

An already in-place legacy system was preferred by most of the users, despite the older system being less capable and less efficient in using system resources. The older system was referred to as Excel Reports and left the implementation of formulas used in data analysis up to the user, whereas NRS includes standard formulas for calculating a wide array of metrics and statistics.

The layout and formatting of reports generated by Excel Reports were not standardised and could be changed by any user, resulting in different or inconsistent layouts between reports or even individual pages within a report. The process was also very inefficient as large calculations needed to be performed every time a report was generated, instead of storing already processed data in the database. These reports are sent to clients, and the inconsistent formatting and results needed constant review.

NRS is a powerful tool that can handle large amounts of calculations and generate large reports for each template. Because the templates are standardised, the reports will always be formatted in the same way.

NRS provides a prime example of a system that is good in theory but has a low user adoption rate, which is why it was perfectly suited as a case study to test the developed methodology developed in this study. By using older systems instead of NRS, the users were actively decreasing their efficiency, and by extension the efficiency of the work completed by the company.

4.3 Initial results

Initial usage numbers

The initial user count was only five out of a potential 60 users during January 2019. This statistic was determined by asking potential users if they actively use the system. This slowly increased to 18 over the next 14 months up to March 2020 as users started to adopt the system. It was evident that adoption was much slower than anticipated. *Figure 23* show the user count at the start of the case study.

Many of the potential users preferred to calculate statistics by hand or by using spreadsheets. Others used the older Excel reports system, which was far less efficient and often led to inconsistent formatting or incorrectly calculated results.

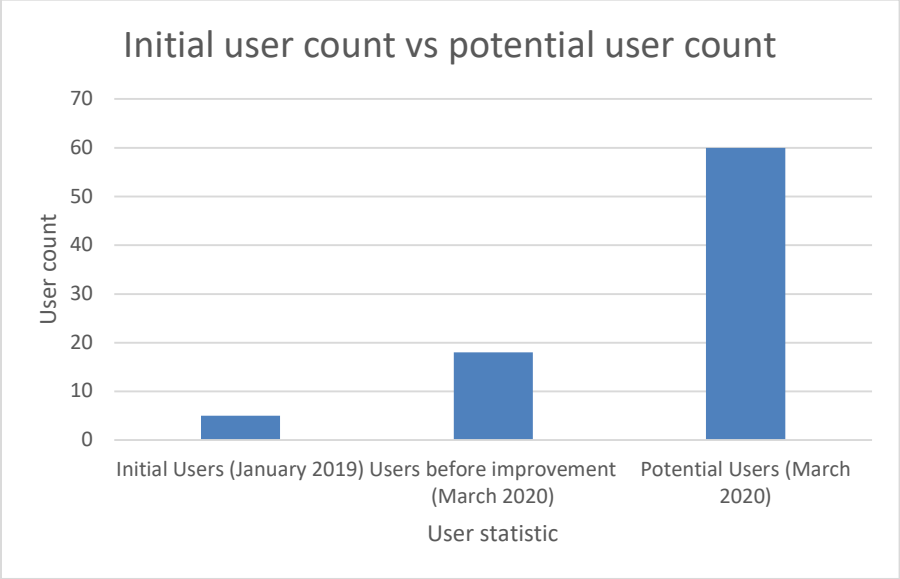


Figure 23: Initial vs potential user count (March 2020)

Distribution of questionnaire

The questionnaire developed in section 3.4 was distributed to all of the active NRS users. The window for responses was two weeks, after which the data was analysed. Out of the 18 questionnaires sent, 15 responses were received. The response rate for the questionnaire is 83.3% and covers most of the current active users.

Example of calculation for each factor

As stated in section 3.5, the results for each subfactor for each questionnaire respondent will be combined and an average score for each factor will be obtained. The validity of the result will be analysed using Cronbach's alpha.

An example of the calculations performed to obtain the combined score and Conbach's alpha value for the questions related to functional suitability is as follows:

Timestamp	The system provides all the required functionality	The system produces the correct results, and the results are accurate	The system provides all of the appropriate functionality to complete the tasks I use it for	Total
2020/04/30 8:54:09 am EET	5	5	5	**15**
2020/04/30 8:59:47 am EET	3	4	4	**11**
2020/04/30 9:27:10 am EET	4	4	4	**12**
2020/04/30 9:27:56 am EET	4	4	4	**12**
2020/04/30 9:33:06 am EET	3	3	2	**8**
2020/04/30 9:51:37 am EET	5	5	5	**15**
2020/04/30 7:04:03 pm EET	4	5	4	**13**
2020/05/01 12:01:25 am EET	4	3	5	**12**
2020/05/05 11:51:35 am EET	4	5	3	**12**
2020/05/06 9:26:46 am EET	2	4	1	**7**
2020/05/07 8:15:43 am EET	2	5	2	**9**
2020/05/08 7:53:15 am EET	3	4	4	**11**
2020/05/08 9:14:35 am EET	4	5	4	**13**
2020/05/08 10:41:17 am EET	3	4	2	**9**
2020/05/21 9:05:51 am EET	4	4	4	**12**
Total	**54**	**64**	**53**	**171**

The variance for each question is calculated using equation 4 in section 3.5. The mean value for each question is calculated by dividing the sum of all the scores per question by the amount of items in the sample.

The following table shows the calculated statistics for each question:

Statistic	Question 1	Question 2	Question 3	Total
Variance	0.7733	0.4622	1.4489	2.6844
Mean	3.6	4.27	3.53	

The formula for Cronbach's alpha is given as follows [84]:

$$\alpha = \frac{k}{k-1}\left(1 - \frac{\sum_{i=1}^{k}\sigma_{y_i}^2}{\sigma_x^2}\right) \tag{8}$$

where:

- k represents the number of items,
- $\sum_{i=1}^{k}\sigma_{y_i}^2$ represents the sum of the variance of each of the item results, and
- σ_x^2 represents the variance of the total scores of each item in the test added together.

The number of questions for this factor is $\boldsymbol{k = 3}$.

The sum of the variances can be calculated as follows:

$$\sum_{i=1}^{k}\sigma_{y_i}^2 = \mathbf{2.6844} \tag{9}$$

The variance of the sum of the scores from each respondent is $\boldsymbol{\sigma_x^2 = 5.04}$.

The final result of the Cronbach's alpha calculation is as follows:

$$\boldsymbol{\alpha = \frac{k}{k-1}\left(1 - \frac{\sum_{i=1}^{k}\sigma_{y_i}^2}{\sigma_x^2}\right) = \frac{3}{3-1}\left(1 - \frac{2.6844}{5.04}\right) = 0.7011} \tag{10}$$

The mean of the average ratings for all subfactors is used to calculate the combined score for each factor. For example, the combined score for functional suitability can be calculated as follows:

$$\boldsymbol{Combined\ score} = \frac{\sum_{i=1}^{n} \boldsymbol{x_i}}{\boldsymbol{k}} = \frac{\boldsymbol{3.60 + 4.27 + 3.53}}{\boldsymbol{3}} = \boldsymbol{3.8} \quad (11)$$

The results from equations 10 and 11 can be seen in *Table 14*.

The rest of the results were calculated in the same way.

Initial questionnaire results

The results were compiled and analysed after the responses were closed. The Cronbach's alpha value for each question's responses was calculated [84], as well as the average combined score for each factor. Where respondents failed to submit a valid answer for a question, it is mentioned, and the overall rate of empty responses is given for each question. Where no non-response information is given for a question, it can be assumed that the non-response rate is 0%.

Table 14: Initial results - Functional Suitability

Subfactors	Average rating	Combined score	Cronbach's alpha
Functional completeness	3.6 / 5	3.80 / 5	0.701
Functional correctness	4.27 / 5		
Functional appropriateness	3.53 / 5		

Table 14 shows a summary of the initial results for functional suitability. All of the factors received good ratings which indicated that the system provided all of the functionality that is expected. The alpha value was 0.701 which is above the recommended level, indicating good consistency between the subfactors. All respondents submitted answers to all questions.

Table 15: Initial results - Reliability

Subfactors	Average rating	Combined score	Cronbach's alpha
Maturity	3.43 / 5	3.52 / 5	0.856
Availability	3.93 / 5		
Fault tolerance	2.80 / 5		
Recoverability	3.93 / 5		

Table 15 shows a summary of the initial results for reliability. Except for fault tolerance, the subfactors related to reliability received good ratings overall. Fault tolerance is the clear weak point in the system regarding reliability and could be improved. The alpha value was 0.856 which is well above the recommended level, indicating good consistency between the subfactors. One respondent did not submit an answer to the question related to maturity, leading to a 7.14% non-response rate for a single question.

Table 16: Initial results - Compatibility

Subfactors	Average rating	Combined score	Cronbach's alpha
Co-existence	3.27 / 5	3.47 / 5	0.832
Interoperability	3.67 / 5		

Table 16 shows a summary of the initial results for compatibility. Both subfactors received good scores, and no improvement is needed. The alpha value was 0.832 which is well above the recommended level, again indicating good consistency between the subfactors. All of the questions were answered by each respondent.

Table 17: Initial results - Security

Subfactors	Average rating	Combined score	Cronbach's alpha
Confidentiality	3.80 / 5	3.24 / 5	0.886
Integrity	3.27 / 5		
Non-repudiation	2.73 / 5		
Accountability	2.60 / 5		
Authenticity	3.85 / 5		

Table 17 shows a summary of the initial results for security. The average ratings for the subfactors related to security are mixed, with confidentiality, integrity and authenticity receiving high scores. However, non-repudiation and accountability received low scores, indicating that improvements are needed. The alpha value was 0.886, which indicates good consistency between the responses. A single respondent failed to submit an answer to the question related to authenticity, thus leading to a 6.67% non-response rate.

Table 18: Initial results - Usability

Subfactors	Average rating	Combined score	Cronbach's alpha
Appropriateness recognisability	2.26 / 5	2.66 / 5	0.877
Learnability	2.60 / 5		
Operability	2.93 / 5		
User error protection	2.53 / 5		
User interface aesthetics	2.67 / 5		
Accessibility	2.93 / 5		

Table 18 shows a summary of the initial results for usability. The average score for subfactors related to usability is lower than average, with no subfactor receiving an average score above

three out of five. Appropriateness recognisability, learnability, user error protection and user interface aesthetics all received scores at or below 2.70 out of 5. The alpha value is 0.877. Thus, the values can be accepted. All of the questions were answered by all of the respondents.

Table 19: Initial results - Performance Efficiency

Subfactors	Average rating	Combined score	Cronbach's alpha
Time behaviour	3.0 / 5	3.36 / 5	0.807
Resource utilisation	3.61 / 5		
Capacity	3.17 / 5		

Table 19 shows a summary of the initial results for performance efficiency. The alpha value was 0.807, indicating good consistency between questions. The non-response rate for the questions was 13.33% for time behaviour and resource utilisation, followed by 20% for capacity.

Time behaviour and capacity both received a decent score, but there is room for improvement. Resource utilisation received a good score, and no improvement is needed

Table 20: Initial results - Portability

Subfactors	Average rating	Combined score	Cronbach's alpha
Adaptability	2.46 / 5	3.11 / 5	0.743
Installability	3.40 / 5		
Replaceability	3.47 / 5		

Table 20 shows a summary of the initial results for portability. The alpha value was 0.637 which indicates a slightly less than ideal consistency between respondents. Two respondents failed to submit answers for the question related to adaptability, leaving the non-response rate at 13.33%

Adaptability received a very poor score of only 2.46 out of 5, indicating the need for possible improvement. Installability and replaceability both received respectability scores above 3.40 and should not be a cause for concern.

Initial result summary

The overall minimum alpha value was 0.637, and the maximum was 0.913. All of the factors' results show good consistency, which indicates that the questions regarding each subfactor were structured correctly and that each question group referred to the same factor.

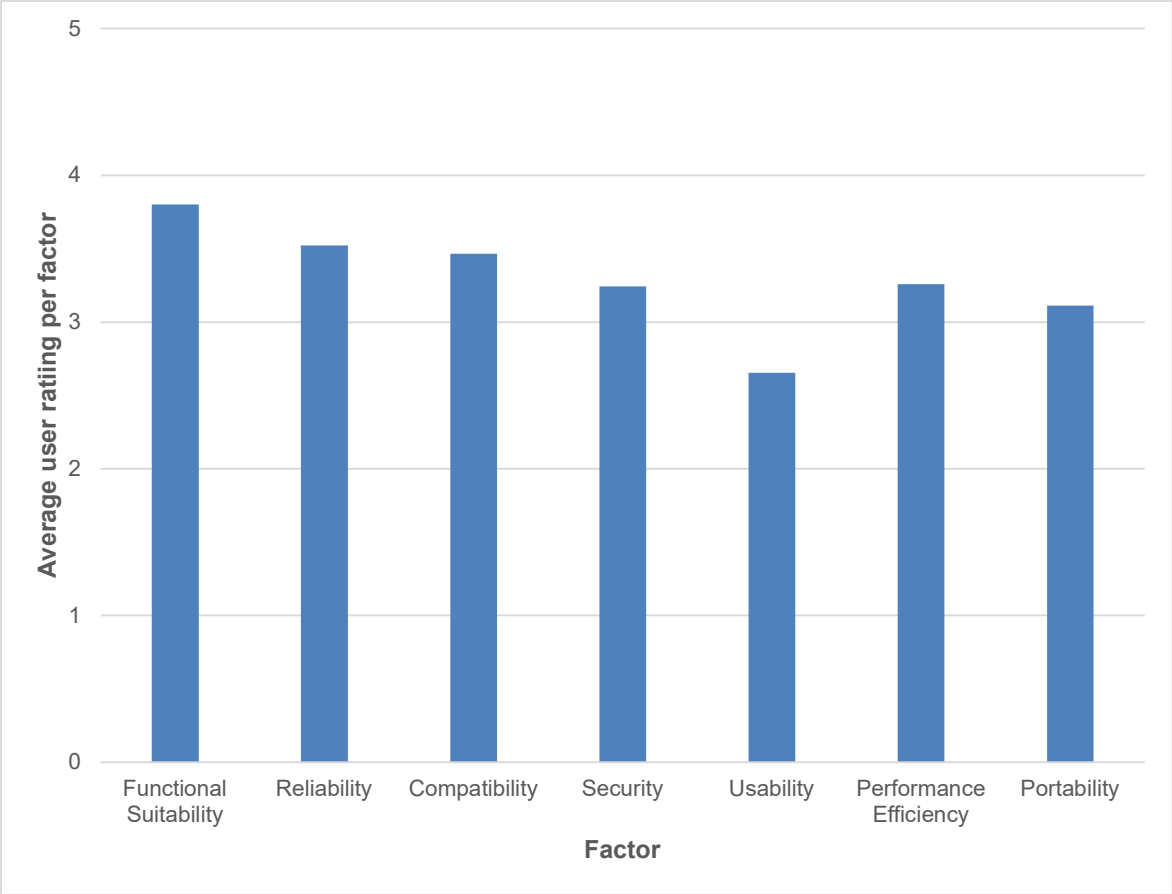


Figure 24: Average initial user rating per factor (March 2020)

Figure 24 shows the average initial user rating per factor. The highest rating was received by functional suitability at 3.8 out of 5, followed by reliability and compatibility at 3.52 and 3.47 respectively.

Usability received the lowest average ratings at only 2.65 out of 5. From the data shown in Figure 24, it was clear that usability was the main problem affecting user acceptance.

Usability shortcomings

When looking at the average results for the usability subfactors in Table 18, Appropriateness recognisability scored the lowest at 2.26, followed by user error protection at 2.53. Next is accessibility at 2.93, learnability at 2.60 and user interface aesthetics at 2.67. Operability was the usability subfactor with the highest score at 2.93.

Appropriateness recognisability received the lowest score out of all 26 subfactors from all categories, while the subfactor in the usability group at 2.93 still scored lower than 15 out of 20 subfactors from other groups.

Judging by the poor usability ratings, it was clear that drastic improvements were needed regarding the way that a user interacts with the system. Users found the system hard to learn, and it was too easy to make mistakes while using the system. Because the templates were set up in Microsoft Excel, the primary user interface was a spreadsheet, which is not user-friendly or pleasant to look at.

Another issue is that none of the available options and settings used to set up a report were pre-filled and had to be looked up in the documentation. Even if the correct setting name was used in the spreadsheet, the input type had to be correct, and the formatting of the spreadsheet needed to be 100% correct for the report to work correctly.

The system also had a very high learning curve, and as a result, new users were discouraged from using the system. Users started using the system with a specific goal in mind but struggled to perform basic functions and thus never realised the full potential of the system.

The next step of the process is to implement improvements to address the shortcomings of the system.

4.4 Technical work and improvements

Improvements were made to address the shortcomings identified by the initial questionnaire results. The initial results showed a clear need for improvement in usability, specifically regarding

appropriateness recognisability, user error protection, learnability and user interface aesthetics. As a result, overall usability was the main focus of the improvements.

Usability improvement

To improve usability, a new user interface was developed while focusing on the aspects discussed in section 3.6 [97]:

- **Use natural and simple wording:** All new views use clear and concise language. No unnecessary information is displayed.
- **Use familiar dialogue:** Familiar terms are used to convey information about the functionality on every page.
- **Minimise memory load:** The layout of the new user interface was designed to be as simple as possible while still providing all of the functionality expected. Every page only contains the necessary information.
- **Consistency:** The system is designed to look the same on each page, with a familiar button and grid layout.
- **Feedback:** Each action taken by the user is met with a status message from the system that indicates the success/failure of the action.
- **Clearly marked exits:** Each page provides a clear and easily accessible method to return to the previous view.
- **Shortcuts:** For advanced users, keyboard shortcuts are implemented to reduce the time between actions performed. Functionality to clone objects could also dramatically reduce the amount of time spent to build reports.
- **Good error messages:** Error messages are clear and concise, indicating the cause of the issue.
- **Prevent errors:** The system is built with error prevention in mind. A user will only be able to enter information in the relevant format that is expected by the system. If an option needs to be selected, only the available options will be displayed to the user. The input box will also limit the input value to the relevant format where required, such as numbers or Boolean values.
- **Help and documentation:** Full documentation, as well as a basic guide, is easily accessible to the user. It will be available only a few clicks away from the main part of the system.

User interface design: main view

The existing system for handling the report templates was programmed in Microsoft .NET C#, while the report templates are stored in a document-oriented database schema (MongoDB). Thus, it was decided to design a new user interface in Microsoft Asp.NET MVC, with Bootstrap 4.0 styling applied.

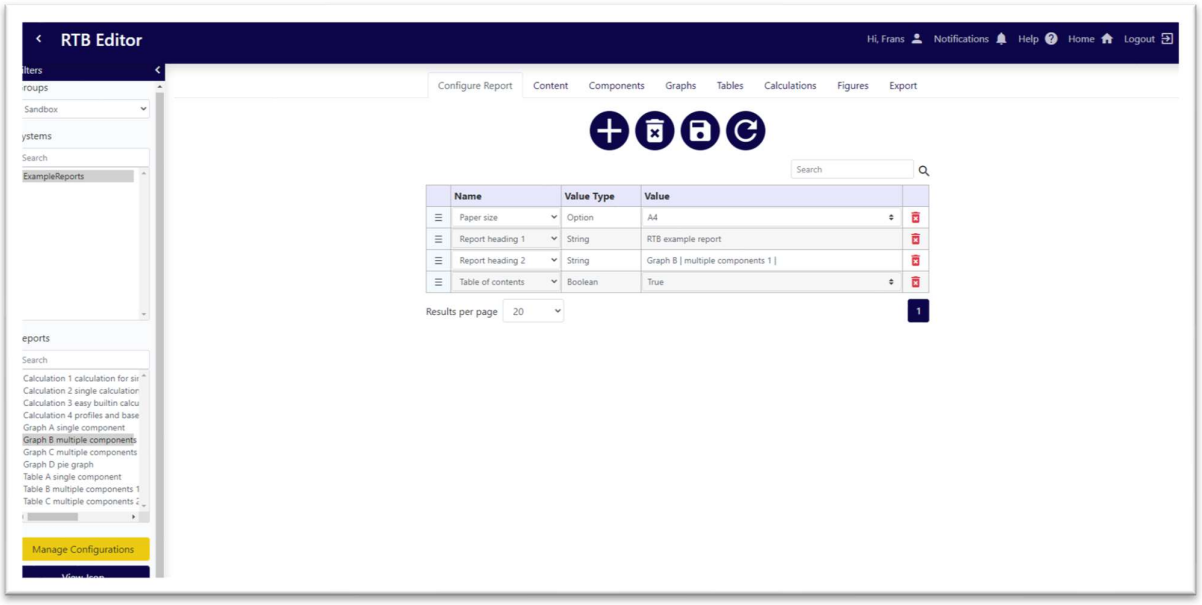


Figure 25: Improved user interface - Main view

Figure 25 shows the main view of the new user interface. The new user interface consists of a single page with several tabs, one tab for each section of the report. The client group, system and active report can easily be selected from an omnipresent sidebar on the left side of the screen. The main content of the user interface is placed in the centre of the page to be the main focus of the user. Buttons with clear icons are used to indicate actions that the user can perform.

The user interface layout is similar to other systems that people who work at the company utilising NRS use every day, and as a result, the user interface should feel very familiar to users. This helps reduce memory load as users are all familiar with the basic actions denoted by each type of icon.

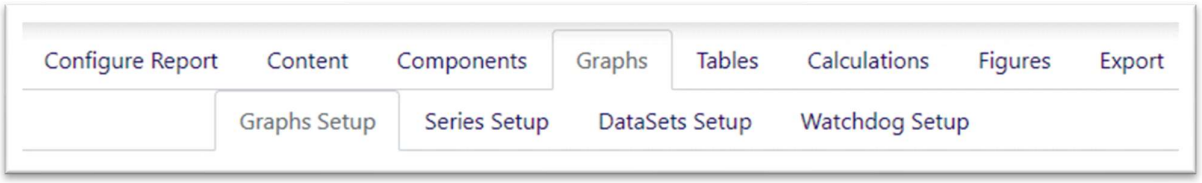


Figure 26: Improved user interface - Tab structure

Figure 26 shows the user interface tab structure. The user interface was designed to be consistent between sections and follows a common layout structure. Main report objects like the configuration page, content list page, graphs and report components are organised into tabs within the page. Different items of the same type are organised in subtabs under each main tab where applicable. The tab structure is always the same, adding to the constancy of the user interface.

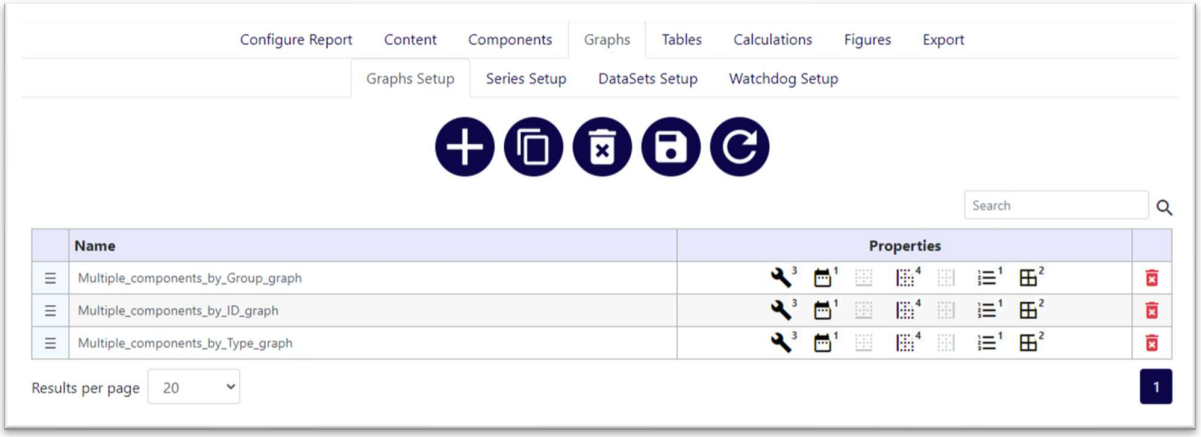


Figure 27: Improved user interface - Main object tabs

Figure 27 shows the main object tab layout. Each main tab contains a single grid with several columns. The grid columns are generally organised in the same way. The leftmost column contains an icon through which the row can be selected, for instance when multiple items need to be deleted in the same operation. The next column contains the name of the object or attribute. Main objects like Graphs, Components or Tables have names that are set by the user and can be changed at will. The next column contains buttons for the properties and subobjects that may be created and edited for each object. The buttons for properties and subobjects each contain a number badge in the upper right corner indicating the number of properties added in the respective subobject.

The last column provides a delete button which brings up a confirmation dialog for the user to confirm whether the item should be deleted.

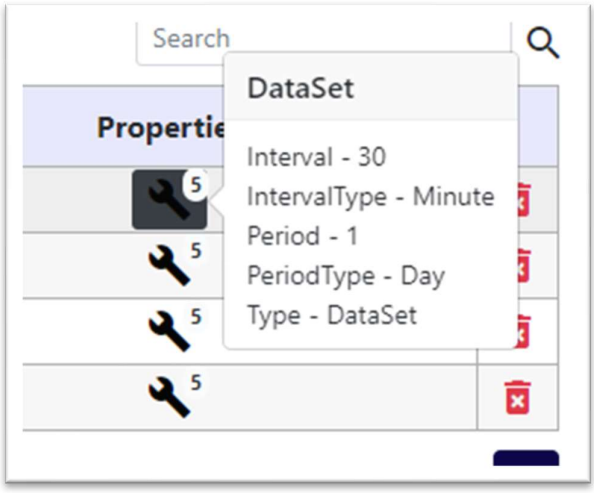


Figure 28: Improved user interface - Property tooltips

Figure 28 shows the property tooltip popups. Hovering over a properties button brings up a tooltip which allows the user to quickly see which properties have been added, and what the values are. The tooltips are pre-loaded to improve performance.

Users interface design: properties dialog

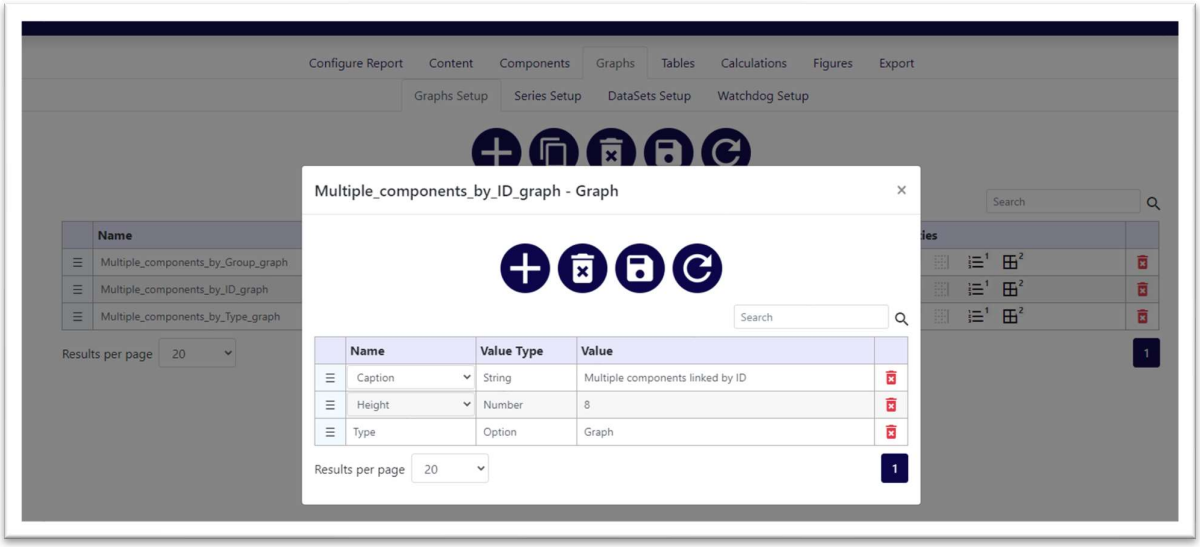


Figure 29: Improved user interface - Viewing properties

Figure 29 shows the properties dialog. Clicking any properties button brings up a dialog from which all the properties, value types and values can be seen and edited. Each entry is set by first selecting the name of the property from a pre-existing list on the left. The value type column will then change to indicate the expected value. If a default value exists for the selected property, it will automatically appear in the value field on the right. If only a certain set of possible values are available, the values are presented to the user in a select list. Otherwise, the user is expected to input the value manually.

The manual input field changes type according to the expected value type. If a string value is expected, any valid string value will be accepted. However, if a number value is expected, the field changes to only accept number values, thereby reducing the number of errors experienced by the user.

Lastly, all dialogs contain an exit button to return to the previous screen. Alternatively, the user can close the dialog by clicking anywhere outside of the modal in the darkened area.

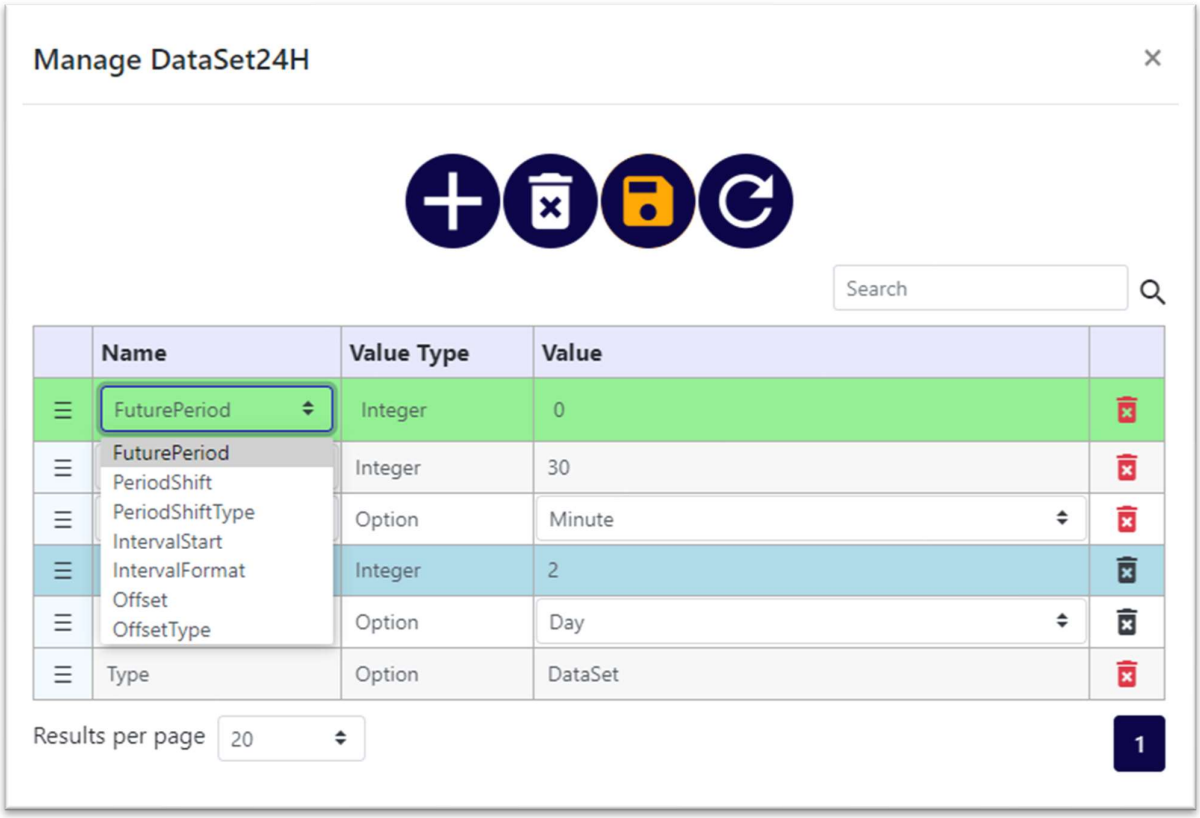


Figure 30: Improved user interface - Adding properties

Figure 30 shows an example of properties being added and edited. New properties are added by clicking the large plus button at the top of the grid. A new row entry appears, and the user can set the property they want to add, followed by the property value. If multiple properties can be set, the user can select the property they want to add using the dropdown menu that appears when clicking a property name. Any changes need to be committed by clicking the save button. New rows are indicated by a green tint while existing rows that have been changed are indicated by a blue tint.

User interface design: components page

Components contain a structure of the data sources linked to the report, and consist of three levels. The top-level is the component itself, which may contain several properties, groups and attributes. Attributes are the second level, and each attribute may contain several parameters. The parameters are the source of all data that is used in the report and can be number values, linked data tags or plain strings. Parameters are a key part of interacting with data sources.

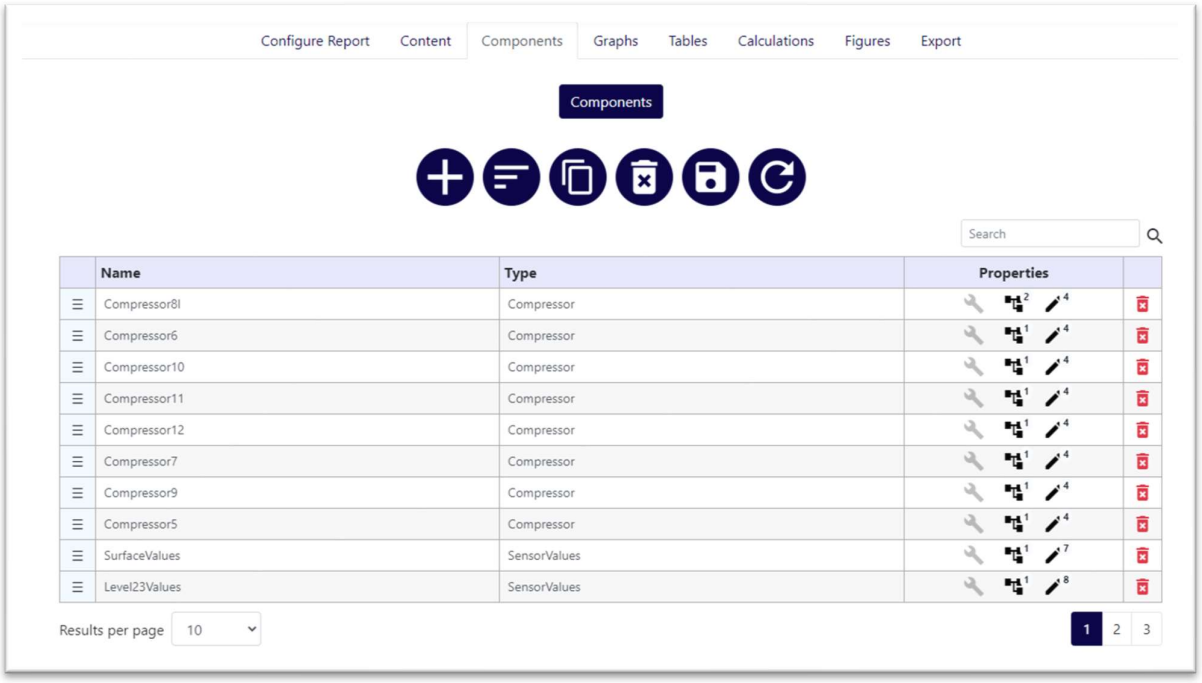


Figure 31: Improved user interface - Components tab

Figure 31 shows the main component setup tab. At first glance, the components tab looks similar to other main object tabs. However, when clicking the attributes button indicated by a pencil, the grid moves towards the left to make room for the attributes grid for the selected component. The component groups and properties are accessible through buttons for each component and are edited through the standard properties dialog.

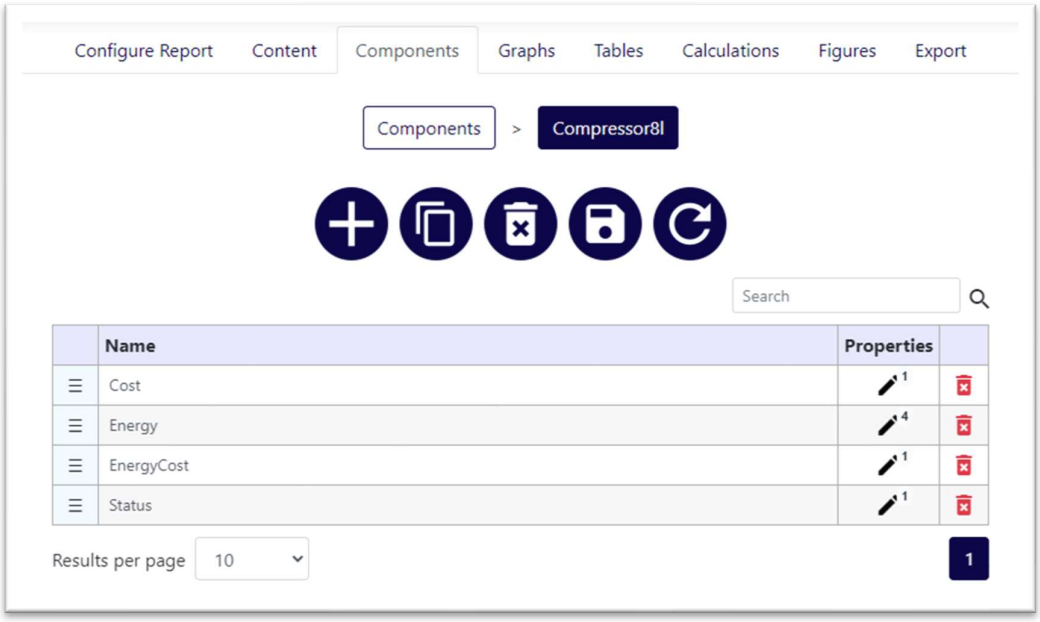


Figure 32: Improved user interface - Component attributes grid

Figure 32 shows the component attributes grid. The attributes grid presents the current attributes for each component in a similar way to other object grids. Attributes are renamable by the user, and attribute parameters can be viewed by clicking the parameters button on the right. Clicking the parameters button shifts the attributes grid to the left to make room for the parameters grid.

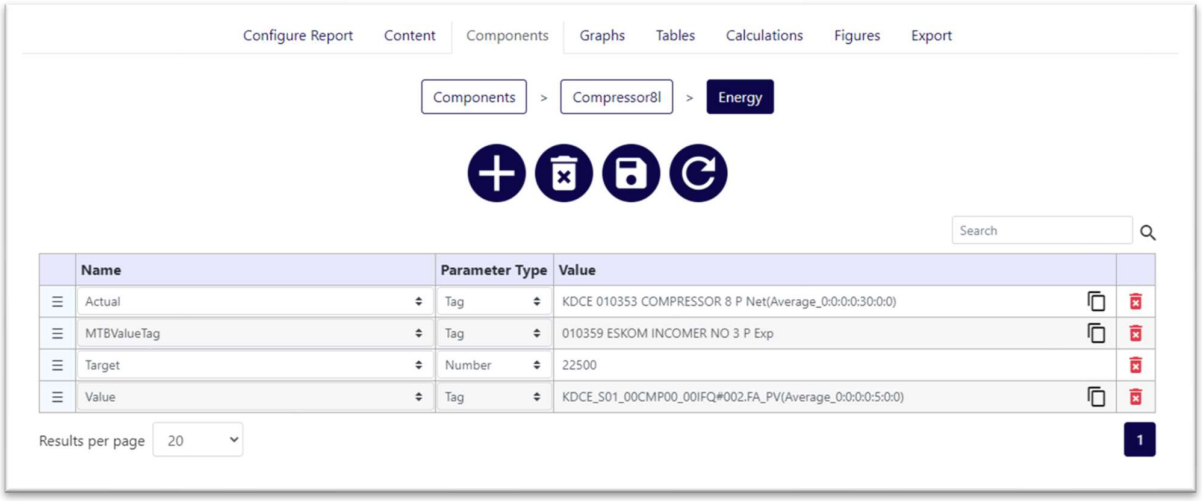


Figure 33: Improved user interface - Component attribute parameters grid

Figure 33 shows the attribute parameters grid. The parameters grid is similar to other grids but can accept tags as values which are the primary data source for reports. Tags link to data in a cloud database.

User interface design: constant feedback

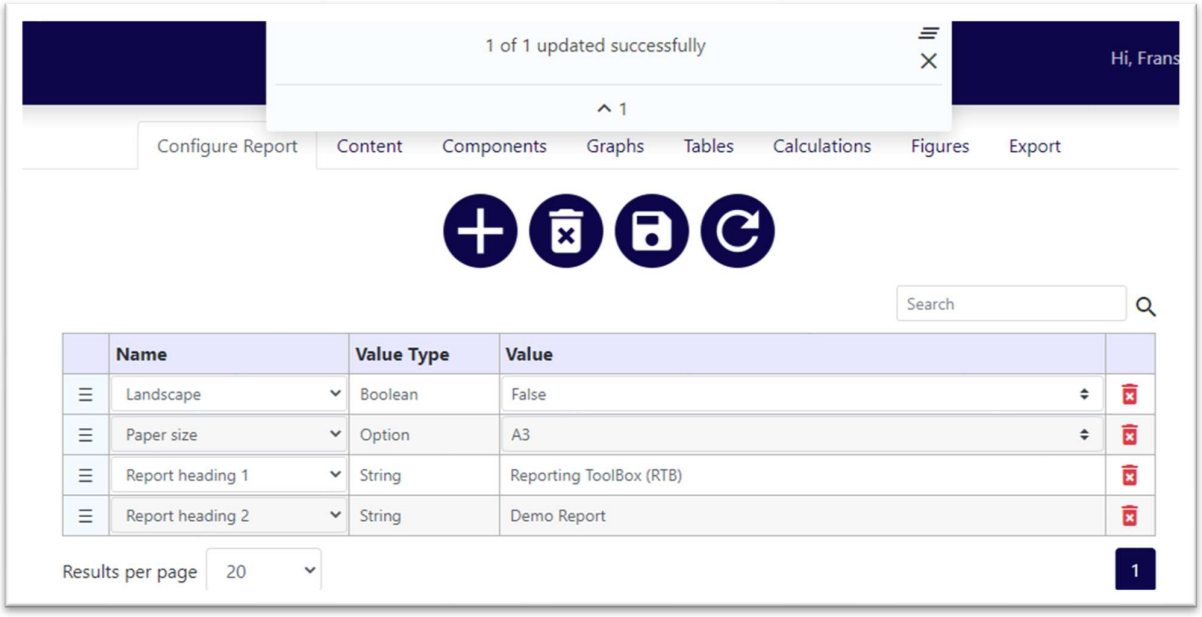


Figure 34: Improved user interface - Return messages

Figure 34 shows the notification messages system. The system keeps the user updated about what is happening and shows a return message at the top of the display for each action. Error messages are also displayed in the same location. The system keeps a history of the feedback messages unless the user explicitly dismisses a message.

If the system is unable to perform an action, the user is notified through a message conveying what the problem was, and which item had the problem.

User interface design: version control

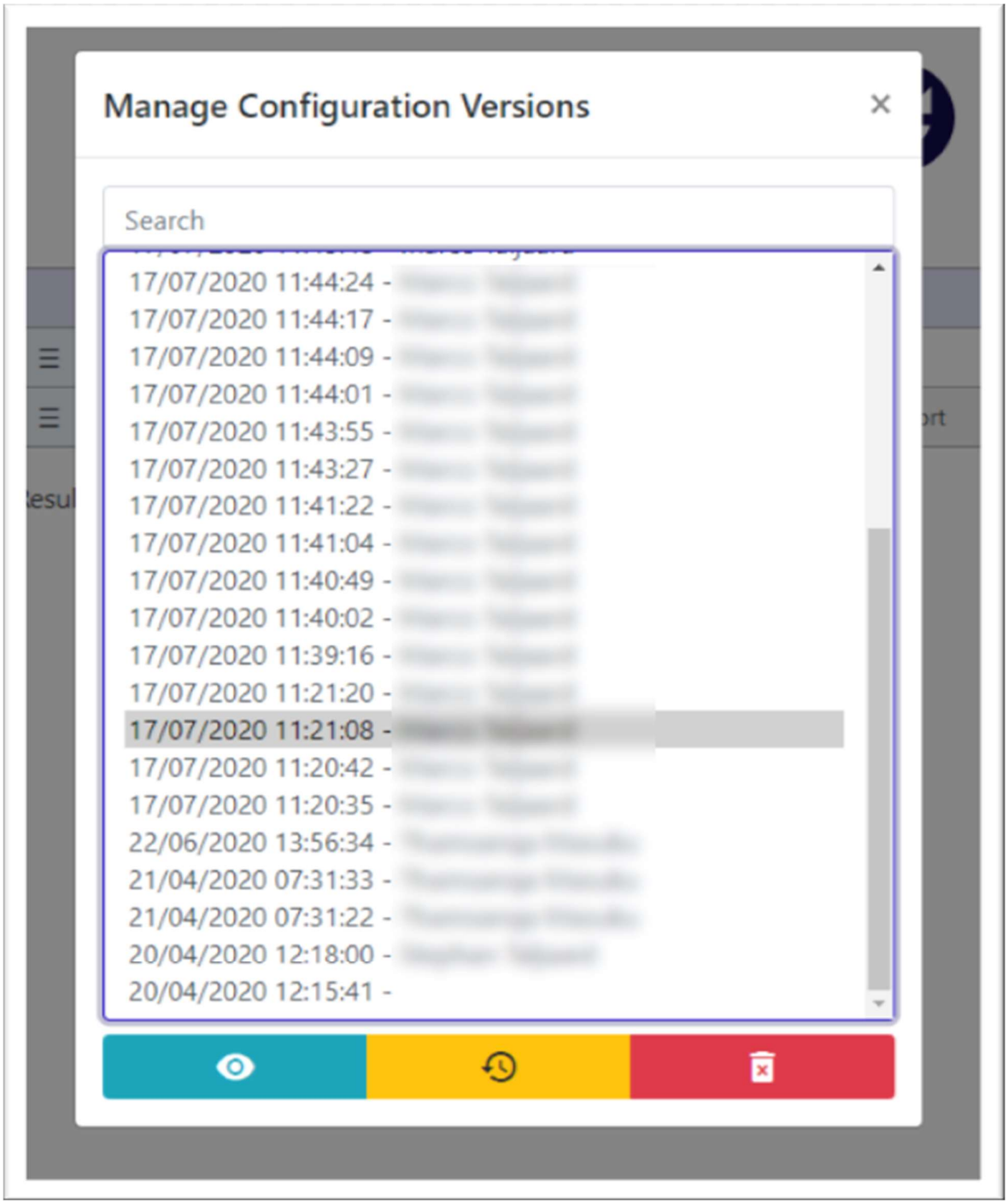


Figure 35: Improved user interface - Version control

Figure 35 shows the version control management user interface. Every change made on a report is logged in a database, which enables the user to see when the last change was made. The user

who made the change is also stored. A user with sufficient access rights could restore the report to a previous version at any point.

User interface design: documentation

The documentation for the system is easily accessible through a similar interface and contains a short guide on how to get started, followed by an in-depth guide on every part of the system. Available options and explanations of each option are also present.

User interface design: cloning

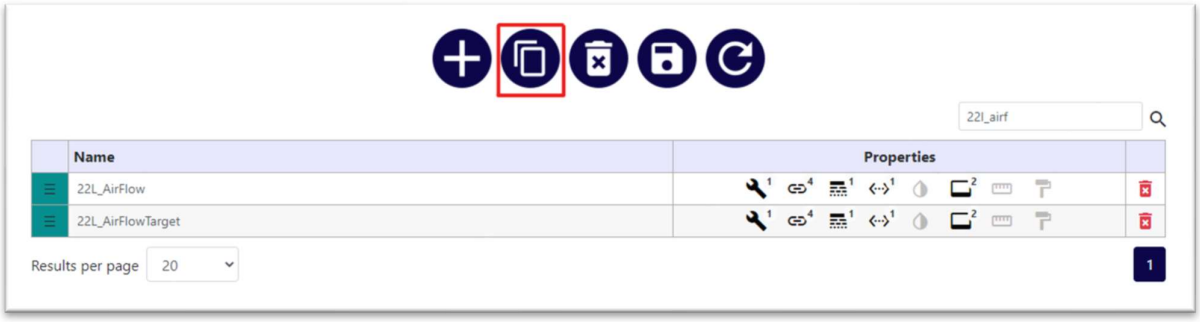


Figure 36: Improved user interface - Clone button

Every main object can easily be cloned by selecting the objects using the handles in the leftmost column, then clicking the Clone Objects button (marked in red on Figure 36). The result is a deep copy of the object, with the name changed to reflect the copy status.

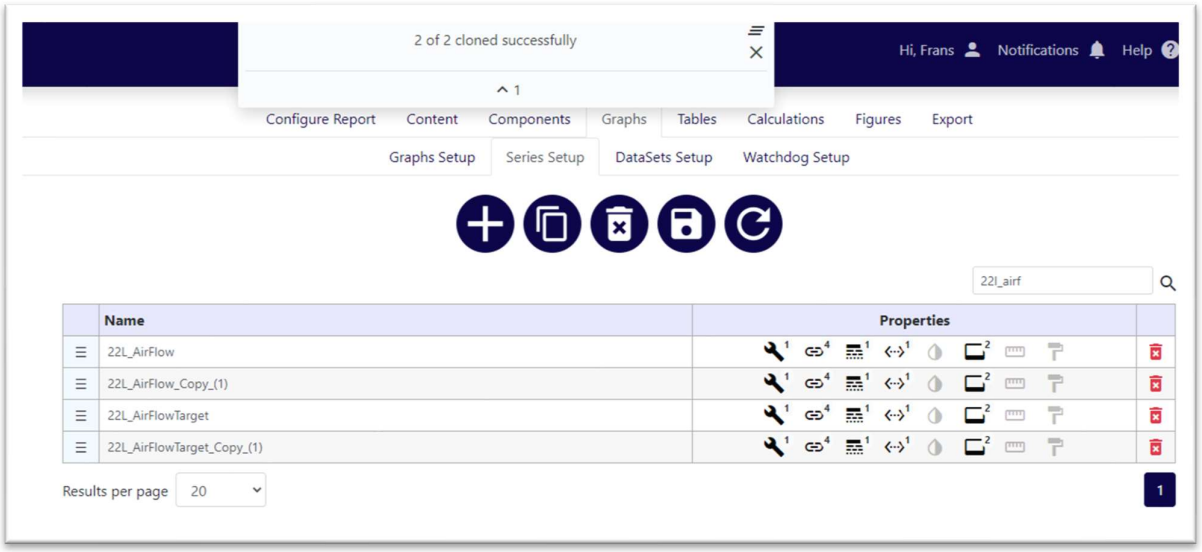


Figure 37: Improved user interface - Clone result

As seen in Figure 37, the two select objects were cloned successfully, with the clones having _Copy_(1) appended to their respective names.

User interface design: shortcuts

The new user interface includes keyboard shortcuts to perform quick actions such as saving changes or confirming actions. This increases efficiency and enables advanced users to perform work much quicker than before. For example, pressing the Control + Enter keys will immediately save any changes made to the active grid.

Other improvements

The new user interface allows for much better security and access control, as previously any user could upload report templates. Portability also improved due to the new user interface being web-based and can thus be accessed from any device with a web browser and internet access.

4.5 Post-improvement results

Post-improvement questionnaire results

The first part of the methodology validation was to obtain user feedback about the improved system and compare the results to the initial results obtained when the old system was still in

use.19 responses were received out of 22, and the response rate is 86.4%. Because the questionnaires were filled in anonymously, the split between experienced users and new users could not be determined.

Similar to the first batch of questionnaires, the second set of results were compiled and analysed after the responses were closed. The Cronbach's alpha value for each factor's questions was calculated, as well as the average combined score for each factor.

Table 21: Post-improvement results - Functional Suitability

Subfactors	Average rating	Combined score	Percentage improvement	Cronbach's alpha
Functional completeness	3.89 / 5	4.25 / 5	11.73%	0.299
Functional correctness	4.63 / 5			
Functional appropriateness	4.21 / 5			

Table 21 shows a summary of the post-improvement results for functional suitability. All of the factors received good ratings which indicated that the system provided all of the functionality that is expected. The alpha value was 0.29 which indicates a lower than ideal level of consistency between respondents. This requires that the data be inspected manually, and other statistics were analysed to gain a conclusion of the validity of the results. There were no missing answers from any of the respondents.

Closer inspection of the data reveals a minimum value of three out of five for all three subfactors, and an average value of 4.25 across all subfactors. The mode value for functional completeness as well as functional appropriateness is 4, while the mode value for functional correctness is 5. The low alpha value is due to high amount of variance in the answers, but the mode value of 4 for 2 of the subfactors confirms that the values can be accepted.

An 11.73% improvement over the initial ratings was achieved. The functionality was not a focus of the improved user interface but improved nonetheless because the system was easier to use and as a result, users could more easily find the functionality they needed from the system.

The statistics lead to the conclusion that the inconsistent values may be as a result of a lower amount of responses compared to the amount of questions.

Table 22: Post-improvement results - Reliability

Subfactors	Average rating	Combined Score	Percentage improvement	Cronbach's alpha
Maturity	3.37 / 5	3.91 / 5	10.95%	0.631
Availability	4.42 / 5			
Fault tolerance	3.42 / 5			
Recoverability	4.42 / 5			

Table 22 shows a summary of the post-improvement results for reliability. The alpha value was 0.631, indicating adequate consistency between the questions. Again, all of the questions were answered by each respondent.

Maturity received a slightly lower score at 3.37 compared to the initial result of 3.43. A possible cause for the decrease is that new users do not necessarily understand the difference between an erroneous report setup compared to a technical failure on the system's part.

The rest of the results showed relatively small improvements, with an average improvement of 10.95%. Fault tolerance improved substantially from 2.8 to 3.42, proving that the new user interface helped to decrease the number of faulty report setups, and the constant user feedback given by the system helps users to better understand the problem. The average user rating of recoverability improved from 3.93 to 4.42 indicating that the report version control system also improved system reliability.

Overall, reliability showed the smallest improvement over the initial results. This is expected as the reliability was not a focus of the improvements in any way.

Table 23: Post-improvement results - Compatibility

Subfactors	Average rating	Combined score	Percentage improvement	Cronbach's alpha
Co-existence	4.0 / 5	4.06 / 5	16.99%	0.889
Interoperability	4.11 / 5			

Table 23 shows a summary of the post-improvement results for compatibility. The alpha value was 0.889 which is well above the recommended level of 0.7, again indicating good consistency between the subfactors. A single respondent failed to provide answers for both questions, which results in a non-response rate of 5.26% for both questions.

Co-existence improved from 3.27 to an average rating of 4.0 out of 5. Interoperability improved from 3.67 to 4.11. Because the new user interface was similar to existing software within the company, the users felt that the user interface facilitated compatibility between the new system and other systems which augment NRS's functionality. Overall, the average ratings improved by 16.99%.

Table 24: Post-improvement results - Security

Subfactors	Average rating	Combined score	Percentage improvement	Cronbach's alpha
Confidentiality	4.67 / 5	4.23 / 5	30.27%	0.888
Integrity	4.44 / 5			
Non-repudiation	3.65 / 5			
Accountability	4.12 / 5			
Authenticity	4.22 / 5			

Table 24 shows a summary of the post-improvement results for security. The alpha value was 0.888, indicating good consistency between all of the answers. A single respondent failed to submit an answer for three out of five questions, thus eliminating his responses for this factor. Two other respondents failed to respond to a single question each, which left the non-response rate at 10.53% for non-repudiation and authenticity, and 5.26% for the rest of the subfactors.

The new user interface includes proper user rights and access control, so it was expected that the user perception of security would increase.

The result is a marked improvement of 30.27% across all subfactors compared to the initial ratings. Confidentiality improved from 3.8 to 4.67, while integrity improved from 3.27 to 4.44. Non-repudiation improved from 2.73 to 3.65. The new user interface's version control view contributed towards 58.37% improvement in accountability from 2.6 to 4.12. Lastly, authenticity improved from 3.85 to 4.22.

Table 25: Post-improvement results - Usability

Subfactors	Average rating	Combined score	Percentage improvement	Cronbach's alpha
Appropriateness recognisability	3.16 / 5	3.54 / 5	33.45%	0.848
Learnability	3.16 / 5			
Operability	3.32 / 5			
User error protection	3.79 / 5			
User interface aesthetics	4.37 / 5			
Accessibility	3.47 / 5			

Table 25 shows a summary of the post-improvement results for usability. The alpha value was 0.848, indicating good consistency between respondents. Responses were received for all questions from all respondents.

The largest increase is expected here, as usability was the main focus of the improvements. A new user interface was developed from the ground up with usability in mind, and the new user interface resulted in an overall improvement from the initial ratings of 33.45%, which is the highest percentage improvement out of all of the factors.

Appropriateness recognisability increased from only 2.27 to 3.16, which means that the users are more likely to recognise that the functionality suits their needs and as a result, they are more likely

to continue using the system. While this is a big increase over the original rating, it received the lowest overall post-improvement score out of all of the subfactors.

Learnability improved from 2.6 to 3.16, while operability improved from 2.93 to 3.32. This is a substantial improvement over the original ratings; however, the final scores remain subpar. There is still room for improvement in these two subfactors.

User error protection saw a good increase from 2.53 to 3.79, which shows that the user interface massively decreases the number of basic errors that users experience.

The user interface aesthetics saw the biggest single subfactor increase from 2.67 to 4.37, which is an increase of 63.82%. Thus, the new user interface was accepted by users and it is confirmed that the correct design methodology was followed. Lastly, as a result of the new user interface's simpler and more consistent layout, the accessibility improved from 2.93 to 3.47.

Table 26: Post-improvement results - Performance Efficiency

Subfactors	Average rating	Combined score	Percentage improvement	Cronbach's alpha
Time behaviour	3.78 / 5	3.89 / 5	17.51%	0.804
Resource utilisation	4.13 / 5			
Capacity	3.83 / 5			

Table 26 shows a summary of the post-improvement results for performance efficiency. The alpha value was 0.804 which confirms good consistency between the answers. Three respondents failed to submit a response to the question related to resource utilisation, and as a result, the non-response rate was 15.79%. The other two questions had non-response rates of 5.26% each with a single response missing from each.

The overall rating improved by 17.51%. Performance efficiency was not a focus of the improvements; however, the new user interface does provide easier and more efficient access to the system. Most of the processing required to set up a report was offloaded to the server on which the system is running, thus freeing up the user's system resources.

Time-behaviour improved from 3.0 to 3.78, showing that the new user interface is substantially faster than the old method of uploading report templates. Resource utilisation improved from 3.61 to 4.13, and capacity improved from 3.17 to 3.83.

Table 27: Post-improvement results - Portability

Subfactors	Average rating	Combined score	Percentage improvement	Cronbach's alpha
Adaptability	3.64 / 5	4.07 / 5	30.95%	0.696
Installability	4.39 / 5			
Replaceability	4.06 / 5			

Table 27 shows a summary of the post-improvement results for portability. The alpha value was 0.696 indicating good consistency between questions.

A single respondent failed to submit an answer for two out of three questions, rendering all of his responses invalid for this factor as the number of unanswered questions was more than 50%. Four other respondents failed to submit an answer to the question related to adaptability. This leaves the non-response rate for adaptability at 26.32%, while installability and replaceability both had a non-response rate of 5.26%.

Adaptability saw a big increase in average ratings from 2.46 to 3.64. The new user interface can be used on any device with internet access and a web browser, whereas the previous user interface could only be used on a computer. Installability also increased substantially from 3.40 to 4.39, as the system can easily be accessed without prior setup. Replaceability increased from 3.47 to 4.06, showing that users believe the new system can effectively replace similar systems.

Post-improvement usage results

The second method of validation was to compare the number of users with historic data to see if the improved system led to increased usage figures. During the improvement phase of the case study, the system on which the user interface is located received a user activity tracking system which enables the ability to see the exact number of active users for each month on the system.

Figure 38: Number of users per month

Figure 38 shows the number of users for each month since January 2019. In 2019, three presentations and training sessions were held by the company in an attempt to increase user adoption of NRS. A company-wide training session that included all of the potential users was held in March 2019, ***indicated by the red arrow***. This had only a small effect on the number of active users.

The research surrounding user acceptance led to the decision to create a new user interface, which was released on 31 March 2020, ***indicated by the green arrow***. The changes were announced through another company-wide presentation and training session. This led to a dramatic increase in active users, as can be seen in Figure 38. The number of active users increased by 133% from March 2020 to the time of writing in October 2020. There was also two months where the system saw no growth in users. This is not ideal, but not unexpected as the less favourable users took longer to see the benefits of the system.

The user acceptance rate of the system increased from 30% to 70% in six months. This substantial increase in users could be attributed to the improvements made to the system. The improvements made to the weakest part of the system(usability) were enough to convince users to try the system and realise the benefits.

4.6 Results summary

By focusing the improvements on the factor with the lowest initial user rating, the weakest aspect of the system was improved. This was enough to convince users to use the system for longer periods than they otherwise would, and led to more and more users realising the benefits that the system brings over time. The amount of active users increased drastically from only 18 at the start of March 2020 to 42 in September 2020. The user acceptance rate improved from 30% to 70% in a few months. This increase in users over time can be seen in Figure 38.

By re-evaluating the users' perception of the system factors through a second round of questionnaires, some insight regarding the cause of the drastic improvements can be gained.

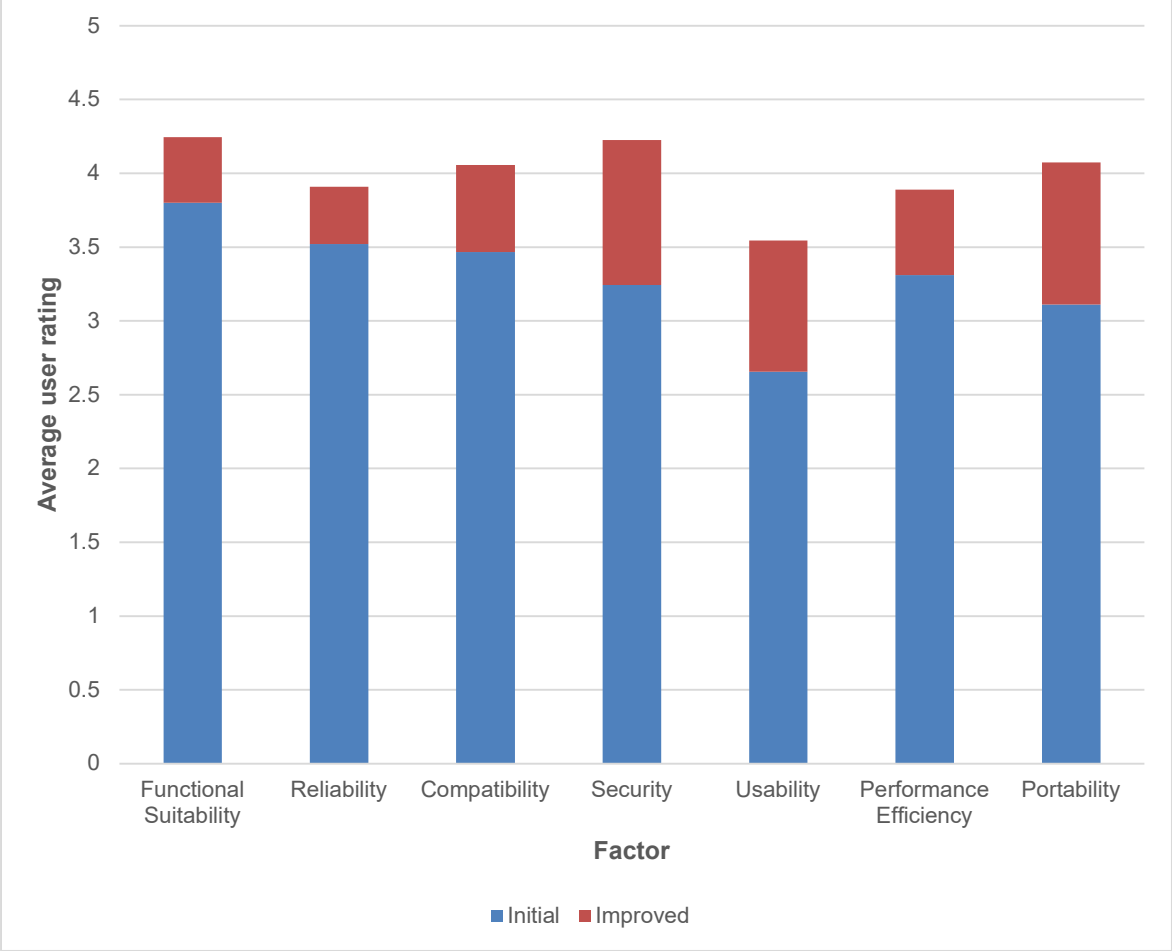


Figure 39: Average post-improvement rating per factor

Figure 39 shows the average initial user rating per factor. Because the usability was the main focus of the improvements, it is expected that usability will see the biggest improvement, with

lesser improvements in other areas. As expected, the biggest percentage improvement was seen in usability. However, the improvements to security and portability were bigger overall. The improvement in the average rating of usability brings it much closer to the rest of the factors, however, at 3.54 it is still the factor that received the lowest score and should still be focused on in the future. However, the drastic improvement in usability brings the minimum user rating for a single factor up from 2.65 to 3.54. This was enough to convince users to use the system, and subsequently the numbers of active users increased over time, leading to increased user adoption of the system.

Functional suitability increased by a smaller, though still significant margin. The core functionality itself did not change much, but users are now more likely to use the system for longer periods, thereby realising the benefits that the system provides. As a result, the factors related to PU received much better user ratings than the initial version of the system.

The new user interface provided an opportunity for better access control, and as a result, security increased by a big margin. Because the user interface is web-based, portability of the system increased substantially, and the performance increase over the previous methods shows in the increased average rating of performance efficiency.

Reliability increased slightly because of the improved user error reduction methods implemented, and compatibility improved as it was easier to link data sources from other systems to NRS.

Interestingly, improving the usability of the system also affected the other factors with varying degrees of improvement. This aligns with the findings of Rajković et al [37]. By improving a factor related to the Perceived Ease of Use (PEOU), other factors related to the Perceived Usefulness (PU) improved.

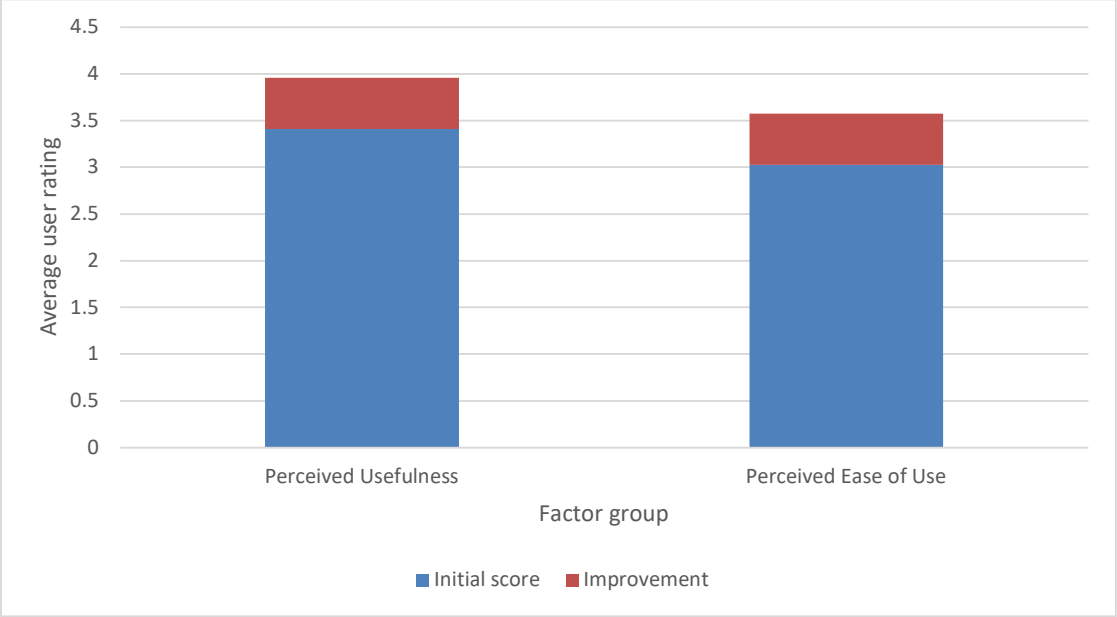


Figure 40: Improvement in PU and PEOU

Both PU and PEOU received notable improvements in user ratings. The average user rating for PEOU improved by 0.55 points, and PU also improved by 0.55 points. The percentage improvement for PEOU was 18.164%, and the percentage improvement for PU was 16.0%. As a result, the percentage improvement was higher for PEOU as expected, even though the overall rating improvement was the same.

The substantial increase in the amount of active users and by extension the user acceptance, as well as the confirmation of increased user PEOU and PU proves that the methodology was effective in increasing user acceptance of software systems.

CHAPTER 5: CONCLUSION

5.1 Preamble

The previous chapter validated the suitability of the methodology through the means of a case study and verified that the methodology is effective in increasing the user acceptance of a software system.

This chapter gives an overview of the problem and methodology, followed by a discussion of the implementation of a case study. The findings are summarised, a conclusion is drawn on the study and recommendations for future studies are provided.

5.2 Summary

Industry 4.0 has driven the digitisation of large scale industrial and mining operations. This introduced the burden of large volumes of data to process. Data from industrial systems is usually processed by specialised systems and software. Numerous software systems exist to process this data and generate reports; however, users may be biased towards working with non-optimal software. This leads to purpose-built software being underused, thus decreasing efficiency and results in wasted resources spent on software development.

From the above background, the following problem statement was derived: User acceptance of software may be affected by personal preference, poorly designed software or a combination of these factors. Using suboptimal software instead may affect the efficiency of a workforce or decrease the profitability of software as resources spent developing the software is wasted. User acceptance of software needs to be improved by identifying and improving the factors that affect user acceptance.

The objectives of the study were the following:

1. Develop a general method to improve user acceptance of an underutilised software system.
2. Implementation of the method on a case study and validating the effectiveness of the methodology.
3. Conclude on the effectiveness of the methodology to improve user acceptance, comment on the findings and provide recommendations for future work.

The first objective was to develop a method to increase user adoption of software by addressing the factors affecting user acceptance. Through the investigation and analysis of existing literature, it was found that the Technology Acceptance Model (TAM) provides a good starting point for improving user acceptance. TAM states that the Perceived Usefulness (PU) and Perceived Ease of Use (PEOU) are the main factors affecting user acceptance.

Secondary factors regarding these two main factors were investigated through literature from others. A method was developed to investigate the state of these factors which uses a scoring and evaluation process. The factors from the ISO 25010 standard was used to formulate a questionnaire to analyse the current state of an underused software system from the perspective of existing users. From the results of the first set of questionnaires, the factor that received the lowest average user rating is selected, and improvements should be made to the selected factor through methods researched from literature. The updated system would then be released to users, and the number of users would be monitored over time. The effectiveness of the improvements is verified using a second set of user questionnaires.

The second objective was to validate the effectiveness of the methodology by implementing it on a case study system. The proposed method was applied on a case study of a reporting and data analytics system with low user acceptance. The baseline for the system performance was obtained through feedback from existing users of the system using the questionnaire that was created. Most of the factors received a good score, however the low usability score was a clear outlier.

The usability of the system was improved by using methods and processes found in the literature, shown in section 3.6. The new user interface was web-based, and proper access control to the system was implemented. The portability increased, as the system could be accessed on any modern device with a web browser. Reliability also increased, as the new user interface helps users to provide the correct input, thereby reducing the number of errors experienced by the system.

The updated system was released, and several training sessions on the new user user interface were held by the company. The methodology proved effective in increasing the user acceptance of the case study system it was tested on and the questionnaire provided a good overview of the state of the system before improvements were made. The number of users increased substiantially after the new user interface was released and user acceptance increased from 30% to 70%.

Qualitative results were also obtained via additional user feedback that evaluated the state of the system after the improvements. The results showed an improvement in all factors, with the usability showing the biggest percentage improvement of 33.5% over the original average rating.

The last objective was to conclude on the effectiveness on the study, comment on the findings and provide recommendations for future work. By applying the principles from the Technology Acceptance Model, the user acceptance of a software system was successfully improved through improving the factors that affect acceptance. By focusing the improvements of the system on the factor with the lowest initial user rating, the system improved by a significant margin, and user adoption grew.

It was also found that the improvements made to factors related to ease of use contributed to perceived improvements to the usefulness of the system. This confirms the findings of Rajković et al [37]. By improving the PEOU, the PU improved as well, leading to increased user adoption of the system.

The improvements in both the user acceptance percentage as well as the improvement in the average user ratings of the software factor confirm that the methodology was effective in increasing user adoption of software systems.

5.3 Recommendations for future studies

The study achieved the objectives, however, there are still improvements that could be made.

Questionnaire improvements

The questionnaire was effective in providing a good general overview of how users perceived the system. However, users failed to submit answers for some of the questions. This did not influence the results obtained in the case study, however, in cases where there are fewer responses, a missing answer could skew the results.

The questionnaire used in the methodology could be modified to ask the user to give direct user ratings about each factor rather than ask the user on which level they agree with a given statement. The option to not submit a response should also be removed from the questionnaire as this could skew the results if there is only a small number of responses. Implementing both of these suggestions will make processing the results much simpler.

Feedback improvements

The methodology was effective in obtaining a general overview of the factors investigated, however in some cases the areas that required improvement may not fall into the questions provided on the questionnaire.

An optional section could be included in the questionnaire for users to provide feedback about specific aspects that affect their PEOU and PU of the system, and to suggest features that would greatly improve the user experience.

Usage statistics

The results obtained from the case study were sufficient to verify the effectiveness of the methodology. The information obtained could be increased if detailed user statistics are used to analyse current usage, such as the total amount of time that users spend while using the system.

Other statistics such as the type of work could also be analysed to see if more specific improvements are needed. The efficiency of users could also be tracked to see if users can do more work in a specific amount of time after the improvements were made, compared to the same statistics that were recorded before.

Effects of interaction with users

The author did not have any contact with the users beyond distributing the questionnaires via email. As a result, there was not believed to be any impact on the influence that the author may have had on the users' acceptance of the new system. However, the effect of developers and system engineers that take the time to listen to the feedback on users on the potential user acceptance of a system could be investigated in a future study where there is more direct interaction with the users.

Effects of change management

Change management did not play a role in this study. However, with proper change management, the results may improve, as users are gradually introduced to the newer system and shown its benefits while usage of the old system is deprecated. The effects of change management combined with the methodology in this study can be investigated in future studies.

BIBLIOGRAPHY

[1] J. Prinsloo, J. C. Vosloo, and E. H. Mathews, “Towards Industry 4.0: A roadmap for the South African heavy industry sector,” vol. 30, no. November, pp. 90–105, 2019.

[2] D. Horváth and R. Z. Szabó, “Driving forces and barriers of Industry 4.0: Do multinational and small and medium-sized companies have equal opportunities?,” *Technological Forecasting and Social Change*, vol. 146, no. October 2018, pp. 119–132, 2019, doi: 10.1016/j.techfore.2019.05.021.

[3] M. H. Lee *et al.*, “How to respond to the Fourth Industrial Revolution, or the second information technology revolution? Dynamic new combinations between technology, market, and society through open innovation,” *Journal of Open Innovation: Technology, Market, and Complexity*, vol. 4, no. 3, 2018, doi: 10.3390/joitmc4030021.

[4] J. Herman, H. Herman, M. J. Mathews, and J. C. Vosloo, “Using big data for insights into sustainable energy consumption in industrial and mining sectors,” *Journal of Cleaner Production*, vol. 197, pp. 1352–1364, Oct. 2018, doi: 10.1016/j.jclepro.2018.06.290.

[5] H. M. Janse van Rensburg, A. G. S. Gous, J. C. Vosloo, and M. van Heerden, “Improving data management for environmental reporting in the gold mining industry,” *South African Journal of Industrial Engineering*, vol. 30, no. 3, pp. 163–173, 2019, doi: 10.7166/30-3-2236.

[6] T. Popović, N. Latinović, A. Pešić, Ž. Zečević, B. Krstajić, and S. Djukanović, “Architecting an IoT-enabled platform for precision agriculture and ecological monitoring: A case study,” *Computers and Electronics in Agriculture*, vol. 140, pp. 255–265, Aug. 2017, doi: 10.1016/j.compag.2017.06.008.

[7] S. Burke, Rick; Mussomeli, Adam; Stephen, Laaper; Marty, Hartigan; Brenna, “The smart factory,” *The smart factory Responsive, adaptive, connected manufacturing*, 2017, doi: 10.1007/978-3-658-16502-4.

[8] J. C. Kabugo, S. L. Jämsä-Jounela, R. Schiemann, and C. Binder, “Industry 4.0 based process data analytics platform: A waste-to-energy plant case study,” *International Journal of Electrical Power and Energy Systems*, vol. 115, no. July 2019, p. 105508, 2020, doi: 10.1016/j.ijepes.2019.105508.

[9] N. Sheikh, *Implementing Analytics*. Elsevier, 2013.

[10] L. Andika, A. Dyck, and H. Lichter, “Towards A Design for An Extendable Reporting Interface,” in *Procedia Computer Science*, 2017, vol. 116, pp. 318–325, doi: 10.1016/j.procs.2017.10.082.

[11] O. Isik, M. Jones, and A. Sidorova, “Business Intelligence (BI) success and the role of BI capabilities,” *Intelligent systems in accounting, finance and management*, vol. 176, no. January, pp. 161–176, 2011, doi: 10.1002/isaf.

[12] M. Miškuf, I. Zolotová, and M. Nemčík, “Application of business intelligence solutions from microsoft and IBM on manufacturing data,” in *SAMI 2016 - IEEE 14th International Symposium on Applied Machine Intelligence and Informatics - Proceedings*, Mar. 2016, pp. 41–45, doi: 10.1109/SAMI.2016.7422979.

[13] F. D. Davis, “Perceived usefulness, perceived ease of use, and user acceptance of information technology,” *MIS Quarterly: Management Information Systems*, vol. 13, no. 3, pp. 319–339, 1989, doi: 10.2307/249008.

[14] J. Piñeiro-Chousa, M. Vizcaíno-González, and J. Caby, “Financial development and standardized reporting: A comparison among developed, emerging, and frontier markets,” *Journal of Business Research*, vol. 101, no. June 2018, pp. 797–802, 2019, doi: 10.1016/j.jbusres.2018.12.012.

[15] M. Kubina, G. Koman, and I. Kubinova, “Possibility of Improving Efficiency within Business Intelligence Systems in Companies,” *Procedia Economics and Finance*, vol. 26, no. 15, pp. 300–305, 2015, doi: 10.1016/s2212-5671(15)00856-4.

[16] F. D. Davis, “A technology acceptance model for empirically testing new end-user information systems : theory and results,” 1985.

[17] F. D. Davis, R. P. Bagozzi, and P. R. Warshaw, “User Acceptance of Computer Technology: A Comparison of Two Theoretical Models,” *Management Science*, vol. 35, no. 8, pp. 982–1003, 1989, doi: 10.1287/mnsc.35.8.982.

[18] S. Verma, S. S. Bhattacharyya, and S. Kumar, “An extension of the technology acceptance model in the big data analytics system implementation environment,” *Information Processing and Management*, vol. 54, no. 5, pp. 791–806, 2018, doi: 10.1016/j.ipm.2018.01.004.

[19] F. D. Davis and V. Venkatesh, “Toward preprototype user acceptance testing of new information systems,” *IEEE Transactions on Engineering Management*, vol. 51, no. 1, pp. 31–46, 2004, doi: 10.1109/TEM.2003.822468 T4 - Implications for software project management M4 - Citavi.

[20] S. Mamone, “The IEEE standard for software maintenance,” *ACM SIGSOFT Software Engineering Notes*, vol. 19, no. 1, pp. 75–76, Jan. 1994, doi: 10.1145/181610.181623.

[21] H. Taherdoost, “A review of technology acceptance and adoption models and theories,” *Procedia Manufacturing*, vol. 22, pp. 960–967, 2018, doi: 10.1016/j.promfg.2018.03.137.

[22] K. Sohn and O. Kwon, “Technology acceptance theories and factors influencing artificial Intelligence-based intelligent products,” *Telematics and Informatics*, vol. 47, no. December 2019, pp. 1–14, 2020, doi: 10.1016/j.tele.2019.101324.

[23] T. J. Madden, P. S. Ellen, and I. Ajzen, “A Comparison of the Theory of Planned Behavior and the Theory of Reasoned Action,” *Personality and Social Psychology Bulletin*, vol. 18, no. 1, pp. 3–9, 1992, doi: 10.1177/0146167292181001.

[24] R. J. Hill, M. Fishbein, and I. Ajzen, “Belief, Attitude, Intention and Behavior: An Introduction to Theory and Research.,” *Contemporary Sociology*, vol. 6, no. 2, p. 244, Mar. 1977, doi: 10.2307/2065853.

[25] V. Venkatesh and F. D. Davis, “Theoretical extension of the Technology Acceptance Model: Four longitudinal field studies,” *Management Science*, vol. 46, no. 2, pp. 186–204, 2000, doi: 10.1287/mnsc.46.2.186.11926.

[26] V. Venkatesh, J. Y. L. Thong, and X. Xu, “Unified theory of acceptance and use of technology: A synthesis and the road ahead,” *Journal of the Association of Information Systems*, vol. 17, no. 5, pp. 328–376, 2016.

[27] I. Ajzen, “From Intentions to Actions: A Theory of Planned Behavior,” *Action Control*, pp. 11–39, 1985, doi: 10.1007/978-3-642-69746-3_2.

[28] I. Ajzen, “The theory of planned behaviour: Reactions and reflections,” *Psychology and Health*, vol. 26, no. 9, pp. 1113–1127, 2011, doi: 10.1080/08870446.2011.613995.

[29] M. M. Tsang, S. C. Ho, and T. P. Liang, “Consumer attitudes toward mobile advertising: An empirical study,” *International Journal of Electronic Commerce*, vol. 8, no. 3, pp. 65–78,

2004, doi: 10.1080/10864415.2004.11044301.

[30] T. Ramayah, K. Rouibah, M. Gopi, and G. J. Rangel, “A decomposed theory of reasoned action to explain intention to use Internet stock trading among Malaysian investors,” *Computers in Human Behavior*, vol. 25, no. 6, pp. 1222–1230, 2009, doi: 10.1016/j.chb.2009.06.007.

[31] K. Rouibah, R. Thurasamy, and O. S. May, “User Acceptance of Internet Banking In Malaysia: Test of Three Competing Models,” *International Journal of E-Adoption (IJEA)*, vol. 1, no. 1, pp. 1–19, 2009, doi: 10.4018/jea.2009010101.

[32] I. Ajzen, “The theory of planned behavior,” *Organizational Behavior and Human Decision Processes*, vol. 50, no. 2, pp. 179–211, Dec. 1991, doi: 10.1016/0749-5978(91)90020-T.

[33] N. O. Ndubisi, “Factor of online learning adoption: a comparative juxtaposition of the theory of planned behaviour and the technology acceptance model.,” *International Journal on E-learning*, vol. 5, no. 4, pp. 571–591, 2006.

[34] N. O. Ndubisi and N. C. Chukwunonso, “On-line learning adoption intention : Comparing the predictive power of two competing models,” *HERDSA 2004 Conference Proceedings*, no. 1998, pp. 242–251, 2004.

[35] T. Teo, M. Zhou, and J. Noyes, “Teachers and technology: development of an extended theory of planned behavior,” *Educational Technology Research and Development*, vol. 64, no. 6, pp. 1033–1052, Dec. 2016, doi: 10.1007/s11423-016-9446-5.

[36] C. Martins, T. Oliveira, and A. Popovič, “Understanding the internet banking adoption: A unified theory of acceptance and use of technology and perceived risk application,” *International Journal of Information Management*, vol. 34, no. 1, pp. 1–13, 2014, doi: 10.1016/j.ijinfomgt.2013.06.002.

[37] P. Rajković, D. Aleksić, D. Janković, A. Milenković, and I. Petković, “Checking the potential shift to perceived usefulness—The analysis of users’ response to the updated electronic health record core features,” *International Journal of Medical Informatics*, vol. 115, no. March, pp. 80–91, 2018, doi: 10.1016/j.ijmedinf.2018.04.011.

[38] M. J. Mortenson and R. Vidgen, “A computational literature review of the technology acceptance model,” *International Journal of Information Management*, vol. 36, no. 6, pp. 1248–1259, 2016, doi: 10.1016/j.ijinfomgt.2016.07.007.

[39] F. Lin, S. S. Fofanah, and D. Liang, “Assessing citizen adoption of e-Government initiatives in Gambia: A validation of the technology acceptance model in information systems success,” *Government Information Quarterly*, vol. 28, no. 2, pp. 271–279, 2011, doi: 10.1016/j.giq.2010.09.004.

[40] Y. Lee, K. A. Kozar, and K. R. T. Larsen, “The Technology Acceptance Model: Past, Present, and Future,” *Communications of the Association for Information Systems*, vol. 12, no. December, 2003, doi: 10.17705/1cais.01250.

[41] G. H. Subramanian, “A Replication of Perceived Usefulness and Perceived Ease of Use Measurement,” *Decision Sciences*, vol. 25, no. 5–6, pp. 863–874, 1994, doi: 10.1111/j.1540-5915.1994.tb01873.x.

[42] D. A. Adams, R. R. Nelson, and P. A. Todd, “Perceived usefulness, ease of use, and usage of information technology: A replication,” *MIS Quarterly: Management Information Systems*, vol. 16, no. 2, pp. 227–247, 1992, doi: 10.2307/249577.

[43] M. P. Bach, A. Čeljo, and J. Zoroja, “Technology Acceptance Model for Business Intelligence Systems: Preliminary Research,” *Procedia Computer Science*, vol. 100, pp. 995–1001, 2016, doi: 10.1016/j.procs.2016.09.270.

[44] S. D. Beesley, J. T. Patrie, and C. M. Gaskin, “Radiologist Adoption of Interactive Multimedia Reporting Technology,” *Journal of the American College of Radiology*, vol. 16, no. 4, pp. 465–471, 2019, doi: 10.1016/j.jacr.2018.10.009.

[45] C. M.Y., “Overview of the Technology Acceptance Model: Origins, Developments and Future Directions,” *Sprouts: Working Papers on Information Systems*, vol. 9, no. 37, 2009.

[46] M. Gellerstedt, S. M. Babaheidari, and L. Svensson, “A first step towards a model for teachers’ adoption of ICT pedagogy in schools,” *Heliyon*, vol. 4, no. 9, p. e00786, Sep. 2018, doi: 10.1016/j.heliyon.2018.e00786.

[47] I. C. Pappa, C. Iliopoulos, and T. Massouras, “What determines the acceptance and use of electronic traceability systems in agri-food supply chains?,” *Journal of Rural Studies*, vol. 58, pp. 123–135, Feb. 2018, doi: 10.1016/j.jrurstud.2018.01.001.

[48] Venkatesh, Morris, Davis, and Davis, “User Acceptance of Information Technology: Toward a Unified View,” *MIS Quarterly*, vol. 27, no. 3, p. 425, 2003, doi: 10.2307/30036540.

[49] P. K. Chopdar, N. Korfiatis, V. J. Sivakumar, and M. D. Lytras, “Mobile shopping apps adoption and perceived risks: A cross-country perspective utilizing the Unified Theory of Acceptance and Use of Technology,” *Computers in Human Behavior*, vol. 86, pp. 109–128, Sep. 2018, doi: 10.1016/j.chb.2018.04.017.

[50] M. J. Kim and C. M. Hall, “What drives visitor economy crowdfunding? The effect of digital storytelling on unified theory of acceptance and use of technology,” *Tourism Management Perspectives*, vol. 34, p. 100638, Apr. 2020, doi: 10.1016/j.tmp.2020.100638.

[51] Venkatesh, Thong, and Xu, “Consumer Acceptance and Use of Information Technology: Extending the Unified Theory of Acceptance and Use of Technology,” *MIS Quarterly*, vol. 36, no. 1, p. 157, 2012, doi: 10.2307/41410412.

[52] K. Baishya and H. V. Samalia, “Extending unified theory of acceptance and use of technology with perceived monetary value for smartphone adoption at the bottom of the pyramid,” *International Journal of Information Management*, vol. 51, p. 102036, Apr. 2020, doi: 10.1016/j.ijinfomgt.2019.11.004.

[53] *ISO/IEC 25010: Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models*, 1st ed. 2011.

[54] N. Bevan, “Quality in use: Meeting user needs for quality,” *Journal of Systems and Software*, vol. 49, no. 1, pp. 89–96, 1999, doi: 10.1016/S0164-1212(99)00070-9.

[55] J. Estdale and E. Georgiadou, “Applying the ISO/IEC 25010 Quality Models to Software Product,” *Communications in Computer and Information Science*, vol. 896, no. December, pp. 492–503, 2018, doi: 10.1007/978-3-319-97925-0_42.

[56] D. Gade, “The Evaluation of Software Quality,” M.Eng, Faculty of Engineering, University of Nebraska - Lincoln, 2013.

[57] “ISO/IEC/IEEE International Standard - Systems and software engineering--Vocabulary,” *ISO/IEC/IEEE 24765:2017(E)*, pp. 1–541, Aug. 2017, doi: 10.1109/IEEESTD.2017.8016712.

[58] B. Behkamal, M. Kahani, and M. K. Akbari, “Customizing ISO 9126 quality model for evaluation of B2B applications,” *Information and Software Technology*, vol. 51, no. 3, pp. 599–609, 2009, doi: 10.1016/j.infsof.2008.08.001.

[59] M. Oriol, J. Marco, and X. Franch, “Quality models for web services: A systematic mapping,” *Information and Software Technology*, vol. 56, no. 10, pp. 1167–1182, 2014, doi: 10.1016/j.infsof.2014.03.012.

[60] J. P. Miguel, D. Mauricio, and G. Rodríguez, “A Review of Software Quality Models for the Evaluation of Software Products,” *International Journal of Software Engineering & Applications*, vol. 5, no. 6, pp. 31–53, 2014, doi: 10.5121/ijsea.2014.5603.

[61] A. Alvaro, E. Santana De Almeida, S. Romero De, and L. Meira, “Quality Attributes for a Component Quality Model,” *10thWOPC/19thECCOP*, pp. 31–37, 2005.

[62] F. Asadi and S. Paydar, “Presenting an evaluation model of the trauma registry software,” *International Journal of Medical Informatics*, vol. 112, no. January, pp. 99–103, 2018, doi: 10.1016/j.ijmedinf.2018.01.013.

[63] N. M. Avouris, “An Introduction to Software Usability,” in *8th Panhellenic Conference on Informatics*, 2001, vol. 2, pp. 514–522.

[64] V. Nagaraju, V. Shekar, J. Steakelum, M. Luperon, L. Fiondella, and Y. Shi, “Practical software reliability engineering with the Software Failure and Reliability Assessment Tool (SFRAT),” *SoftwareX*, vol. 10, p. 100357, 2019, doi: 10.1016/j.softx.2019.100357.

[65] G. Mcgraw, “Software security,” *IEEE Security & Privacy Magazine*, vol. 2, no. 2, pp. 80–83, Mar. 2004, doi: 10.1109/MSECP.2004.1281254.

[66] N. Subramanian and L. Chung, “Metrics for Software Adaptability,” *Software Quality Management, SQM 2001*, pp. 95–108, 2001.

[67] M. C. Van der Bank, “A modular framework for the effective development of web-based applications.,” M.Eng, Faculty of Engineering ,North-West University, 2018.

[68] J. Botma, “An integrated software platform for the efficient management of projects,” M.Eng, Faculty of Engineering, North-West University, 2018.

[69] P. Jalote, A. Palit, P. Kurien, and V. T. Peethamber, “Timeboxing: A process model for iterative software development,” *Journal of Systems and Software*, vol. 70, no. 1–2, pp. 117–127, 2004, doi: 10.1016/S0164-1212(03)00010-4.

[70] R. Sherman, *Business Intelligence Guidebook*. Elsevier, 2015.

[71] V. Kannan, S. Jhajharia, and S. Verma, “Agile vs waterfall : A Comparative Analysis,” *International Journal of Science, Engineering and Technology Research (IJSETR)*, vol. 3, no. 10, pp. 2680–2686, 2014.

[72] M. L. Despa, “Comparative study on software development methodologies,” *Database Systems Journal*, vol. V, no. 3, pp. 37–56, Jul. 2013.

[73] G. Coleman and R. Verbruggen, “A quality software process for rapid application development,” *Software Quality Journal*, vol. 7, no. 2, pp. 107–122, 1998, doi: 10.1023/A:1008856624790.

[74] S. Misra, V. Kumar, U. Kumar, K. Fantazy, and M. Akhter, “Agile software development practices: evolution, principles, and criticisms,” *International Journal of Quality & Reliability Management*, vol. 29, no. 9, pp. 972–980, Oct. 2012, doi: 10.1108/02656711211272863.

[75] T. Laukkanen, “Consumer adoption versus rejection decisions in seemingly similar service innovations: The case of the Internet and mobile banking,” *Journal of Business Research*, vol. 69, no. 7, pp. 2432–2439, 2016, doi: 10.1016/j.jbusres.2016.01.013.

[76] R. Scherer, F. Siddiq, and J. Tondeur, “The technology acceptance model (TAM): A meta-analytic structural equation modeling approach to explaining teachers’ adoption of digital technology in education,” *Computers and Education*, vol. 128, no. 0317, pp. 13–35, 2019, doi: 10.1016/j.compedu.2018.09.009.

[77] A. U. Jan and V. Contreras, “Technology acceptance model for the use of information technology in universities,” *Computers in Human Behavior*, vol. 27, no. 2, pp. 845–851, 2011, doi: 10.1016/j.chb.2010.11.009.

[78] N. Antonius, J. Xu, and X. Gao, “Factors influencing the adoption of Enterprise Social Software in Australia,” *Knowledge-Based Systems*, vol. 73, pp. 32–43, 2015, doi: 10.1016/j.knosys.2014.09.003.

[79] M. Day, G. Demiris, D. P. Oliver, K. Courtney, and B. Hensel, “Exploring underutilization of videophones in hospice settings,” *Telemedicine Journal and e-Health*, vol. 13, no. 1, pp. 25–31, 2007, doi: 10.1089/tmj.2006.0023.

[80] Z. Hussein, “Leading to Intention: The Role of Attitude in Relation to Technology Acceptance Model in E-Learning,” in *Procedia Computer Science*, 2017, vol. 105, pp. 159–164, doi: 10.1016/j.procs.2017.01.196.

[81] C. Kim, J. Jahng, and J. Lee, “An empirical investigation into the utilization-based information technology success model: Integrating task-performance and social influence perspective,” *Journal of Information Technology*, vol. 22, no. 2, pp. 152–160, 2007, doi: 10.1057/palgrave.jit.2000072.

[82] D. Nam, J. Lee, and H. Lee, “Business analytics adoption process: An innovation diffusion perspective,” *International Journal of Information Management*, vol. 49, no. August, pp. 411–423, 2019, doi: 10.1016/j.ijinfomgt.2019.07.017.

[83] W. H. DeLone and E. R. McLean, “Information systems success: The quest for the dependent variable,” *Information Systems Research*, vol. 3, no. 1, pp. 60–95, 1992, doi: 10.1287/isre.3.1.60.

[84] L. J. Cronbach, “Coefficient alpha and the internal structure of tests,” *Psychometrika*, vol. 16, no. 3, pp. 297–334, 1951, doi: 10.1007/BF02310555.

[85] A. Gillies, *Software Quality: Theory and Management*. 2011.

[86] Q. Li, “A novel Likert scale based on fuzzy sets theory,” *Expert Systems with Applications*, vol. 40, no. 5, pp. 1609–1618, 2013, doi: 10.1016/j.eswa.2012.09.015.

[87] S. E. Harpe, “How to analyze Likert and other rating scale data,” *Currents in Pharmacy Teaching and Learning*, vol. 7, no. 6, pp. 836–850, 2015, doi: 10.1016/j.cptl.2015.08.001.

[88] K. S. Taber, “The Use of Cronbach’s Alpha When Developing and Reporting Research Instruments in Science Education,” *Research in Science Education*, vol. 48, no. 6, pp. 1273–1296, 2018, doi: 10.1007/s11165-016-9602-2.

[89] R. A. L. F. van Griethuijsen *et al.*, “Global patterns in students’ views of science and interest in science,” *Research in Science Education*, vol. 45, no. 4, pp. 581–603, 2015, doi: 10.1007/s11165-014-9438-6.

[90] J. E. Scott and L. Kaindl, “Enhancing functionality in an enterprise software package,” *Information & management*, vol. 37, no. 3, pp. 111–122, 2000, doi: 10.1016/S0378-7206(99)00040-3.

[91] J. Zhang, Y. Wang, and T. Xie, “Software feature refinement prioritization based on online user review mining,” *Information and Software Technology*, vol. 108, no. May 2018, pp. 30–34, 2019, doi: 10.1016/j.infsof.2018.12.002.

[92] S. Arora, R. B. Misra, and V. M. Kumre, “Software reliability improvement through operational profile driven testing,” in *Annual Reliability and Maintainability Symposium, 2005. Proceedings.*, pp. 621–627, doi: 10.1109/RAMS.2005.1408433.

[93] S. Farooq and S. Quadri, “Evaluating effectiveness of software testing techniques with emphasis on enhancing software reliability,” *Journal of emerging trends in Computing and Information Sciences*, vol. 2, no. 12, pp. 740–745, 2011.

[94] N. Pavlov, A. Iliev, A. Rahnev, and N. Kyurkchiev, “Analysis of the Chen’s and Pham’s software reliability models,” *Cybernetics and Information Technologies*, vol. 18, no. 3, pp. 37–47, 2018, doi: 10.2478/cait-2018-0037.

[95] I. C. Yoon, A. Sussman, A. Memon, and A. Porter, “Effective and scalable software compatibility testing,” *ISSTA’08: Proceedings of the 2008 International Symposium on Software Testing and Analysis 2008*, pp. 63–73, 2008, doi: 10.1145/1390630.1390640.

[96] R. Kumar and R. Goyal, “Modeling Continuous Security: A Conceptual Model for Automated DevSecOps using Open-source software over Cloud (ADOC),” *Computers & Security*, p. 101967, Jul. 2020, doi: 10.1016/j.cose.2020.101967.

[97] J. Nielsen, *Usability Engineering*. Elsevier, 1993.

[98] M. Woodside, G. Franks, and D. C. Petriu, “The future of software performance engineering,” *FoSE 2007: Future of Software Engineering*, pp. 171–187, 2007, doi: 10.1109/FOSE.2007.32.

[99] A. Johansson, J. Svensson, and M. Persson, “Techniques for Software Portability in Mobile Development,” 2009.

[100] I. Malavolta, “Beyond native apps: web technologies to the rescue!,” in *Proceedings of the 1st International Workshop on Mobile Development - Mobile! 2016*, 2016, pp. 1–2, doi: 10.1145/3001854.3001863.

[101] M. G. Avram, “Advantages and Challenges of Adopting Cloud Computing from an Enterprise Perspective,” *Procedia Technology*, vol. 12, pp. 529–534, 2014, doi: 10.1016/j.protcy.2013.12.525.

APPENDIX A: RESEARCH QUESTIONNAIRE (GOOGLE FORMS)

The research questionnaire can be found at the following link:

https://forms.gle/NTQ3GokZpGP1ZFiG8

The following screenshots represent the questionnaire that was used:

Software user evaluation

Thank you for taking the time to fill in this questionnaire. It should take only a few minutes.

The questions pertain to the setup of Modular Reporting (RTB) using the RTB Editor on ATB.

The idea behind the questionnaire is to evaluate the current state of the system in terms of Perceived Usefulness and Perceived Ease of Use.

*Required

Perceived Usefulness

This section contains questions related to the Perceived Usefulness of the system.

1. The system provides all the required functionality. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

2. The system produces the correct results, and the results are accurate. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

3. The system provides all of the appropriate functionality to complete the tasks I use it for. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

4. The system does not crash or fail frequently. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

5. The system is always available. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

6. The system handles errors gracefully. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

7. Work can easily be recovered in the event of a crash. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

8. The system can be used alongside other software with no issues. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

9. The system can exchange accurate information with other systems and use this information correctly. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

10. The work completed on this system is only accessible by users with authorization. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

11. The system prevents unauthorized access and modification of data. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

12. Actions taken while working on the system are traceable and these traces cannot be removed without authorization. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

13. Actions can be traced to a unique user or entity. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

14. The source of the data provided by this system can be proven. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

Perceived Ease of Use

This section contains questions related to the Perceived Ease of Use of the system.

15. This system is easy to understand and intuitive to use. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

16. It is easy to learn how to use the system. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

17. This system is easy to use to get the intended results. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

18. The system helps the user to provide the correct input. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

19. The system is well organized and nice to look at. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

20. The system can be used by people with a wide variety of skill levels. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

21. The system responds quickly to any input. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

22. The system uses resources (e.g. CPU, RAM, Network) efficiently. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

23. The system can handle large workloads or many concurrent users without issues. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

24. The system works well on different platforms and devices. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

25. The system is easy to install or access. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer

26. The system can replace similar systems in the same environment for the same purpose. *

Mark only one oval.

- ◯ Strongly Disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly Agree
- ◯ I prefer not to answer