Towards a measurable standard for software product quality

A quantification of the ISO 25010 software product quality standard

Kouros Pechlivanidis (6527450)

Utrecht University

Master's Thesis
Business Informatics

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands

Supervisor: Dr. Gerard Wagenaar
Second supervisor: Dr. Sietse Overbeek

4 April 2024

Preface

You are about to read the master thesis: "Towards a measurable standard for software product quality: A quantification of the ISO 25010 software product quality standard". This thesis has been written to fulfill the requirements of the master's degree in Business Informatics at Utrecht University. Foremost, I want to express my gratitude to Dr. Gerard Wagenaar for assisting me during this research project during our sparring sessions. Besides that, I want to thank Dr. Sietse Overbeek for fulfilling a role as a second supervisor and taking the time to review my work. My deep appreciation also goes to my parents, who have always put their trust in my academic career over the past five and a half years. I want to express my gratitude to the participants of the focus groups and interviews for their enthusiastic and proactive attitude during their participation in the study. Another word of appreciation goes to my girlfriend, who listened to me and gave her support whenever I was stuck on a problem while writing this thesis. Next, I would like to thank my housemates for providing me with a professional and supportive environment while I was working on this thesis. Finally, I would like to thank my fellow students and colleagues who reviewed my work. Thank you all for your support and I hope you enjoy your reading.

Abstract

To compete in a competitive environment, continuous improvement of software product quality is needed. Improvements in software product quality can be made by applying a software product quality model. One of the most widely discussed software product quality models is the ISO 25010 software product quality model. This study aims to assess the feasibility of measuring software product quality using the ISO 25010 standard through a literature review, focus groups, domain experts interviews, and a case application. 127 quantitative data values were identified for the eight quality characteristics. Subsequently, domain experts evaluated the feasibility of the quantitative data values, concluding that for 52 quality characteristic, there would be no substantial effort in collecting the data, and little technical expertise would be required to make the measurement. These results were evaluated by applying the quantitative data values in a case. Ultimately, results show that 27 metrics could be directly derived from the case. Limitations of the study include the lack of generalisability of results, and a lack of evaluation on the quality of the identified quantitative data values **Keywords:** ISO 25010, Software Product Quality, Metrics

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the topic of this thesis by explaining what software product quality is, why organisations benefit from measuring software product quality and what attempts have been made to measure software product quality. In addition, the motivation of this study will be clarified by demonstrating a gap in literature. The motivation will be concluded by listing the research aim and research questions.

1.1 Software Product Quality

Kitchenham and Pfleeger [27] state that software product quality is a complex and abstract concept that can not be defined by a single accepted definition. Instead, the definition of software product quality is dependent on the given perspective. From a manufacturers' perspective, software product quality can be possibly defined by the conformance to specification, while a user might emphasise fitness to use. A different definition by Fitzpatrick [14] acknowledges all possible perspectives by seeing software product quality as the extent to which an industry-defined set of desirable features are incorporated into a product so as to enhance its lifetime performance. This definition emphasises that the quality of a software product is evaluated to maximise the value delivered by the product over its entire lifetime. A final definition by Pressman [40] further states that software product quality is the conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software. By consolidating the definitions by Fitzpatrick [14] and Pressman [40], we can conclude that when evaluating software product quality, it is important to consider both functional and non-functional requirements, as well as considering that software product quality is evaluated in the dimension of time. To compete in today's market, continuous improvement of software product quality is needed [41]. Software product quality is tightly connected to the economics of the software product. To increase the quality of a software product, the manufacturer of the software product incurs costs. By incurring these costs, the manufacturer aims to generate benefits and reduce future costs that might arise due to a lack of quality [41]. By measuring software product quality, manufacturers can make

well-informed choices on whether the benefits of quality gains outweigh the costs of quality improvement.

1.2 Evaluating Software Product Quality

Significant improvements in software product quality can be made by applying a software product quality model [11]. Over the past two decades, numerous frameworks have been proposed with the aim of providing guidelines for evaluating software product quality. A software product quality model is a model that aims to describe, assess and predict the quality of a software product. A widely discussed attempt at creating a generalisable software product quality model is the ISO 9126 software product quality model [5, 7]. This claim is attributed to the consensus that the strength of the model lies in distinguishing between internal and external quality attributes. Here, internal quality attributes are seen as attributes that can be measured during the development process, where as external quality attributes are attributes that are measured during testing [7]. The model is visualised in Figure 1.1. In 2011, the ISO 9126 standard was replaced by ISO 25010, which extends ISO 9126 by including more features to describe software product quality [1, 38]. The ISO 25010 model is shown in Figure 1.2. ISO 9126 and ISO 25010 are hierarchical quality models. These are models that decompose software product quality into multiple quality characteristics. Each of these quality characteristics can be further decomposed in sub-characteristics. This process is repeated until the chosen level of detail is achieved [49].

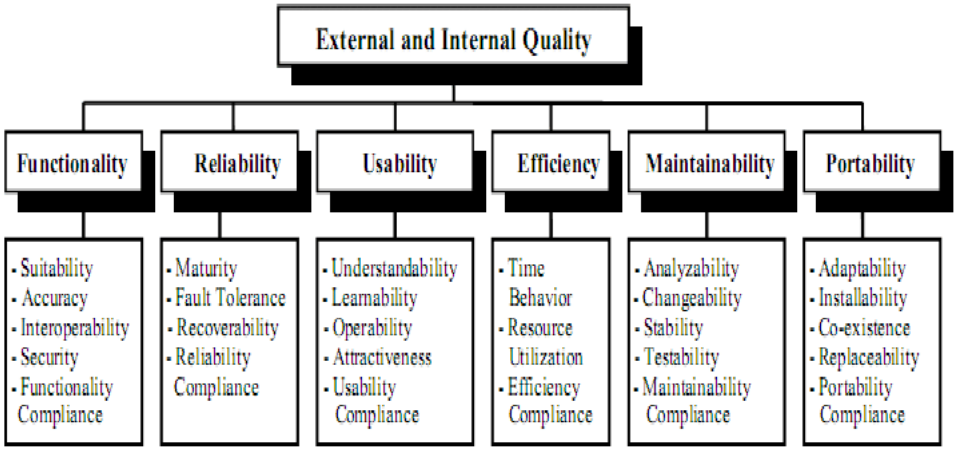

Figure 1.1: The ISO 9126 Software Product Quality Model [31]

1.3 Research Aim and Research Questions

A previous study on ISO 9126 (the precedent of ISO 25010) aimed to measure software product quality by using a questionnaire that asked users of a software product questions

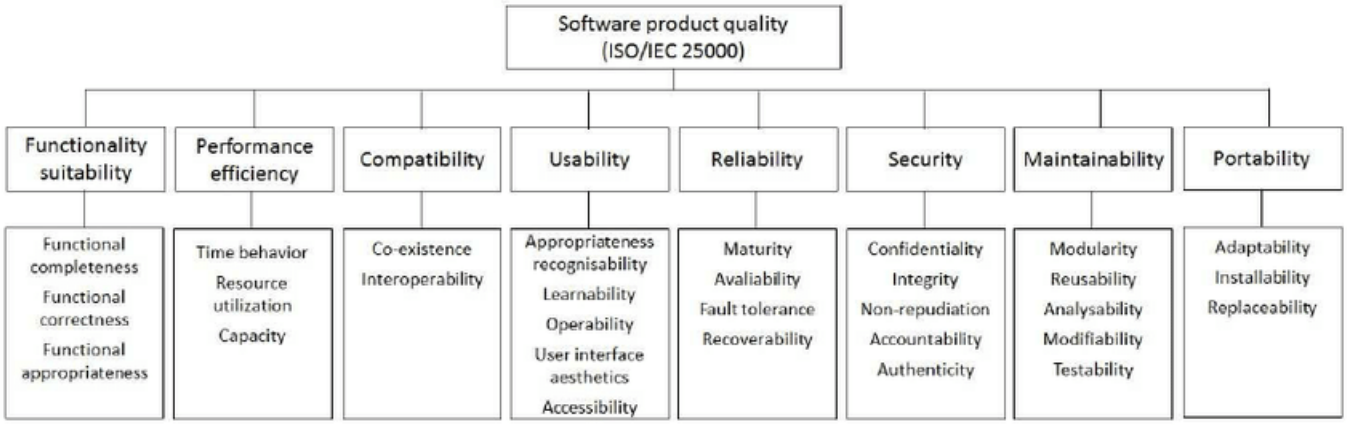

Figure 1.2: The ISO 25010 Software Product Quality Model [15]

on each of the six characteristics in ISO 9126 [25]. While a questionnaire might deem useful for measuring a participants perception on a characteristic, it introduces subjectivity in the evaluation [17]. As subjective findings might contradict facts on quality characteristics, it is an interesting research avenue to find quality metrics that objectively measure each of the characteristics in the software product quality standard. A different study aimed to create an index for software product quality based on ISO 9126 by comparing the importance of each of the six quality characteristics [5]. Subsequently, a checklist was used to test whether the software product satisfied required criteria for fulfilment of the characteristics. The authors conclude that a checklist is an insufficient measure for the quality characteristics and more research on the quantification of the standard's quality characteristics can be done [5]. An attempt to solve the shortcoming of insufficient metrics for quality characteristics was done in the ISO 25023 standard. This standard is used in collaboration with ISO 25010 and provides metrics that can be used to quantify each of the quality attributes in ISO 25020. For instance, ISO 25023 describes that the ISO 25010 quality characteristic *'Performance effiency'* can be measured with the metric *'Errors in task - the number of errors made by the user during a task'*. However, Nakai et al. [32] conclude that not every metric proposed by ISO 25023 sufficiently measures the characteristic it aims to measure, thus demonstrating the need to further study appropriate quantitative measures for ISO 25010. Additionally, not all metrics proposed in ISO 25023 are objective and measurable, making it difficult for organisations to accurately measure the software product quality of their products. Finally, ISO 25023 does not draw any conclusions on how measurable each of the standard's metrics are, making it difficult to apply the standard to real world software products. [4].

As a result of this gap in literature, this study addresses the following main research question:

To what extent can software product quality be empirically measured using the ISO 25010 standard?

By choosing this research question, the scope of the research will be to evaluate to what extent the current ISO 25010 can be used to empirically measure software quality for all software products, therefore choosing not to tailor the standard to a specific domain, accepting the shortcoming that for certain domains an adaptation of the standard might be required. To elaborately answer the main research question, the following sub-questions will be answered:

- **SQ1: How can the characteristics of the ISO 25010 standard be defined?**
  Some characteristics included in ISO 25010, such as usability, are ambiguous and difficult to comprehend [3]. As a result, we need to further define each of the characteristics with the purpose of finding quantitative data values that accurately represent the characteristic according to its definition.

- **SQ2: What quantitative data values can be used to measure each of the standards characteristics?**
  Nakai et al. [32] concluded that the metrics in ISO 25023 do not always accurately represent the construct they are supposed to measure and there are no studies proposing an alternative set of metrics. As a result, quantitative data values that accurately measure the characteristic they are supposed to measure need to be found. Additionally, the scope of answering this sub-question will also be to evaluate whether it is feasible to collect the required data values for a software product.

- **SQ3: To which degree can the quantitative data values identified in SQ2 be measured in practical usage scenarios?**
  After identifying a set of quantitative data values, practical contribution will be provided by evaluating how feasible it is to measure the quantitative data values in a real world situation.

Chapter 2

Research Method

This chapter provides an overview of the research method chosen to conduct this study. For each activity in the research method, an elaboration will be given on how the activity contributes to answering the main research question.

As stated in Section 1.3, one of the research goals will be to find quantitative metrics for the ISO 25010 characteristics and to evaluate how measurable these metrics are. As the goal of this study will be to create an artifact (a set of quantitative data values), the problem under discussion can be considered a design science problem. A design science problem is a problem in which the aim is to (re)design an artifact, so that a chosen goal can be better achieved [10, 50]. In this study, the chosen goal is to find a more factual representation of software product quality by finding quantitative metrics for the ISO 25010 standard.

2.1 Choosing a Design Science Methodology

Research on design science has resulted in the creation of several design science process models [34]. Choosing the appropriate design science process model is dependent on the situation at hand. Venable et al. [48] have attempted to create a set of rules that can be used as a guideline for choosing a suitable design science process model. When choosing a design science process model for our research, the following rules should be considered:

1. The artefact should be demonstrated in a real-world situation, as a requirement for answering SQ3 is to evaluate whether the found quantitative data values can be applied in an actual usage scenario.

2. Research findings have to be communicated in form of a scholarly paper.

The Design Science Research Methodology (DSRM) process model by Peffers et al. [37] satisfies the stated rules as the process model contains a separate demonstration phase , and the communication of findings is done in the form of a research paper. Additionally, Venable et al. [48] emphasise that DSRM is especially useful in cases where extensive adaptation to daily use of the artefact is needed. As a requirement for answering SQ3 is to apply

the the quantitative data values to an actual usage scenario, this criteria can be applied to the design science problem under discussion. The DSRM process model is summarized in Figure 2.1. The next sections will describe how each of the activities in the process model will be applied to our design science problem. A different well-cited design science process model that was considered is the model by Vaishnavni and Kuechler [47], as this model was explicitly created for design research in information systems. However, this model has not been chosen because it does not explicitly describe a demonstration step, which is required according to the previously mentioned rules. A second design science process model reviewed is the design science cycle by Wieringa [50]. This model uses an iterative approach that cycles between identifying problems, designing a treatment, validating the treatment, implementing the treatment in a case and evaluating the treatment. This iterative cycle consists of similar activities as DSRM and was considered as a candidate design science process model for this study. In the end, the model by Peffers et al. [37] has been chosen because it provides explicit guidelines for communicating the results of the design research, which is a requirement for this study. Additionally, the model by Peffers et al. [37] includes dedicated steps for defining the objectives of a solution and designing the solution, where as the design science cycle by Wieringa [50] implicitly mentions both steps in the designing a treatment phase. This distinction helps provide a clearer understanding of each phase.

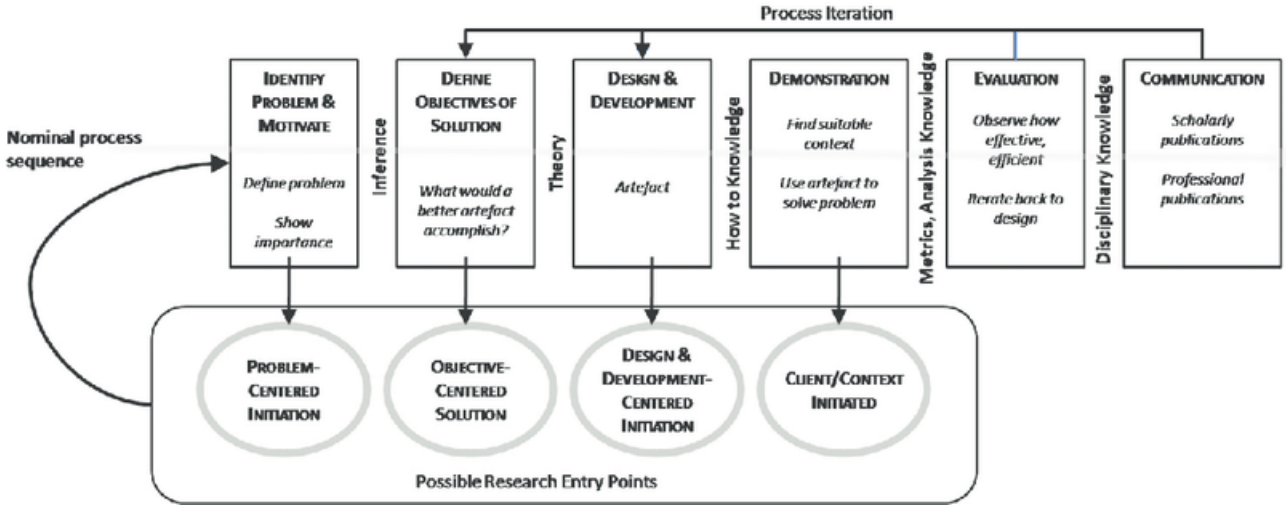

Figure 2.1: The DSRM process model [37]

2.2 Applying DSRM

This section will elaborate on how DSRM will be applied to the situation at hand.

2.2.1 Identify Problem & Motivate

The goal of this activity is to identify the problem to which the designed artefact will be a solution, and to justify the value of a solution [37]. Problem identification and motivation was already performed in Chapter 1 and will therefore not be elaborately discussed in this section. To briefly summarise the problems identified in Chapter 1:

- Previous attempts at measuring software product quality through the ISO standards only used subjective metrics or elementary checklists, which might contradict facts [5, 25].
- The proposed metrics in ISO 25023 do not measure the construct they are supposed to measure well-enough and do not draw conclusions on how measurable each of the standard's metrics are in an actual usage scenario [32].

The value of a solution can be justified through related work that states that improvements in software product quality are necessary to compete in today's market, and significant improvements in software product quality can only be made when applying a software product quality model [11, 41].

2.2.2 Define Objectives of Solution

The second activity in the process model is to define the objectives of a solution. After performing this activity, it should be clear how a solution distinguishes itself from current initiatives. The objectives of a solution should be rationally inferred from the problem identification [37].

By inferring the objectives of a solution from the problem statement, the conclusion can be made that the solution should use quantitative data values for measuring software product quality. The solution should also be evaluated to justify that it solves the main shortcoming of ISO 25023, meaning that the metrics should accurately represent the construct they are supposed to measure. As ISO 25010 is a generalised software product quality model that is not tailored to a certain industry, it is the objective to create an index that can be applied to all sectors as well [18]. As some industries might consider certain quality characteristics as more important than others, it will be the objective of the solution to make the results generalisable to all industries.

2.2.3 Design & Development

The artefact that aims to solve the design science problem will be created during the design & development activity. To develop the artefact, two activities will be performed: a literature review and focus groups.

Literature Review

A literature review is a review of existing literature, with the goal of building a solid theoretical foundation for the proposed study. A literature review can be used to justify the approach of the study and demonstrate a knowledge gap in the topic under discussion [29, 20]. A literature review can be considered systematic if a strict protocol is used for searching and evaluating literature [6]. A systematic literature review is deemed appropriate when it is necessary to address all research on a given topic [33]. In this study, a literature review is used to answer SQ1: "How can the characteristics of the ISO 25010 standard be defined?". Furthermore, the literature study will explore to what extent ISO 25010 has already been applied in corporate settings and what attempts at quantifying the standard have already been made. To achieve these objectives, evaluating all research on the topic is redundant, making a systematic literature review unnecessary and time consuming. However, the literature review can still be planned using a systematic literature review protocol discussed in scientific studies, with the only difference being that the literature review conducted for this study does not address all papers on the topic. Xiao and Watson [51] conclude that each successful systematic literature review uses the same five steps for conducting the review, namely:

1. Search the literature
2. Screen for inclusion
3. Assess quality
4. Extract data
5. Analyze and synthesize data

In the first step, literature will be searched through an automated search on the Google Scholar search engine. Xiao and Watson [51] state that during this phase, only the title of candidate papers are considered. The search terms provided as input on Google Scholar are retrieved from the research questions. Ultimately, this led to the following list of search terms that will be applied:

(Measuring OR Applying) AND (ISO 9126 OR ISO 25010) AND (Characteristics OR Definitions)

In the second step, a list of inclusion and exclusion criteria will be used to create a subset of the papers found when using the previously mentioned search terms. Xiao and Watson [51] conclude that during the second step, both the title and abstract of the candidate papers are assessed. Inclusion criteria define the characteristics of subjects in the study, where as exclusion criteria describe attributes that prevent a subject from being included in the study [8]. Patino & Ferreira [36] conclude that inclusion and exclusion criteria are unique as each

literature review has its own purpose, but criteria usually concern either the study population, the nature of the intervention, the outcome variables, the time period, the cultural and linguistic range or the methodological quality. The inclusion criteria that will be used for this study are as follows:

1. Research papers that further define the characteristics of the ISO 9126 or ISO 25010 standard are included, as this topic contributes towards the goal of answering SQ1.

2. Research papers that apply ISO 9126 or ISO 25010 in a corporate setting, as these studies can help identify potential quantitative data values that are feasible to measure within an organisation.

3. Research papers that aim to quantify ISO 9126 or ISO 25010, as the methods and techniques used in these studies can be valuable for this research.

The following list describes the selected exclusion criteria, also including a rationale for exclusion:

1. Research papers not written in English or Dutch will be excluded, due to our limited proficiency in other languages.

2. Research papers that discuss software product quality of software products developed for academia will be excluded, as the scope of our study is to describe software product quality of software products developed in a corporate environment.

3. Research papers that focus on extending the ISO 9126 or ISO 25010 software product quality model for a specific industry will be excluded, as the goal of our study is to research whether it is feasible to create a quantitative model that can be applied to all industries.

During the third step, the quality of the papers will be assessed. Quality assessment of papers is often done using a checklist. Different contents of the checklist are usually required for qualitative and quantitative research [51]. As research in the field includes both qualitative and quantitative studies, a separate checklist will be used dependent on the research method. Kmet et al. [28] designed two quality assessment checklists (for both qualitative and quantitative studies) meant to be used in conjunction. These checklists have been chosen because they have been well-cited, and explicitly support performing quality assessment for both qualitative and quantitative research. When using these checklist, the quality score is calculated by summing up the points scored by the paper for each checklist item, and dividing by the total number of points available. Papers that scored under a certain threshold were excluded. Kmet et al. [28] suggest that the chosen threshold can be dependent on the time and budget constraints of the research, and can range from 0.55 (liberal) to 0.75 (conservative). As the goal of the literature study is to further explore what has already been studied in the domain of software product quality, the liberal score of 0.55 has been

chosen. The checklists for assessing the quality of qualitative and quantitative studies have been shown in Table 2.1 and Table 2.2 respectively.

Criteria	**YES (2)**	**PARTIAL (1)**	**NO (0)**
1. Question / objective sufficiently described? 2. Study design evident and appropriate? 3. Context for the study clear? 4. Connection to a theoretical framework / wider body of knowledge? 5. Sampling strategy described, relevant and justified? 6. Data collection methods clearly described and systematic? 7. Data analysis clearly described and systematic? 8. Use of verification procedure(s) to establish credibility? 9. Conclusions supported by the results? 10. Reflexivity of the account			

Table 2.1: Quality assessment for qualitative studies [28]

Criteria	YES (2)	PARTIAL (1)	NO (0)	N/A
1. Question / objective sufficiently described? 2. Study design evident and appropriate? 3. Method of subject/comparison group selection or source of information/input variables described and appropriate? 4. Subject (and comparison group, if applicable) characteristics sufficiently described? 5. If interventional and random allocation was possible, was it described? 6. If interventional and blinding of investigators was possible, was it reported? 7. If interventional and blinding of subjects was possible, was it reported? 8. Outcome and (if applicable) exposure measure(s) well defined and robust to measurement / misclassification bias? Means of assessment reported? 9. Sample size appropriate? 10. Analytic methods described/justified and appropriate? 11. Some estimate of variance is reported for the main results? 12. Controlled for confounding? 13. Results reported in sufficient detail? 14. Conclusions supported by the results				

Table 2.2: Quality assessment for quantitative studies [28]

The fourth step of the literature review is to extract the data from the selected papers. For each paper, its author, topic, research method, results and conclusions will be noted. These findings are used as input for the next step.

During the final step, the data will be analysed and synthesized. During this step, the data will organised into a textual description, accompanied by figures and tables where necessary. The output of this step is shown in Chapter 3.

Focus Groups

To develop the discussed artefact (quantitative data values for the ISO 25010 standard) and answer SQ2, a series of focus groups will be conducted. A focus group is a technique used in research in which a group of individuals selected by a researcher discuss a topic chosen by the researcher based on personal experience [16]. The difference between a focus group

and a group interview is that a focus group relies on the interaction between participants, where as a group interview emphasises the questions and responses between participants and researcher [16].

The participants will be asked to find quantitative data values for each of the considered quality characteristics. Before participants are allowed to discuss, a definition of the quality characteristic is given. To take into account that participants will have limited time to participate in the focus groups, the focus groups will be time-boxed to one and a half hour. During the session, participants will be instructed to write down the identified quantitative data values on a set of post-its. After finishing this activity, a list of quantitative data values will be created.

2.2.4 Demonstration

To further demonstrate the feasibility of measuring the found metrics in a practical setting, the results will be presented to domain experts. For each identified metric, the domain experts will be shown the identified the data value. Afterwards, the following two questions will be asked:

1. What source system(s) can be used as input for the data value?
2. How difficult is it to measure the data value from its source?

Through this demonstration, a light will be shed on any issues that might occur when trying to measure the data value. The output of this activity will be an enriched list of data values, where a source is provided for each identified data value. Additionally, each data value will be accompanied by an indication of how difficult it would be to measure the data value from its source.

2.2.5 Evaluation

The main goal of the evaluation activity is to observe how well the artefact serves as a solution to a problem [37]. To achieve this goal and further answer SQ3, an attempt will be made to measure a subset of the identified quantitative data values in an actual usage scenario. This will be done by looking at a single case of a deployed software product. The methods used to obtain the metrics from the case depend on the evaluated source during the demonstration phase. Due to time constraints, we will prioritize measuring only those quantitative data values that domain experts classified as readily measurable.

To further illustrate what has been discussed in the past three subsections, an overview of the DSRM stages design & development, demonstration and evaluation, their corresponding activities, and the research questions they address are shown in Figure 2.2.

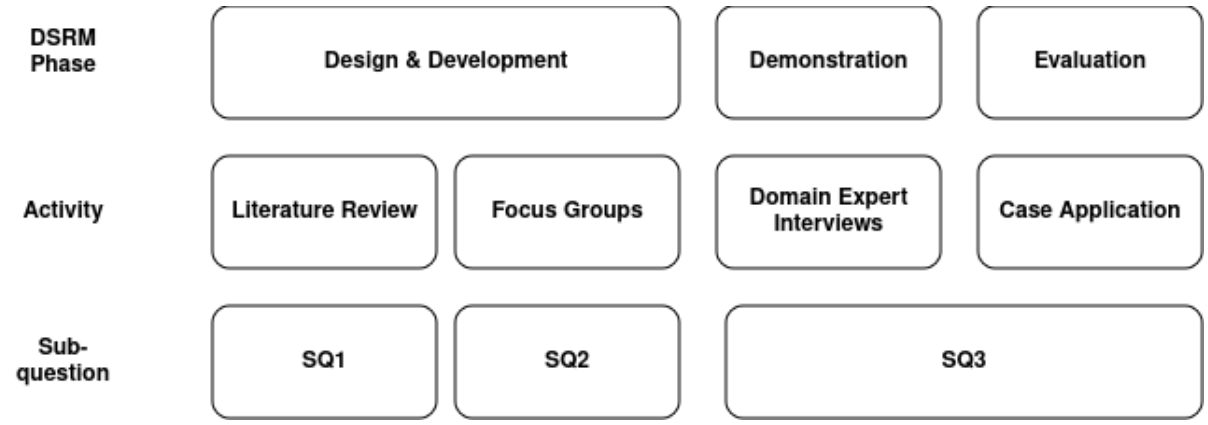

Figure 2.2: DSRM application overview

2.2.6 Communication

The final activity in the DSRM process model is communication. During this activity, the identified problem and the importance and effectiveness of the proposed artefact will be displayed to relevant audiences [37]. In this study, relevant audiences will be other researchers in the field of information technology, but also organisations that cooperated in this study and benefit from the created artefact. Communication will be done by documenting research findings in a thesis. Additionally, a presentation will be given to shed a light on the key findings of this study.

Chapter 3

Literature Review

As discussed in Chapter 2, a systematic literature review will be performed as part of the design and development activity in the DSRM process. The literature review results consist of three parts. First, SQ1 will be answered by further elaborating on the eight characteristics of the ISO 25010 Software Product Quality Model. Additionally, the definitions found in this part will be used as input for the focus groups. Second, existing attempts at applying and measuring ISO 25010 in a practical setting will be discussed. The results from this section will be used to further explore what has already been studied on the topic of applying and measuring the ISO 25010 software product quality model, and to apply their leanings to our research. Third, a brief conclusion will be given to draw upon the key takeaways from the systematic literature review. After using the search criteria listed in Chapter 2 and applying the inclusion and exclusion criteria, a total of 18 papers was evaluated. After assessing the quality of the papers, 14 papers were found to be eligible for inclusion. The performed quality assessment of both included and excluded papers can be found in the Github repository of the research project [1] Additionally, the official ISO 25010 standard [23] has also been referenced to further define the software product quality model.

3.1 Defining ISO 25010

As an international standard for providing guidelines to assess software product quality, ISO 25010 has been widely researched. The ISO 25010 standard consists of two quality models, each serving their own purpose. First, the quality-in use model is designed to measure the quality of software from a users perspective. Second, the software product quality model provides characteristics related to static properties of software and dynamic properties of the software product [12]. As the scope of our study is to assess software product quality by looking at quantitative data values organisations can capture, only the software product quality model will be regarded, meaning that further assessment of the quality-in-use model will be out of scope for this study. The ISO 25010 software product quality model is a three-

[1] `https://github.com/KourosNL/ThesisMBI/tree/master/QualityAssessements`.

level hierarchical model that divides software product quality (level one) into eight software product quality characteristics (level two). Each software product quality characteristic is further decomposed into multiple sub-characteristics. The software product quality model has been previously visualised in Figure 1.2. In total, the standard has 31 sub-characteristics [19]. To facilitate the focus groups and assist domain experts in finding quantitative data values, it is vital to gain a more thorough understanding of each characteristic. Thus, the following sub-sections will further discuss each of the eight quality characteristics. To limit the scope of this literature review, only the level two quality characteristics will be defined. A table of sub-characteristics and their definitions can be found in Appendix A.

3.1.1 Functional Suitability

In the ISO 25010 standard, functional suitability is the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions [23]. Typically, the functional suitability characteristic is tested by comparing requirements against included functionality [13]. In the preceding ISO 9126 standard, functional suitability was one of the five sub-characteristics of the functionality quality characteristic [5].

3.1.2 Performance Efficiency

ISO/IEC [23] defines performance efficiency as the "performance relative to the amount of resources used under stated conditions". To further elaborate on this definition, a software system that scores well in the performance efficiency dimension has a response time that fits the system requirements, as well as not taking up more resources than stated in the requirements.

3.1.3 Compatibility

Compatibility is the degree to which a product, system or component can exchange information with other products, systems or components [23]. Fahmy et al. [13] suggest that compatibility is usually tested by checking whether the system is compatible with the available browsers, networks, hardware, mobile devices, and operating systems.

3.1.4 Usability

ISO/IEC [23] lists usability as a quality characteristic defined by "the degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use". From this definition, it is still unclear what the specified users, goals, context and system are. Speicher [45] suggests that before usability is evaluated, it is necessary to provide specific information on these elements.

3.1.5 Reliability

Reliability is the degree to which a software product performs specified functions under specified conditions for a specified period of time [23]. When testing reliability, a software system will usually be exposed to unexceptionally high volumes and frequency of requests, with the purpose of checking whether functionality still works under unusual conditions.

3.1.6 Security

In the ISO 25010 software product quality model, security is 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 authorization" [23]. Saptarini et al. [44] suggest that to better measure security, security compliance could be added as a sub-characteristic. However, the study by Saptarini et al. [44] only applied the characteristic to academic information systems, not making implications about the generalisability of this claim.

3.1.7 Maintainability

According to Peters and Aggrey [38], maintainability is the ability of a software product to be modified, corrected, or adapted based on changes in the environment. This definition is similar to the intended definition provided by ISO/IEC [23], where maintainability is the "degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers". For this study, the definition by Peters and Aggrey [38] will be used, as in the definition by ISO/IEC [23], it is not further specified who the intended maintainers are. Additionally, Peters and Aggrey [38] provide further elaboration to the definition by stating that modifications, corrections are adaptations are done due to changes in the environment.

3.1.8 Portability

ISO/IEC [23] give the following definition for 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". As both portability and compatibility are concerned with the deployment of software systems to different varying environments, it is important to note their differences. Portability is about making sure that software can be moved between environments, while compatibility concerns the ability of a software system to operate within a given environment (and not about the transfer between environments).

3.2 Applying and Measuring ISO 25010

Applying the ISO 25010 software product quality model (and the preceding ISO 9126 software product quality model) in a practical setting has been a widely researched topic. Fahmy et al. [13] applied the software product quality model to the domain of e-book applications and concluded that for this domain, the characteristics functional suitability, reliability, usability, and performance efficiency were of higher importance. In this study, characteristics were assessed by conducting a questionnaire asking participants whether the characteristic was implemented 'Poor', 'Fair', 'Good' or 'Excellent' in the software product. The paper concluded by claiming that their results could help in identifying areas in which e-book applications could use improvement. The claim that the importance of the individual quality characteristics differs between industries is supported by Kadi et al. [26]. In their study on applying ISO 25010 to cardiac decision support systems by looking at system requirements, results indicated that for this particular domain, quality characteristics functional suitability, reliability and performance efficiency were most important. A study with a similar scope was conducted by Izzatillah [24]. In this study, a survey was conducted to prioritise the importance of the sub-characteristics for an application within the transport services domain. Study results indicated that functional appropriateness, non repudiation, and appropriateness recognizability were the three sub-characteristics most prioritised for the given application. Izzatillah [24] also studied whether the importance of sub-characteristics varied between Android and IOS users, but no significant differences were found.

A different study applied ISO 25010 to the domain of cloud computing. The purpose of this study was to evaluate whether ISO 25010 can be used to measure the quality of cloud computing applications. Due to time constraints, the study only aimed to measure the time behavior sub-characteristic that is part of the performance efficiency characteristic. Study results indicated that in a typical cloud application, it is feasible to collect the required data for measuring time behavior. However, the researchers also conclude that data collection is a complex issue, as it is a time-intensive task and collecting data on system infrastructure may affect the performance of the infrastructure itself [42]. A related study that also limited its scope to applying only a single quality characteristic in a practical setting was conducted by applying the functional suitability quality characteristics to ERP systems. In this study, the ERP system under discussion was divided into functional modules. To determine the functional suitability of a module, the required functionalities elicited from the business were listed. The researchers subsequently tested features on whether they were present in the module, and if the result from using the feature was the same as the expected result [35].

While the previous studies applied ISO 25010 to assess the quality of a software product, Haoues et al. [19] applied the quality characteristics in the software product quality model to select software architecture. In this research, the five most commonly used architectures in the industry were analysed. After that, the study highlighted which quality characteristic is used the most by each architecture. Finally, Haoues et al. [19] conclude that it is important to study the relationships between quality characteristics to maintain software at a high quality. Figure 3.1 displays the found relationships between quality characteristics. Where a

positive relationship between quality characteristics is coded as a '+', negative relationships are coded as a '-', and neutral relationships are coded as a '0'. Finally, relationships that can be both positive and negative, depending on quality characteristic usage, have been coded as '±'. Additionally, the authors conclude that a quality characteristic has a positive relationship with itself, as the implementation of one of its sub-characteristics positively affects the other sub-characteristics. The relationships between quality characteristics have been further explored in a dedicated study on the mutual influences of software quality characteristics [21]. Where the study by Haoues et al. [19] uses a survey based on literature to identify relationships, Hovorushchenko and Pomorova [21] opted for a different research method. In this research, the previously introduced ISO 25023 standard was used to identify measures. As the measures in this standard can attribute to more than one sub-characteristic, dependencies were found between sub-characteristics that use a same measure.

	Functional suitability	Performance efficiency	Usability	Compatibility	Reliability	Security	Maintainability	Portability
Functional suitability	+	−	+	0	+	−	+	0
Performance efficiency	0	+	−	−	0	−	−	−
Usability	+	±	+	0	+	0	±	0
Compatibility	0	0	0	+	0	−	±	+
Reliability	+	0	+	0	+	0	+	0
Security	0	−	−	−	+	+	0	0
Maintainability	+	−	0	+	±	±	+	+
Portability	0	−	0	+	0	0	+	+

Figure 3.1: Relationships between quality characteristics [19]

Besides the studies applying ISO 25010 in a practical setting, applying the preceding ISO 9126 standard to a software system in use has also been the topic of a few research papers. As ISO 25010 is an extension and replacement of ISO 9126, it is still useful to evaluate papers that applied ISO 9126. Djouab and Bari [9] applied ISO 9126 to e-learning systems. In their study, Djouab and Bari [9] aimed to map the key characteristics of e-learning systems to ISO 25010. Ultimately, the conclusion was made that to measure software product quality in the e-learning domain, additional quality characteristics were needed. A broader attempt to apply the standard in a practical setting was done in mobile environments. In this study, the researchers used ISO 9126 to evaluate how limitations of mobile applications (frequent disconnection, limited storage capacity) affected software product quality. The conclusion was drawn that mobile limitations will impact some ISO 9126 characteristics, making the evaluation of the software product quality of mobile applications a complex task [22]. The field of measuring ISO 9126 was further explored by Behkamal et al. [5] and was previously introduced in Chapter 1. To briefly repeat and elaborate on what has already been discussed, this study consisted of three parts. First, the ISO 9126 model was customised

for the B2B domain. Second, analytical hierarchy process, a prioritisation technique, was applied to prioritise the characteristics in ISO 9126. Third, the model was evaluated in a case by carrying out a checklist derived from literature that asked participants whether a sub-characteristic was present in the software product under discussion. The researchers concluded that a checklist is insufficient for assessing the final quality of the software and more research into finding quantitative criteria is required [5].

A study that uses a similar research method as Behkamal et al. [5] to measure the quality of a software product was carried out on a software system in health care [39]. In this study, analytical hierarchy process was used to calculate the importance of the quality characteristics of ISO 25010 for the given application. A combination of objective and subjective metrics were used to measure the sub-characteristics. To test the functional suitability of the software product, black-box testing was applied. Black-box testing is a testing method in which the actual output is compared to the expected output, also used to measure functional suitability in the previously discussed study by Panduwiyasa et al. [35]. Additionally, the reliability of the system was assessed using a technique called stress testing. During a stress test, a software system is put under stress by sending large amounts of data with very high frequency and volume to the system. Subsequently, it is measured how the system functions under heavy load conditions. Finally, subjective data values were introduced by distributing a questionnaire to measure the usability of the system.

3.3 Summary

This chapter further explored related work on the ISO 25010 software product quality model. The definitions listed in the first subsection will be used in the focus groups, with the purpose of gaining a common understanding on the quality characteristics. In the second subsection, a consolidation of studies on the application and measurement of ISO 25010 was given. A few conclusions on ISO 25010 can be drawn from this section. First, multiple studies suggest that the importance and priority of the quality characteristics is dependent on the domain for which the application is developed [13, 24, 26]. Second, quality characteristics are dependent on each other, and researching these relationships is important for further improving the quality of software products [19, 21]. Finally, multiple techniques, such as checklists, can be used to measure the sub-characteristics of the software product quality models [5]. Examples of techniques that measure objective values are black-box testing and stress testing, where as questionnaires can be used as a technique to elicit subjective data values [35, 39].

Chapter 4

Results

In this chapter, the results of the conducted focus groups and expert interviews will be discussed. Moreover, the found quantitative data values have been applied in a case, which will also be discussed in this chapter.

4.1 Focus Groups

As discussed in Section 2.23, the goal of the focus groups was to gain understanding into the possible quantitative values that can be used to measure the eight ISO 25010 defined quality characteristics. As each focus group was conducted in a different context (different industry domains, different levels of experience, different technology stack, different roles within the domain of software development), the following subsections will also elaborate on the environment in which the focus group was conducted. The output of each focus group (a list of quantitative data values per quality characteristic), can be found in Appendix B. Additionally, the created transcripts can be found on the Github repository for the research project [1].

4.1.1 Focus Group 1 (FG1)

The first focus group was conducted with three members of a data engineering platform team in an organisation in the logistic sector, from now on referred to as LogOrg. The focus group comprised two data engineers and one solution consultant, each with five, three, and twenty years of experience, respectively, within the software industry. In total, the participants came up with 32 quantitative data values. The focus group revealed a few insights that require elaboration. The members of the focus group faced difficulties finding quantitative measures for the portability quality characteristic and came to the conclusion that measuring portability is not always wished for. One participant indicated: "This assumes that this is a good quality to have, to make it portable. Well, a lot of companies, including our company,

[1] `https://github.com/KourosNL/ThesisMBI/tree/master/Transcripts/FocusGroups`.

made a very specific choice to make it a very low priority.". The participants also faced difficulties when trying to find objective measures for the security quality characteristic. When the number of data breaches was proposed as a quantitative measure for security, one of the participants made the remark that some quantitative data values for security (such as the number of data breaches) can only be measured after an incident has happened, making security a difficult quality characteristic to measure beforehand. On the other hand, the participants agreed that the reliability quality characteristic was the easiest quality characteristic to be demonstrated through quantitative data values, with one participant stating: "I think this is very measurable ... I think this is by far the most measurable thing.". Moreover, the participants expressed to have difficulties finding baselines for the found metrics. To exemplify, in the discussion on reliability, the participants agreed that using system up time was a good metric. However, the participants faced trouble on finding out what should be the standard to compare the up time with. Ultimately, the participants came to the conclusion that up time should be compared to the service level agreement for up time. In this discussion, all participants agreed that by comparing up time against the service level agreement, some sort of subjectivity was added as the metric is dependent on how the service level agreement is defined. Finally, the focus group revealed that a single quantitative data value does not necessarily represent the quality characteristic as a whole. When discussing the number of people working on a product as a potential measure for maintainability, the comment was made that a high number of people working on a product does not necessarily mean that the product is not maintainable.

4.1.2 Focus Group 2 (FG2)

The second focus group was conducted with three members of a consultancy organisation working together on a project with the goal of building a data platform for an organisation in the logistic sector. The focus group consisted of a data engineer, a full-stack developer, and a back-end developer, each with 1, 6, and 7 years of experience, respectively, within the software industry. During this focus group, the members of the focus group noted 23 quantitative data values. During the discussion, the participants made a remark similar to what was discussed during the first focus group, concluding that the portability quality characteristic is not always important for a software product. Besides that, the discussion uncovered new insights on the quality characteristics. During the discussion, an interesting note on portability was raised. While not being included as a separate measure, the members of the focus group proposed the idea that the measures of the other seven quality characteristics should still hold when porting from one usage environment to another, thus being objective representations for portability as well. When posed the question on whether a measure for portability was related to the other quality characteristics, a participant answered: "It still has to be reliable, you still need to be able to work with it, it still needs to perform, it still needs to do what it used to do.". The previous note that a better representation of a quality characteristic is given by adding more quantitative data values was once again exemplified by the members of this focus group. The participants came to the consensus that measuring

just the up time of a product does not provide a complete view on reliability, but measuring the up time per functionality could also be of importance: "Discord may be up 24 hours a day, but if a hundred people still lose their conversations every day because they drop out, you're still not a hundred percent reliable, despite your up time being so.". A different topic that the participants briefly touched upon was the topic of interpretation. The participants came to the consensus that the desired value of some of the quantitative data values was dependent on the type of product. This was further illustrated through the quantitative data value 'usage time' for the usability quality characteristic. It was argued that some software products, such as games, have better quality if users spend more time using the product. Other products, such as web shops, may be more usable if users spend less time using the product.

4.1.3 Focus Group 3 (FG3)

The third focus group consisted of four participants. All participants were students at Utrecht University with professional experience working in the software industry besides theirs studies. The participants were two data engineers, a data analyst, and a machine learning operations engineer, each with 1.5, 1, 1.5 and 1 year of experience working in the software industry. The members of the focus group proposed 17 quantitative data values. During the focus group, an interesting discussion that requires elaboration took place. The members of this focus group found it difficult to define a baseline for a measure, indicating uncertainty on what to compare the found quantitative data value to. As a result, they proposed the idea of adding interpretation to the values by measuring the value over time for a given product. Finally, they also concluded that it is difficult to compare quality characteristics between software products. As a result, the measured data values can only be interpreted when comparing the product with itself in different iterations of the product.

4.1.4 Focus Group 4 (FG4)

The fourth focus group consisted of a lead engineer, a DevOps engineer, and one front-end engineer working in a team implementing event driven software products within LogOrg, with 18, 6.5, and 6 years of experience working in the software industry respectively. In these event driven software products, large quantities of business events are received each day. After processing the events and applying business logic, they are represented in the front-end of the application. In total, the members of the focus group introduced 41 quantitative data values. During the focus group, the members had an interesting discussion on the definition of functional suitability, delving more deeply into the the stated and implied needs aspect. One participant argued that compliance to organisation rules should be a measure of functional suitability as an implemented security requirement satisfies stated or implied needs. However, a different participant argued that the product would still function without the implemented security requirement, and that it should therefore not be a part of functional suitability. In the end, the participants did not agree consensus on this definition. A different

discussion on the definition of the quality characteristics was made when starting to discuss compatibility. One of the participants made the critical remark that the industry usually describes compatibility as the degree to which the system can be run on multiple platforms, as opposed to the ISO 25010 definition, where compatibility is the degree to which the product can exchange information with other products. Another interesting note on measure interpretation was made during this focus group. The participants support the argument made by the members of the third focus group, and also indicated that measures should be interpreted by comparing the product with itself over different iterations. Measuring an individual data value, or measuring the value over different products would not add any meaning to the measurement. Furthermore, the participants further illustrated that for some quantitative measures, it might be of value to measure them over a time frame, so no additional penalty is provided for having a malfunctioning product in previous iterations: "Also interesting, because let's say a system is not secure, I've experienced a data breach, and I've heavily invested in improvement, yet I still experienced a data breach in the past. Does that say something about the security of my application now?". Finally, the participants agreed with the previously made remarks, indicating that bad measurements for a certain quality characteristic do not necessarily result in a bad product: "If you would measure all metrics and your score for compatibility is zero. That is not necessarily a bad thing if it is not a requirement.".

4.1.5 Focus Group 5 (FG5)

The fifth focus group was performed with four data solution consultants, working on different data platforms for their clients. The focus group members were a solution consultant and two data engineers, each with twelve, five and six years of experience, respectively, within the software industry. The participants introduced 41 different quantitative data values. The previously made observation that measuring a single does not necessarily represent the quality characteristic as a whole was once again confirmed during this focus group. While the participants agreed that testing could be used to measure functional suitability, a high success rate of functional tests does not necessarily result in a product that is functional suitable for it's purpose, with one participant commenting: "Testing indeed, but is that the same as appropriateness? Something can be perfectly tested but not matching its purpose. Think of building a bike and getting squared wheels, or six wheels.". Another interesting notion was made on maintainability. Furthermore, one participant reasoned that maintainability could potentially be the inverse of compatibility. When a product does not support many exchange formats, less knowledge is required to maintain the product. Besides introducing new ideas, the members of the focus group also confirmed a notion made by other focus groups. The participants of the focus group agreed that the desired value of some of the quantitative data values was dependent on the type of product, which was also mentioned in FG4. Finally, one of the members of the focus group also made a remark on the difficulty of thinking of objective measures for the quality characteristics, and concluded that usability was the easiest quality characteristic to find quantitative data values for. The participant

reasoned that this might be attributed to the fact that usability has many relevant aspects, so separate data values can be used for measuring the efficiency, effectiveness, and satisfaction of the software product.

4.1.6 Focus Group 6 (FG6)

The final focus group was conducted with three data solution consultants working together on a project with the goal of building a data platform for LogOrg. The focus group members were two data engineers, and a machine learning engineer, each with 8, 3 and 2.5 years of experience, respectively, within the software industry. In total, the participants wrote down 36 quantitative data values. During this group discussion, the participants shared new ideas on the topic of selecting a fitting baseline. When discussing performance efficiency, one participant noted: "You are writing a query and do not know that it can be faster until you know that it can be faster ... So that is what I have in my mind. You only know it once you have found the optimisation.". This citation shows that it is difficult to select a baseline, as the optimal value for the metric is often unknown. This fits the previously introduced idea that for some quantitative data values, meaning is added by measuring the same value multiple times over different iterations of the product. In the same context of measuring the performance of a query, one of the participants posed that subjectivity can be minimised by enriching a subjective measure with objective information: "You are performing the measurement based on your own experience. That includes subjectivity ... You can strengthen subjectivity with objective information ... This query has to take this long because we have this number of joins and each joins should take this long.". Finally, during their discussion on functional suitability, the members of the focus group also highlighted the importance of data quality. While they came to a consensus that the number of complaints sent to customer support could be valuable for functional suitability, the observation was made that the number of complaints sent to customer support does not always represent the actual number of complaints: "In that case you should be able to measure complaints. Of course, there will be a person who has a complaint, but does not file it.".

4.1.7 Creating a model

The results of the focus groups have been summarised in eight models (one per quality characteristic). When creating this model, a few considerations have been made:

- Duplicate quantitative data values were found between focus groups. As a result, the model will contain a column indicating the **Evidence** for the found value. This number indicates the number of focus groups proposing the measure (FG1, FG2, FG3, FG4, FG5, FG6).
- Participants made the argument that some quantitative data values are specific to a certain type of application or infrastructure. As a result, the model will contain a column named **Context**. This column will contain information on the context in

which the data value was mentioned. If the context was not explicitly stated by the participants, this column will be left empty. If none of the quantitative data values for a given quality characteristic have a value for this column, the column will be emitted from the model (FG4).

- When looking at the list of identified data values, it becomes apparent that some metrics can only be measured over a period of time as they provide insights into past performance or outcomes. An example of this would be the number of data breaches for the security quality characteristic, as this quantitative data value is always measured over a particular time interval. On the other hand, some metrics can be measured or evaluated at the current point in time, providing immediate feedback on the state of the software product, such as the lines of code for the portability quality characteristic. Ultimately, this led to the decision of including a **Type** column in the model, which can have values 'over time' or 'point in time' (FG1, FG2, FG3, FG4, FG5, FG6).

- Some of the proposed metric rely on a subjective assessment. A record was considered to be subjective if the value of measurement was dependent on human evaluation. An example of a metric that falls under this category is the 'Net Promoter Score'. As the Net Promoter Score is measured by asking users how likely they are to recommend the software product to others, it is dependent on human evaluation. Records that rely on a subjective assessment have been flagged by adding an asterisk to the metric name (FG1, FG2, FG3, FG4, FG5, FG6).

Furthermore, a few proposed metrics have been excluded from the model by applying a single exclusion criterion. Metrics that do not describe a measurable value were excluded: During the focus groups, the participants wrote down their ideas on post-its. Some metrics introduced abstract concepts that could not be captured in a single quantitative data value. If the meaning and measurement of the concept could not be directly derived from the transcript, the decision was made to drop the metric from the model to prevent misinterpretation. An example of a metric that was dropped following this exclusion criteria is the measure 'Maturity of tests' for the maintainability quality characteristic. As it is unclear what the maturity part of this metric refers to, and it does not result in a single measurable value, the decision was made to exclude the metric from the maintainability model.

After applying the exclusion criterion, 31 values were excluded from the models. The metrics dropped based on this exclusion criterion are found in Table 4.1.

Quality Characteristic	**Metric**
Functional Suitability	Prod test
	Load Test / target Load
	User acceptance test
	Number of search terms in search functionality
	Click behaviour
Performance efficiency	Asking users if it fits performance needs with a large sample Stress test
	Logging
Reliability	Logging per function
	Number of differences between software instantiations
Usability	Text to speech: how long to get to relevant records
	A/B testing
	Time of day
	User analysis: fits software implementation
	Type of users
Portability	Number of skills needed to change environments
Compatibility	How actual is used technique
	Number of interfaces
	Types of fields
	Number of changes to data structure
	Version of the software / number of use cases
Security	Security controls of common chosen framework implemented
	Percentage that fits security requirements
	Software up to date
Maintainability	Maturity of CI/CD
	Number of configuration parameters required to change
	Total language popularity
	Time spent between OPS and DEV
	Used programming paradigm

Table 4.1: Metrics excluded by applying the first exclusion criteria

After the exclusion criterion was applied, a total of 127 distinct metrics remained in the models. This included 16 metrics for functional suitability, 17 metrics for performance efficiency, 18 metrics for reliability, 12 metrics for usability, 15 metrics for portability, 13 metrics for compatibility, 19 metrics for security, and 17 metrics for maintainability. The following tables show the created models for each quality characteristic. The tables have been sorted on the evidence column in descending order, thus showing the most frequently named metrics at the top of the table. The applied definitions for the identified quanitative data values can be found in the Github repository dedicated to the research project [2].

Metric	**Evidence**	**Type**
Number of delivered functions / number of required functions	2	Point in time
Percentage of succeeding tests	2	Point in time
Net Promoter Score*	2	Point in time
Number of changes needed to fit to original requirements	1	Point in time
Market share	1	Point in time
Number of unsolvable issues sent to customer support	1	Over time
Number of delivered functions / number of functions that contribute to main goal	1	Point in time
Number of clicks before action is finished	1	Point in time
Test coverage	1	Point in time
Percentage of pixels equal to UI design	1	Point in time
Large language model agrees that implementation fits goal	1	Point in time
Team uses sprint review	1	Point in time
Implementation fits acceptance criteria	1	Point in time
Number of issues	1	Over time
Number of users / number of issues sent to customer support	1	Over time
Uptime	1	Over time

Table 4.2: Functional Suitability Metrics

[2] `https://github.com/KourosNL/ThesisMBI/tree/master/Definitions`.

Metric	Evidence	Type	Context
Memory utilisation	4	Over time	
CPU utilisation	4	Over time	
Network utilisation	3	Over time	
Time to interactive	2	Point in time	Front-end
Latency per user	1	Over time	
Latency per increasing user	1	Over time	
Latency difference per location	1		
Resource costs / expected resource costs*	1	Over time	
Used current / expected used current*	1	Over time	
Cyclomatic complexity	1	Point in time	
Mobile app battery usage	1	Over time	Mobile app
Cost per retry	1	Over time	Event driven architecture
Number of retries	1	Over time	Event driven architecture
Percentage of dead code	1	Point in time	
Number of connections between source system and other systems	1	Point in time	
Execution time per function	1	Point in time	
Execution time per function before refactoring / execution time per function after refactoring	1	Point in time	

Table 4.3: Performance Efficiency Metrics

Metric	Evidence	Type
Uptime / service level agreement uptime	2	Over time
Uptime	2	Over time
Mean time between failures	1	Point in time
Amount of data lost on failure	1	Point in time
Duration of 1% longest downtime	1	Point in time
Distribution downtime / max downtime	1	Point in time
Uptime per functionality / service level agreement uptime per functionality	1	Over time
Number of reliability functionalities implemented / number of reliability functionalities required	1	Point in time
Number of failed events / number of successful events	1	Over time
Number of requests / service level agreement number of requests	1	Over time
Number of functions that always give the same result on the same input / number of functions	1	Point in time
Number of backup options	1	Point in time
Number of users	1	Over time
Number of functions working under stress test / number of functions	1	Point in time
Accuracy of decimal values	1	Point in time
Latency per user	1	Over time
Mean response time to incidents	1	Over time
Mean recovery time to incidents	1	Over time

Table 4.4: Reliability Metrics

Metric	**Evidence**	**Type**	**Context**
Number of clicks to complete task	3	Point in time	
Time to complete a task	3	Point in time	
Product complies with accessibility standards	2	Point in time	
Conversion rate	2	Point in time	
Time of usage	2	Over time	
Number of recurrent users	2	Over time	
Number of social media tags	1	Over time	
Net Promoter Score*	1	Point in time	
Number of errors in front-end	1	Point in time	Front end
Number of users using the application each day	1	Over time	
Number of times user expresses positive emotion (facial recognition)	1	Point in time	
Difference in task time between fastest user and slowest user	1	Point in time	

Table 4.5: Usability Metrics

Metric	Evidence	Type	Context
Number of major platforms you can run on without code modifications	2	Point in time	
Time to port to different platform	2	Point in time	
Number of features functional on platform / number of features still working for new platform	2	Point in time	
Number of actions needed to port to different platform	2	Point in time	
Number of supported clouds	2	Point in time	
Platform specific lines of code / lines of code	1	Point in time	
Number of cloud agnostic services used	1	Point in time	
Number of supported devices / Number of potential users	1	Point in time	
Number of supported browser versions	1	Point in time	Web application
Number of supported mobile platforms	1	Point in time	Mobile application
Number of supported OS versions	1	Point in time	
Number of supported real time versions	1	Point in time	
Lines of code	1	Point in time	
Number of functions covered by documentation / number of functions	1	Point in time	
Number of code changes needed per function to port	1	Point in time	

Table 4.6: Portability Metrics

Metric	Evidence	Type
Number of data exchange formats used accepted within industry	3	Point in time
Number of connected systems	2	Point in time
Percentage of API endpoints compliant to organization standards	1	Point in time
Product has standardized API documentation	1	Point in time
Number of functions exposed by API / Number of functions available for user	1	Point in time
Number of connected systems / Goal of number of connected systems	1	Point in time
Number of successful requests / number of requests	1	Over time
Number of protocols used for sharing information	1	Point in time
Number of supported operating systems	1	Point in time
Number of packages / Number of packages that can be shared with different techniques	1	Point in time
Systems are connected through standardized authentication	1	Point in time
Number of data formats not accepted within industry	1	Point in time
Number of API functions / number of documented API functions	1	Point in time

Table 4.7: Compatibility Metrics

Metric	Evidence	Type
Number of security protocols used	2	Point in time
Number of persons for which data is lost	1	Over time
Percentage of data encrypted	1	Point in time
Product complies to GDPR	1	Point in time
Security breaches / Goal*	1	Over time
Number of users with access to production / Number of users with access	1	Point in time
Open security issues per severity code	1	Point in time
Open security issues for infrastructure	1	Point in time
Number of people with admin privilege	1	Point in time
Number of times break the glass procedure activated	1	Over time
Number of actions performed with elevated privileges	1	Over time
Number of lines of defense	1	Point in time
Number of data breaches per year / number of attacks	1	Over time
Number of points of failure	1	Point in time
Number of illegal users	1	Point in time
Product uses SSO for authentication	1	Point in time
Number of users not in a security group	1	Point in time
Number of security layers	1	Point in time
Product has internet access	1	Point in time

Table 4.8: Security Metrics

Maintainability	**Support**	**Type**
Lead time of new feature	2	Point in time
Number of developers with knowledge of techstack	2	Point in time
Number of documented functions / number of functions	2	Point in time
Cycle time of new feature	1	Point in time
Cycle time of bugfix	1	Point in time
Lead time of bugfix	1	Point in time
Code coverage	1	Point in time
Percentage of intended maintainers who can maintain	1	Point in time
Time to end of life for toolstack	1	Point in time
Cyclomatic complexity	1	Point in time
Time until new developer makes first deployment to production	1	Point in time
Percentage of dead code	1	Point in time
Number of code changes per timeframe	1	Over time
Number of functions / number of classes	1	Point in time
Number of bad code smells	1	Point in time
Number of domain knowledge holders	1	Point in time
Years of experience maintainers	1	Point in time

Table 4.9: Maintainability Metrics

4.2 Domain expert interviews

The purpose of the interviews with domain experts was to gather more information on the quantitative data values proposed during the focus groups. The interview transcripts are provided in the Github repository for the research project [3]. Specifically, We were interested in discovering the source of the data value (what system or tool can be used to collect the quantitative data value), and how feasible it would be to measure the value in a given context. To determine the feasibility of measuring the data value, the domain experts were presented two dimensions:

- **Difficulty of obtaining data**: Experts were asked to evaluate how difficult it was to obtain the data. An ordinal scale consisting of values low, moderate and high were used to assess how difficult it would be to extract the data from the source. A metric would be labeled as low if the data was mostly available, or minimal effort was required to make the data available. A metric would be labeled as moderate if the data could possibly be extracted from the source, but cleaning or prepossessing would be needed to make the data available. Finally, a metric would be labeled as high if the data was mostly unavailable from the source, and extensive effort would be required to make the data available.
- **Technical expertise required**: Experts were given the task to evaluate how much technical expertise would be required to extract the data from its source. Again, an ordinal scale consisting of values low, moderate and high were used. A metric was labeled as low if basic technical skills were required to set up the measurement (such as reading the data from a dashboard). A metric would be labeled as moderate if technical skills were required, but implementing the measurement could be done by most software developers. A metric was labeled high if advanced technical skills were needed to extract the data from its source.

4.2.1 Pilot interview

A pilot interview with a principal software engineer was conducted with the goal of validating the interview questions. During the pilot interview, the participant made two major remarks on the questions posed during the interview.

First, the participant indicated that before answering questions on the source and the feasibility of measuring the data, the context in which data was collected needed to be specified as the source and feasibility of collecting the data is dependent on the context in which the measurement is made. This was further illustrated by looking at the memory utilisation metric for the reliability quality characteristic. In an on-premise environment, memory utilisation is measured in a different way than in a cloud environment. As a result, the participant faced difficulties answering questions on the quality characteristics without

[3] `https://github.com/KourosNL/ThesisMBI/tree/master/Transcripts/DomainExpertInterviews`.

a predefined scope. Due to these difficulties, the decision was made to not use the results from the pilot interview for the evaluation of the identified quantitative data values.

Second, the participant suggested that it was difficult to assign values on the chosen ordinal scales for the feasibility dimensions as the descriptions of the categories were too vague, making it difficult to distinguish between values low, moderate and high.

Based on the insights from the pilot interview, the interview questions were adjusted. An attempt was made to minimise the difference in context between interviews to ensure that all quality characteristics were evaluated from a similar perspective. This was done by selecting one organisational context, and only select interviewees from two teams within this context. The selected context will be further elaborated on in the next subsection. Additionally, participants were explicitly asked to evaluate the quantitative data values using their experience from the environment they were currently operating in.

Besides that, the descriptions for the ordinal scales were improved for both the difficulty of obtaining data and technical expertise dimension. This refinement facilitated participants in classifying the quantitative data value into one of the labels low, moderate or high. Consequently, the following descriptions were used for the feasibility dimensions.

Difficulty of obtaining data:

- Low: Data is readily available, minimal effort required to make the data available (1 hour - 1 day)
- Moderate: Data is possibly available, but effort is required for preproccessing and cleaning (1 day - 1 week)
- High: Data is unavailable, extensive effort is required to make the data available (more than 1 week)

Technical expertise required:

- Low: Basic technical skills required for implementing the measurement
- Moderate: Technical skills or knowledge on a specific technology are required, but implementing the measurement can be done by most software developers.
- High: Advanced technical skills or in-depth knowledge on a specific technology are required implementing the measurement

4.2.2 Selected context

To select an appropriate scope for evaluating the discovered quantitative data values, all interviewees were data engineers or software engineers working for LogOrg. LogOrg is an international enterprise operating in the logistics sector. In 2023, the organisation employed over 30.000 people and had an annual revenue of over 3000 million euro. The organisation deploys their software products to the cloud, using AWS as a cloud provider. Both teams

use a combination of Python and Typescript to code their software products. In total, seven interviews were conducted. The first two interviews were conducted with members of a data engineering team at LogOrg. As an international enterprise, LogOrg produces large volumes of data. Within the data engineering team, the interviewees work on building software solutions that allow other teams within the organisation to consume the data according to their needs. The other five interviews were carried out with software engineers working for an enabling team in the e-commerce domain of the organisation. As an enabling team, the software engineers build internal software products to enable other software teams within the domain to develop their software products easier and faster.

4.2.3 Interview results

The figures below show the assessment of the domain experts on the metrics proposed during the focus groups. For each figure, the caption will clarify whether the assessment was performed with a member of the data engineering team or a member of the enabling team. The cells for the columns 'Difficulty of obtaining data' and 'Technical expertise required' were color coded. Green for all values that were assessed as 'Low', orange was used for all values evaluated as 'Moderate', and using red for all values assigned as 'High'. When a participant indicated that they did not know what label to assign, or the metric was not applicable to their context, the color grey was given to the cell. The next eight sections will illustrate the main outcomes of the interviews by showing the evaluation of the quantitative data values in a table. In addition, a brief textual explanation will be provided to elaborate on the results shown in the table. To deal with time constraints and allow the participants to elaborate on the context of their answers, each quality characteristic was only evaluated once.

Functional Suitability

The functional suitability quality characteristic was conducted with a member of the enabling team. In total, 10 quantitative data values were assessed as 'low' on the difficulty of obtaining data dimension, and five measures were evaluated as 'moderate' on this dimension. For the remaining metric (percentage of pixels equal to UI design), the difficulty of obtaining data was evaluated as N/A as the participant did not have any prior experience comparing a screenshot against a UI design and therefore expressed difficulty assessing this metric. Looking at the technical expertise required dimension, 8 quantitative data values were evaluated as 'low', 6 values were assessed as 'moderate', and 1 value was determined to be 'high'. For the metric determined to require a high level of technical expertise (number of changes needed to fit to original requirements), the participant expressed that setting up this measurement requires an aggregation of multiple sources. First, the number of changes needed should be retrieved from the code base. Second, the original requirements need to be retrieved from a project management tool. As combining these sources require extensive knowledge on how changes to the codebase are applied, the technical expertise required for this value was determined to be high. The participant defined a total of eight different sources for the fifteen evaluated metrics. A project management tool, such as Jira or Github projects, was

the most commonly named source. Additionally, the participant concluded that no source could be assigned to determine the market share, as the participant suggested that the source for this metric varies per situation. It usually involves doing some research on competitors, but the interviewee did not draw any conclusions on how this research could be done. An overview of the evaluated metrics is shown in Table 4.10.

Metric	**Source**	**DOOD**	**TER**
Number of delivered functions / number of required functions	Project management tool		
Percentage of succeeding tests	CI/CD tool		
Net Promoter Score	User review		
Number of changes needed to fit to original requirements	Project management tool, Version control repository		
Market share	N/A		
Number of unsolvable issues sent to customer support	Project management tool		
Number of delivered functions / number of functions that contribute to main goal	Project management tool		
Number of clicks before action is finished	Analytics Tool		
Test coverage	Code quality tool		
Percentage of pixels equal to UI design	N/A		
Large language model agrees that implementation fits goal	Large language model		
Team uses sprint review	Team calendar		
Implementation fits acceptance criteria	Project management tool		
Number of issues	Project management tool		
Number of users / number of issues sent to customer support	Project management tool		
Uptime	Cloud dashboard		

Table 4.10: Functional Suitability Evaluation

Performance Efficiency

The quantitative data values identified for the performance efficiency quality characteristic were demonstrated in an interview with a member of the data engineering team. For the difficulty of obtaining data dimension, 7, 3, and 3 quantitative data values were assessed as low, moderate, and high respectively. The values that were evaluated as low are metrics that are readily available by built-in dashboards provided by cloud services, or metrics that can be easily retrieved from external tools, such as code analysis tools. As a result, all metrics assessed as 'low' on the difficulty of obtaining data dimension were also evaluated to be 'low' on the technical expertise required dimension. For all metrics related to the latency per user, the participant labeled the metrics to have a high difficulty of obtaining data. In

the context of the interviewee, there are many types of users (data warehouses consuming data exposed in the application by the interviewee, analysts querying the data). As the source used to determine the latency various per user, and there are various use cases, the participant concluded that there is not a single source, and collecting the data per case would be time intensive. For the technical expertise required, evaluated values were similar to the difficulty of obtaining data. The only difference was found for metrics related to performing retries, where the technical expertise required was assessed as high. The participant stated that knowing how to identify retries in a complex software environment requires a great amount of technical knowledge on the produced software product, and could therefore not be evaluated as moderate. For this quality characteristic, the most commonly identified source was a cloud dashboard. As discussed before, the participant stated that the cloud services used in their work provide dashboards that have built-in functionality for showing some of the metrics for the performance efficiency quality characteristic. Finally, for eight quality characteristics the participant was not able to evaluate a source, and therefore the N/A value has been filled in for the metric. The evaluation of the performance efficiency quality characteristic is displayed in Table 4.11.

Metric	**Source**	**DOOD**	**TER**
Memory utilisation	Cloud dashboard		
CPU utilisation	Cloud dashboard		
Network utilisation	N/A		
Time to interactive	N/A		
Latency per user	Dependent on user		
Latency per increasing user	Dependent on user		
Latency difference per location	Dependent on user		
Resource costs / expected resource costs	Cloud dashboard		
Used current / expected used current	N/A		
Cyclomatic complexity	N/A		
Mobile app battery usage	N/A		
Cost per retry	N/A		
Number of retries	N/A		
Percentage of dead code	Code analysis tool		
Number of connections between source system and other systems	N/A		
Execution time per function	Cloud dashboard		
Execution time per function before refactoring / execution time per function after refactoring	Cloud dashboard		

Table 4.11: Performance Efficiency Evaluation

Reliability

The reliability quality characteristic was evaluated by a member of the enabling team. For the difficulty of obtaining data dimension, 7, 2, and 2 quantitative data values were assessed as low, moderate, and high respectively. For the remaining metrics, a value of N/A was provided. For these metrics, the participant indicated that they would not know how to set up the measurement for the given value, and therefore they could not give an accurate estimation on how difficult it would be to obtain the data and how much technical expertise would be required. For the technical expertise required, 9 metrics were evaluated as low. For both the moderate and high categories, only one quantitative data value was assigned. For one metric (Number of functions that always give the same result on the same input / number of functions), the participant concluded that extensive effort would be required to measure the value, but no source could be identified. For the remaining metrics, a cloud dashboard (similar to the previously discussed source for performance efficiency), was the most commonly named source. The reliability evaluation is shown in Table 4.12.

Metric	**Source**	**DOOD**	**TER**
Uptime / service level agreement uptime	Cloud dashboard		
Uptime	Cloud dashboard		
Mean time between failures	Dependent on failure		
Amount of data lost on failure	Source tracking		
Duration of 1% longest downtime	Logs		
Distribution downtime / max downtime	Cloud dashboard		
Uptime per functionality / service level agreement uptime per functionality	Cloud dashboard		
Number of reliability functionalities implemented / number of reliability functionalities required	N/A		
Number of failed events / number of successful events	Cloud dashboard		
Number of requests / service level agreement number of requests	N/A		
Number of functions that always give the same result on the same input / number of functions	N/A		
Number of backup options	Availability zones		
Number of users	N/A		
Number of functions working under stress test / number of functions	Cloud dashboard		
Accuracy of decimal values	Source data, Target data		
Latency per user	N/A		
Mean response time to incidents	Dependent on incident		
Mean recovery time to incidents	Dependent on incident		

Table 4.12: Reliability Evaluation

Usability

For this quality characteristic, an interview with a software engineer working for the enabling team was conducted. For the difficulty of obtaining data, 6 metrics were evaluated as low, 4 metrics were evaluated as moderate, and 1 metric was evaluated as high. Evaluation for the technical expertise required was identical, except that the interviewee evaluated the technical expertise required for the metric 'number of social media tags' to be low. The rationale behind this choice was that if the data were to be available, it does not rely on a complex technical solution. As the products developed by the enabling team are internal products not mentioned on social media, the interviewee interpreted this measure as the number of times the product was mentioned by other teams in demo's, considering a demo to be a social medium. As it would be a labour intensive and manual process to count the number of times the product would be mentioned, the difficulty of obtaining data was set to high. The most commonly named source was an observability tool, which is a tool that has access to large volumes of data (such as user behaviour data), and reports this data in a highly granular level to its user [30]. During this interview, one metric was assessed as N/A (Number of times user expresses positive emotion). The interviewee expressed that legally it would not be feasible to collect the data, and therefore they could not make a proper assessment on both the difficulty of obtaining data and the technical expertise required. The assessment for the usability quality characteristic is provided in Table 4.13.

Metric	**Source**	**DOOD**	**TER**
Number of clicks to complete task	Observability tool		
Time to complete a task	Observability tool		
Product complies with accessibility standards	User review		
Conversion rate	Observability tool		
Time of usage	Observability tool		
Number of recurrent users	Observability tool		
Number of social media tags	Relevant social media		
Net Promoter Score	User review		
Number of errors in front-end	Analytics tool, user review		
Number of users using the application each day	Observability tool		
Number of times user expresses positive emotion (facial recognition)	N/A		
Difference in task time between fastest user and slowest user	Observability tool		

Table 4.13: Usability Evaluation

Portability

In total, a member of the enabling team evaluated 15 metrics for the portability quality characteristic. For the difficulty of obtaining data dimension, 10, 2, and 2 quantitative

data values were assessed as low, moderate, and high respectively. Looking at the technical expertise required, we conclude that more metrics are evaluated as moderate, with 6 metrics assessed as moderate, 5 metrics assessed as low, and 3 metrics assessed as high. In the discussion on performance efficiency, we concluded that the difficulty of obtaining data and technical expertise required were equal for all quantitative data values retrieved using that source (all metrics that could be retrieved via a cloud dashboard were evaluated as low on both dimensions). This statement does not hold for the evaluation made for this quality characteristic. The source named version control repository, a storage solution used to keep track of different versions of the code base, was used as source for five different metrics. For this source, three quantitative data values were assessed as low on both dimensions, where as two quantitative data values were assessed as high on both dimensions. For the metrics evaluated as low, the interviewee concluded that the measurement could be directly derived from its source, or that an engineer could infer the value through a quick look through the code base. The other two metrics (the time, and number of actions needed to port to a different platform) could not be directly found in the source. Additionally, the interviewee concluded that to measure this value, an analysis of both the current platform and the desired platform is required to test whether a product is actually ported to the new platform. The evaluation for portability is displayed in Table 4.14.

Metric	Source	DOOD	TER
Number of major platforms you can run on without code modifications	Version control repository		
Time to port to different platform	Version control repository		
Number of features functional on platform / number of features still working for new platform	Version control repository		
Number of actions needed to port to different platform	Version control repository		
Number of supported clouds	User agent string		
Platform specific lines of code / lines of code	Version control repository		
Number of cloud agnostic services used	N/A		
Number of supported devices / Number of potential users	Analytics tool, User review		
Number of supported browser versions	User agent string		
Number of supported mobile platforms	User agent string		
Number of supported OS versions	User agent string		
Number of supported real time versions	User agent string		
Lines of code	Version control repository		
Number of functions covered by documentation / number of functions	Documentation platform		
Number of code changes needed per function to port	Documentation platform		

Table 4.14: Portability Evaluation

Compatibility

A member of the enabling team was interviewed to create the evaluation for the compatibility quality characteristic. We can conclude that most quantitative data values for this characteristic score low or moderate on the difficulty dimension, with each of the two categories being assigned 3 and 10 values respectively. On the other hand, 7 metrics were assessed as difficult on the technical expertise required dimension. During the interview, the participant concluded that for many quantitative data values, in depth knowledge on a specific technology was needed. For example, when looking at the percentage of API endpoints compliant to organisation standards, the participant stated that organisation standards can be rather complex, and in-depth technical knowledge would be required to assess whether an endpoint would be compliant. Similar to the portability quality characteristic, the version control repository was the most commonly named source. The evaluation for this quality characteristic can be found in Table 4.15.

Metric	Source	DOOD	TER
Number of data exchange formats used accepted within industry	Version control repository, Domain knowledge		
Number of connected systems	IP tracing		
Percentage of API endpoints compliant to organization standards	Version control repository, Company knowledge		
Product has standardized API documentation	Documentation Platform		
Number of functions exposed by API / Number of functions available for user	Version control repository		
Number of connected systems / Goal of number of connected systems	IP tracing		
Number of successful requests / number of requests	System logs		
Number of protocols used for sharing information	Version control repository		
Number of supported operating systems	Version control repository		
Number of packages / Number of packages that can be shared with different techniques	Version control repository		
Systems are connected through standardized authentication	Version control repository		
Number of data formats not accepted within industry	Version control repository, Domain knowledge		
Number of API functions / number of documented API functions	Version control repository, Documentation platform		

Table 4.15: Compatibility Evaluation

Security

This characteristic was assessed by a member of the data engineering team. The evaluation of this characteristic is summarised in Table 4.16. By looking at Table 4.16, is is apparent that a large number of metrics was evaluated as N/A. In the interview, the participant stated that for the implementation of many security requirements, his team was dependent on a different team within LogOrg. As this team is responsible for managing all cloud environments (and their security) within LogOrg, the evaluation of the metrics would not be in the participants scope. For 14 metrics, the participant was able to make an evaluation. In total, the difficulty of obtaining data was evaluated as low for 9 metrics, moderate for 1 metric, and high for 4 metrics. Looking at the technical expertise required dimension, 10 metrics were evaluated as low, 3 metrics were evaluated as moderate, and 1 metric was evaluated as high. Looking at the sources for the evaluation, we can conclude that to measure the metrics relevant to this characteristic, many different sources are used, varying from built-in security tools provided by the cloud provider used by the interviewee, the version control repository and the resources found in a cloud environment. Additionally, a data contract was specified as a source for three metrics. In the context of the interviewee, a data contract is an agreement

on two parties on how data is shared between two teams. The data contracts used by the interviewee's team also includes technical information, such as a description on how the integration used for sharing data was built.

Metric	Source	DOOD	TER
Number of security protocols used	N/A		
Number of persons for which data is lost	Data Contract		
Percentage of data encrypted	Script		
Product complies to GDPR	GDPR specification		
Security breaches / Goal	Cloud security tool		
Number of users with access to production / Number of users with access	Cloud environment		
Open security issues per severity code	Cloud security tool		
Open security issues for infrastructure	Cloud security tool		
Number of people with admin privilege	Version control repository		
Number of times break the glass procedure activated	N/A		
Number of actions performed with elevated privileges	N/A		
Number of lines of defense	Version control repository		
Number of data breaches per year / number of attacks	N/A		
Number of points of failure	Data contract, Version control repository		
Number of illegal users	Cloud environment		
Product uses SSO for authentication	Company portal		
Number of users not in a security group	Cloud environment		
Number of security layers	N/A		
Product has internet access	Data contract		

Table 4.16: Security Evaluation

Maintainability

The final quality characteristic that was assessed during the domain expert interviews is maintainability. A summary of the evaluation, that was performed with a member of the enabling team, is shown in Table 4.17. By looking at this table, we can conclude that maintainability is the only quality characteristic that did not include any metrics evaluated as high on either one of the two used dimensions. For the difficulty of obtaining data, 6 metrics were evaluated as moderate, where as the remaining 11 metrics were assessed as low. For the technical expertise required dimension, 2 metrics were assessed as moderate, thus having the remaining 15 metrics assessed as low. By looking at the sources, we can see that

most metrics either use the version control repository or the project management tool as a source. For all metrics evaluated as low, the participant stated that to make a measurement, the value could be directly derived from its source, resulting in a low difficulty of obtaining data. As looking up the value in a source is not a complex technical task, the technical expertise required for most metrics was also evaluated as low.

Metric	**Source**	**DOOD**	**TER**
Lead time of new feature	Project management tool		
Number of developers with knowledge of tech stack	Version control repository, Project management tool		
Number of documented functions / number of functions	Documentation platform		
Cycle time of new feature	Project management tool		
Cycle time of bugfix	Project management tool		
Lead time of bugfix	Project management tool		
Code coverage	Test coverage tool		
Percentage of intended maintainers who can maintain	Version control repository		
Time to end of life for tool stack	Tool stack documentation		
Cyclomatic complexity	External tool		
Time until new developer makes first deployment to production	Project management system, version control repository		
Percentage of dead code	Linter, Integrated development environment		
Number of code changes per time frame	Version control repository		
Number of functions / number of classes	Version control repository		
Number of bad code smells	External tool		
Number of domain knowledge holders	Version control repository, Project management tool		
Years of experience maintainers	Survey		

Table 4.17: Maintainability Evaluation

4.3 Case Application

To further answer SQ3 and perform the evaluation phase of the DSRM process, a selected subset of quantitative data values will be measured in a case. The following sections will discuss the selected case, the data values that were measured, and their outcomes.

4.3.1 Selected case

To accurately evaluate the insights given by domain experts, it is essential to evaluate the quantitative data values for a software product within a context that closely mirrors the context of the domain experts. The selected software product is a product created within LogOrg, and is developed using a technology stack resembling the stack used by the interviewed domain experts (cloud applications in AWS, with Typescript used as programming language). To further understand the context of the case, it is vital to elaborate on the organisational and technical context of the software product.

As a large international enterprise, the software ecosystem of LogOrg is a complex environment that consists of numerous software products. All these software applications produce large volumes of data, with data being shared between applications. Some applications expose their data through events, meaning that each time a business activity is conducted, an event is sent out by the application. The goal of the chosen software product is to combine and transform the events from these various sources, and provide a simple operational view that can be used by other teams within LogOrg to answer business questions for various use cases. To fulfill this goal, a cloud application deployed in AWS has been created. Events from various sources are retrieved, and transformed in server-less environments. Afterwards, the events are loaded in a database, which contains the latest state of an event. Each time a change in database state takes place, an event is sent out to the users of the application, notifying them of the newest state. This sequential process is visualised in Figure 4.1. The same process is applied for each source. In total, the software product uses data from five different sources. At the moment of writing, the product is still being fine tuned, and users can not use the application yet.

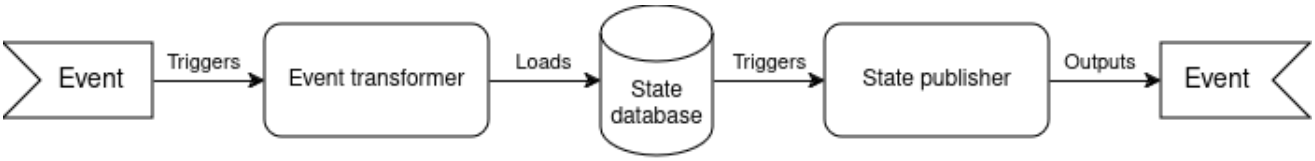

Figure 4.1: Software product event process

4.3.2 Quantitative data values selection criteria

To deal with time-constraints, the decision was made to only measure quantitative data values that experts evaluated as low on both the difficulty of obtaining data, and the technical expertise required dimension. In total, domain experts evaluated 52 out of 127 quantitative measures as low on both dimensions. Furthermore, two exclusion criteria were applied to select appropriate quantitative data values for the case:

- **Exclusion Criterion 1 (EC1): Metrics for which the source was unavailable for the product were excluded**. For instance, a domain expert indicated that the conversion rate of an application could be measured through on observability tool.

When an observability tool is available, the expert evaluated the metric as low on both dimensions. However, the software product under discussion does not use an observability tool. As a result, the metric was not directly measurable for the chosen software product.

- **Exclusion Criterion 2 (EC2): Metrics for which the software product does not have the appropriate characteristics were excluded**. An example of a metric that was excluded based on this exclusion criterion is the compatibility metric 'number of successful requests / total number of requests'. As the application does not expose functionality through an API (and this is not a desired attribute), it is not appropriate to measure this metric for the given software product.

After applying the exclusion criteria, a total of 16 metrics were excluded in the subset of metrics being measured. A list of all the excluded quantitative data values, and the rationale behind exclusion are listed in Table 4.18.

Quality Characteristic	**Metric**	**EC**
Functional Suitability	Implementation fits acceptance criteria	EC2
Performance Efficiency	Resource costs / expected resource costs	EC2
	Execution time per function before refactoring / execution time per function after refactoring	EC2
Reliability	Uptime / service level agreement uptime	EC2
	Uptime per functionality / service level agreement uptime	EC2
	Number of functions working under stress test / number of functions	EC2
Usability	Number of clicks to complete task	EC1
	Time to complete a task	EC1
	Conversion rate	EC1
	Time of usage Number of recurrent users	EC1 EC2
Portability	Number of features functional on platform / number of features still working for new platform	EC2
Compatibility	Product has standardized API documentation	EC1
	Number of successful requests / number of requests	EC1
	Number of API functions / number of documented API functions	EC1
Security	Security breaches / Goal	EC2
Maintainability	Time until new developer makes first deployment to production	EC2

Table 4.18: Quantitative data values excluded from case

4.3.3 Measured data values

After excluding 16 metrics, a total of 36 metrics were evaluated for the software product under discussion.

Selecting a measurement timeframe

In Section 4.1.7, the quantitative data values were categorized into two types: 'over time' metrics (measured over a period to provide insights into past performance or outcomes) and 'point in time' metrics (offering immediate feedback on the software product's state). To measure 'over time' metrics effectively, determining the appropriate time frame for the measurement is necessary. As discussed in Section 4.1.4, one of the focus groups made a remark on selecting an appropriate time frame. Here, the participants argued that when a product scored low on a metric in the past, but efforts were made to improve the score of the metric, the product should not be penalised for its past deficits. To apply these insights to our case, the decision was made to use the time elapsed since the latest release of the software product as the time frame for measuring the quantitative data values. The latest release of the software product was made six days before performing measurement.

Measuring quantitative data values

Using the selected time frame, the remaining quantitative data values were evaluated. Each metric will be assessed to determine whether it could be evaluated from the case, accompanied by a rationale explaining why measuring the metric was feasible or not. Results show that interpretation and measurement are highly dependent on the software product and the context it is deployed in.

Table 4.19 shows how many metrics were evaluated for each quality characteristic, and how many metrics were directly measurable in the case.

Quality Characteristic	**# Metrics**	**# Measurable Metrics**
Functional Suitability	5	2
Performance Efficiency	5	4
Reliability	3	2
Usability	2	2
Portability	4	3
Compatibility	1	1
Security	7	7
Maintainability	9	6

Table 4.19: Number of included metrics per quality characteristic

In total, a measurement was made for 27 metrics. For the remaining metrics, two common reasons for not being able to perform a measurement were identified:

1. For six metrics, it was unclear how the value could be extracted from the source. An example of this is the metric 'Cycle time of new feature'. The used project management tool did not specify which user stories are features, and which user stories are bug fixes. Thus, making it unclear how the quantitative data value could be measured. Other metrics in this category are: 'Number of delivered functions / number of required functions', 'Number of unsolvable issues sent to customer support', 'Distribution downtime / max downtime', 'Cycle time of bugfix' and 'Cycle time of feature'.

2. For two metrics, it was clear how the value could be extracted from its source. However, the value could not be directly derived, and additional effort would be required to make the data available. An example of a metric in this category is 'platform specific lines of code / lines of code'. As the software product has a code base with over 50.000 lines of code, it is a time intensive task to filter on platform specific lines of code. The other metric in this category is 'the percentage of dead code', which is a metric for both portability and performance efficiency,

The final metric that could not be measured and does not fit into the two described categories was 'Large language model agrees that implementation fits goal' for the functional suitability quality characteristic. As LogOrg does not allow sensitive company specific data to be put into external large language models, this metric could not be measured.

The measured quantitative data values are illustrated in Table 4.20. To represent all measurements as numeric values, boolean values have been coded as 1 for the value 'True', and 0 for the value 'False'.

Additionally, for a few metrics, a brief elaboration on how they were extracted from their sources is required. First, to measure 'Memory utilisation' and 'CPU utilisation', the average over a total of 11 serverless functions was used, as the product is not deployed to a single instance. Second, to measure 'the number of major platforms you can run on without code modifications', cloud platforms were considered to be major platforms, as the software product under discussion is a cloud application. Finally, two programming languages were used to develop the product: Python and Typescript. As the metric 'Time to end of life for tool stack' does not indicate what value should be used, the value shows the end of life for the technology with the shortest end of life out of the two.

Metric	**Value**
Team uses sprint review	0
Number of issues	1
Memory utilisation	78%
CPU Utilisation	0.003%
Number of connections between source system and other systems	4
Execution time per function	191 milliseconds
Uptime	99.95%
Number of failed events / number of successful events	0.00000260598%
Number of recurrent users	0
Number of users using the application each day	0
Number of major platforms you can run on without code modifications	1
Lines of code	51574
Number of functions covered by documentation / number of functions	5%
Number of packages / Number of packages that can be shared with different techniques	0
Number of persons for which data is lost	0
Percentage of data encrypted	100
Open security issues per severity code	0
Open security issues for infrastructure	0
Number of people with admin privilege	2
Product uses SSO for authentication	1
Number of users not in a security group	0
Number of developers with knowledge of tech stack	2
Percentage of intended maintainers who can maintain	100
Time to end of life for tool stack	29 months
Number of code changes per time frame	3 since last release (6 days)
Number of domain knowledge holders	2
Years of experience maintainers	3

Table 4.20: Applied metrics

Chapter 5

Conclusion

The goal of this study is to evaluate the extent to which it is possible to measure the quality of a software product using the ISO 25010 standard. To achieve this goal and structure the study, the DSRM process model by Peffers et al. [37] was applied. The main research question was addressed through three sub-questions. The first sub-question was addressed by conducting a literature review. To address the second sub-question, six focus groups were conducted to identify quantitative data values for the eight quality characteristics defined in the ISO 25010 standard. The third sub-question was addressed through two subsequent activities. First, eight interviews with domain experts were conducted to assess the feasibility of measuring the identified quantitative data values in an actual usage scenario. Then, the results from these expert interviews were evaluated by measuring a subset of quantitative data values for a given software product. The results indicate that a set of quantitative data values can be identified, but their measurement depends on the type of software product and the context the product is deployed in.

5.1 Sub-question 1

The first sub-question answered in this research paper was: "How can the characteristics of the ISO 25010 standard be defined?". The eight quality characteristics of the ISO 25010 standard have been further defined in Section 3.1. In this section, the descriptions from the official specification of the ISO 25010 standard were used [23]. Furthermore, the literature review revealed additional insights into the quality characteristics. Specifically, related work found that the relative importance of the quality characteristics was dependent on the domain in which the ISO 25010 model was deployed [13, 24, 26], and that there is a dependency between quality characteristics, meaning that by improving a software product on one of the quality characteristics, others would be affected [19, 21]. Additionally, it was determined that while checklists can assess the quality characteristics of software product quality models, they are deemed insufficient, indicating a need for further research into identifying quantitative criteria [5].

5.2 Sub-question 2

The second sub-question addressed in this study was: "What quantitative data values can be used to measure each of the standards characteristics?". In total, 127 distinct quantitative data values were found during six focus groups. The identified quantitative data values for each quality characteristic are listed in Table 4.2 to Table 4.9. We found that some quantitative data values were suggested during multiple focus groups, thus indicating more evidence for the metric. Additionally, we concluded that some metrics are not applicable for all software products and could only be measured for a specific type of software product (such as mobile applications, or products deploying an event driven software architecture). The found metrics could be divided into two main groups. The first group of metrics consisted of quantitative data values that could be directly measured for a product, providing immediate feedback on the state of the software product. The other group consisted of metrics that could only be accumulated over a period of time as they provide insights into past performances. A final observation made during the focus groups was that the identified metrics only provided value if measured between different iterations of a software product. As each software product is deployed in a unique context, participants concluded that comparing metrics between different products was not useful.

5.3 Sub-question 3

The final sub-question discussed in this paper was: "To which degree can the quantitative data values identified in SQ2 be measured in practical usage scenarios". Results from a pilot interview indicated that selecting a scoped context is essential to determine the source and assess the difficulty of obtaining the data, as well as the technical expertise required to measure a quantitative data value. During a series of domain expert interviews, we demonstrated that there was a difference in the measurability between the identified data values. In total, domain experts determined that 52 out of 127 metrics had both a low difficulty in obtaining the data and a low requirement for technical expertise. The evaluation made by domain experts was presented in Table 4.10 to Table 4.17. We made an attempt to measure these 52 metrics in an actual software product. Results show that out of the 52 metrics, 36 metrics were appropriate for the chosen software product. Out of these 35 metrics, 27 of those could be measured, while the remaining metrics could not be derived due to either requiring additional effort for data collection or due to uncertainty regarding how to collect the metric from its source. The measured values for the 27 quantitative data values that could be measured were shown in Table 4.20.

5.4 Main research question

The answers of the sub-questions can be consolidated to answer the main research question: "To what extent can software product quality be empirically measured using the ISO 25010

standard?". Results indicate that it is possible to identify quantitative data values that empirically measure the quality of software products using the quality characteristics from the ISO 25010 standard. However, the conclusion can be drawn that the applicability of the metric is dependent on the type of software product, and not each identified quantitative data value is generalisable to all software products. Additionally, we conclude that the feasibility of measuring the quantitative data values in an actual usage scenario is also dependent on the type of software product. To make an accurate assessment on the difficulty of obtaining the data and the technical expertise required to set up the measurement, the context in which the measurement is made needs to be specified. Finally, we have found that there is a difference in what experts evaluate to be feasible to measure, and what can actually be measured. Experts expected 55 quantitative metrics to be easily measurable in their context, but only 27 metrics could be measured in a given case performed in a context similar to the context the domain experts operated in. These research findings are in line with related work on the ISO 25010 standard, where the relative importance of the quality characteristics was dependent on the domain the standard was applied in, meaning that to properly apply the ISO 25010 standard, the context of the software product is of importance [13, 24, 26]. As differences were found in the feasibility evaluation made by experts, and the actual feasibility of measuring the quantitative data values in a case conducted in the same domain, we argue that the ISO 25010 model should not only be applied differently between domains, but also within a singular domain between software products.

Chapter 6

Discussion

The final chapter of this thesis will pose a discussion on the research scope, elaborate on the threats to validity of the study, and discuss opportunities for future research.

6.1 Scoping the research

Scoping the research has been an integral part of conducting this research project. By identifying a gap in literature and a need to find a more empirical way of measuring software product quality, the research was first scoped with the goal of creating one generalisable model with quantitative data values applicable to all software products. Subsequently, the research objective would be to assign weights to the created model, so that the model could be used to identify a quality score for a software product. The argument is posed that in the realm of software products, there is too much variety in product type, and the context it is deployed in varies too much to create a list of quantitative data values that are relevant to each product. As a result, it was not desired to assign weights to the created model, as quantitative data values used for measurement would not be equal for all products. As a result, the research was rescoped with the objective to find out how feasible it would be to empirically measure software product quality using the ISO 25010 standard.

6.2 Threats to validity

The validity of the study concerns the trustworthiness of the study, and to what extent the results are not biased by the researchers' perspective [43]. In a study on the guidelines for reporting studies in the domain of software engineering, the conclusion was drawn that four types of validity are relevant to studies within the domain: Construct validity, internal validity, external validity, and reliability [43]. The next four sub-sections will analyse each of these validity threats for our research.

6.2.1 Construct validity

Construct validity refers to the extent to which the studied measure represents what the researcher has in mind [43]. An example related to construct validity in our research is whether participants in the focus groups share the same perception of the quality characteristic as the researcher. When differences in perception align, the created quantitative data values might be inappropriate for the quality characteristic they are supposed to measure, thus introducing a threat to construct validity. To minimise this threat to validity, focus group participants were shown a definition of the discussed quality characteristic, ensuring that the various participants apply the same definition when identifying quantitative measures. This threat to validity could have been further mitigated by applying an additional evaluation phase to the study. During this evaluation phase, the quality of the suggested quantitative data values could be further assessed by asking experts on the ISO 25010 standards whether they agree that the quantitative data value contributes to the construct it is supposed to measure. Furthermore, construct validity is also relevant to the feasibility assessment made during the domain expert interviews. When domain expert interviews do not have the same perception on the meaning of the quantitative data values as the researcher, misalignment in assessment can occur. To mitigate this threat to validity, an interview approach was chosen instead of asking the participants to perform the feasibility assessment via a questionnaire. By conducting an interview, the participants had the opportunity to ask for an elaboration when the definition of the quantitative data value was unclear. Furthermore, a definition of the quantitative data value was provided for metrics that were not self-explanatory. A limitation of this approach is that researchers' bias was introduced when participants asked to elaborate on the definition of the quantitative data values. This resulted primarily from the researcher's previous experience in the software industry. This threat to validity could have better mitigated by conducting an additional feasibility assessment with the participants of the focus group suggesting the quantitative data values. By adding this additional phase, the results from the domain expert interviews could have been better validated.

6.2.2 Internal validity

Internal validity refers to a case when a factor is affected by an investigated factor, but the researcher might be unaware that the factor might also be affected by a third factor [43]. When conducting focus groups, a possible threat to internal validity could be that the conversation might be dominated by a few participants. This introduces bias, as it appears that all participants support the choice for a suggested quantitative data value, but in reality the choice to include the suggested value was made by the dominating party. In this case, the investigated factor (a suggested quantitative data value, and its support among developers) is affected by an unknown third factor (an imbalance between participants). During the domain expert interviews, the threat of internal validity was mitigated by selecting participants working on software products in a similar context. Ideally, this context would be a single development team where each developer works on the same software product. As the sample size was too small to evaluate all quality characteristics with one development

team, participants from two development teams were recruited. This introduces a threat to internal validity, as there might be more unknown differences between the context of the two teams that affect their assessment on the quantitative data values.

6.2.3 External validity

External validity is defined as the extent to which it is possible to generalise the findings, and to what extent the findings are of interest to parties not involved in the case [43]. An example related to the external validity of this study is the degree to which the identified quantitative data values are applicable to all software products. A threat to external validity is posed here, as convenience sampling was used to select participants. As a result of convenience sampling, three out of six focus groups were conducted using participants developing software products for LogOrg. Ultimately, this poses the question whether the found quantitative data values are applicable to software products not developed for LogOrg. To mitigate this threat, the three other focus groups were carried out with participants outside of LogOrg. The same threat of external validity is relevant to the insights drawn from the domain expert interviews and case application. As all domain expert interviews were conducted with engineers at LogOrg, and the selected software product is a product developed at LogOrg, we have to be cautious when making general conclusions. The external validity of the study could be improved by conducting a multi-case study, which is a type of study where multiple cases that differ in some aspects are evaluated [46]. By conducting a multi-case study, the differences and similarities between cases become apparent, thus opening up the possibility to make more generalisable conclusions.

6.2.4 Reliability

Finally, reliability is the degree to which data and its analysis are dependent on the researchers [43]. As the study has identified numerous quantitative data values, and each data value was individually assessed, the research is prone to threats to reliability.Two measures were taken to mitigate threats to reliability. First, all interview transcripts have been anonymised and published on a Github repository. This adds to the reproducibility of the study, as this allows other researchers to make similar decisions to include and exclude quantitative data values from the models. Second, all decisions on the inclusion and exclusion of quantitative data values have been documented in the paper, and the results of inclusion and exclusion are shown in tables. One limitation related to reliability can be found in the case study. The paper does show how a measurement was made for each measured quantitative data value. As a result, a different researcher might come to different results when applying the quantitative data values to the same case. As the sources used for measuring the quantitative data values contain company sensitive information, they can not be exposed in this paper. This threat of reliability could have been mitigated by selecting an open-source case, where permission would be granted to publish the measurements in a scientific paper.

6.3 Opportunities

When addressing the threats to validity, the study can be an inspiration for future studies. One future research avenue could be to address threats in construct validity, and further evaluate the quality of the identified quantitative data values. This can be done by presenting the quality characteristic models to experts conducting related work on ISO 25010. Based on their classification, quantitative data values could be excluded, or possibly moved to more fitting quality characteristics.

In this study, the sub-characteristics of the ISO 25010 standard have not been taken into consideration. A possible research opportunity would be to link the identified quantitative data values to the sub-characteristics. A possible result of this study could be that not all quantitative data values could be linked to a sub-characteristic. When this is the case, the study can evaluate whether the metric is not suitable to be included as a measurement within ISO 25010, or whether the ISO 25010 standard needs to be extended with additional sub-characteristics.

Furthermore, another research opportunity that aims to mitigate threats to external validity could be to apply the found quantitative data values to multiple cases. By applying the model to a series of cases, a deeper understanding on the generelisability of the results is gained. If the model is proven to contain insufficient metrics for a certain type of software products, this study could also identify missing quantitative data values for the evaluated type of software product.

A final interesting research avenue would be to apply our research findings to a case study. In this case study, prioritisation techniques, such as analytical hierarchy process, could be used to assign weights to the quality characteristics and their quantitative data values [2]. When weights are assigned to both quality characteristics and their metrics, measurements can be used to compute a quality score for the software product under discussion. Using the measurements and weights, product managers can make data driven decisions on what features should be built to maximise gains in software product quality.

Bibliography

[1] Adewole Adewumi, Sanjay Misra, and Nicholas Omoregbe. Evaluating open source software quality models against iso 25010. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 872–877. IEEE, 2015.

[2] Kamal M Al-Subhi Al-Harbi. Application of the ahp in project management. *International journal of project management*, 19(1):19–27, 2001.

[3] Hiyam Al-Kilidar, Karl Cox, and Barbara Kitchenham. The use and usefulness of the iso/iec 9126 quality standard. In *2005 International Symposium on Empirical Software Engineering, 2005.*, pages 7–pp. IEEE, 2005.

[4] Vineta Arnicane, Juris Borzovs, and Anete Nesaule-Erina. Do we really know how to measure software quality. 2020.

[5] Behshid Behkamal, Mohsen Kahani, and Mohammad Kazem Akbari. Customizing iso 9126 quality model for evaluation of b2b applications. *Information and software technology*, 51(3):599–609, 2009.

[6] Sebastian K Boell and Dubravka Cecez-Kecmanovic. On being 'systematic'in literature reviews. *Formulating Research Methods for Information Systems: Volume 2*, pages 48–78, 2015.

[7] Pere Botella, Xavier Burgués, Juan-Pablo Carvallo, Xavier Franch, Gemma Grau, Jordi Marco, and Carme Quer. Iso/iec 9126 in practice: what do we need to know. In *Software Measurement European Forum*, volume 2004, 2004.

[8] Lynne M Connelly. Inclusion and exclusion criteria. *Medsurg nursing*, 29(2), 2020.

[9] Rachida Djouab and Moncef Bari. An iso 9126 based quality model for the e-learning systems. *International journal of information and education technology*, 6(5):370, 2016.

[10] Aline Dresch, Daniel Pacheco Lacerda, José Antônio Valle Antunes Jr, Aline Dresch, Daniel Pacheco Lacerda, and José Antônio Valle Antunes. *Design science research*. Springer, 2015.

[11] R. Geoff Dromey. A model for software product quality. *IEEE Transactions on software engineering*, 21(2):146–162, 1995.

[12] John Estdale and Elli Georgiadou. Applying the iso/iec 25010 quality models to software product. In *Systems, Software and Services Process Improvement: 25th European Conference, EuroSPI 2018, Bilbao, Spain, September 5-7, 2018, Proceedings 25*, pages 492–503. Springer, 2018.

[13] Syahrul Fahmy, Nurul Haslinda, Wan Roslina, and Ziti Fariha. Evaluating the quality of software in e-book using the iso 9126 model. *International Journal of Control and Automation*, 5(2):115–122, 2012.

[14] Ronan Fitzpatrick. Software quality: definitions and strategic issues. 1996.

[15] Joyce MS França and Michel S Soares. Soaqm: Quality model for soa applications based on iso 25010. In *ICEIS (2)*, pages 60–70, 2015.

[16] Anita Gibbs. Focus groups. *Social research update*, 19(8):1–8, 1997.

[17] Thelma Jean Goodrich. Strategies for dealing with the issue of subjectivity in evaluation. *Evaluation quarterly*, 2(4):631–645, 1978.

[18] Oleksandr Gordieiev, Vyacheslav Kharchenko, Nataliia Fominykh, and Vladimir Sklyar. Evolution of software quality models in context of the standard iso 25010. In *Proceedings of the Ninth International Conference on Dependability and Complex Systems DepCoS-RELCOMEX. June 30–July 4, 2014, Brunów, Poland*, pages 223–232. Springer, 2014.

[19] Mariem Haoues, Asma Sellami, Hanêne Ben-Abdallah, and Laila Cheikhi. A guideline for software architecture selection based on iso 25010 quality related characteristics. *International Journal of System Assurance Engineering and Management*, 8:886–909, 2017.

[20] Chris Hart. *Doing a literature review: Releasing the research imagination*. Sage, 2018.

[21] Tetiana Hovorushchenko and Oksana Pomorova. Evaluation of mutual influences of software quality characteristics based iso 25010: 2011. In *2016 XIth International Scientific and Technical Conference Computer Sciences and Information Technologies (CSIT)*, pages 80–83. IEEE, 2016.

[22] Ali Idri, Karima Moumane, and Alain Abran. On the use of software quality standard iso/iec9126 in mobile environments. In *2013 20th Asia-Pacific Software Engineering Conference (APSEC)*, volume 1, pages 1–8. IEEE, 2013.

[23] ISO/IEC JTC1/SC7/WG6. Systems and software engineering – systems and software quality requirements and evaluation (square) – system and software quality models. Technical Report 25010, ISO/IEC, 2011.

[24] Millati Izzatillah. Quality measurement of transportation service application go-jek using iso 25010 quality model. *Simetris: Jurnal Teknik Mesin, Elektro dan Ilmu Komputer*, 10(1):233–242, 2019.

[25] Ho-Won Jung, Seung-Gweon Kim, and Chang-Shin Chung. Measuring software product quality: A survey of iso/iec 9126. *IEEE software*, 21(5):88–92, 2004.

[26] Ilham Kadi, Ali Idri, and Sofia Ouhbi. Quality evaluation of cardiac decision support systems using iso 25010 standard. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–8. IEEE, 2016.

[27] Barbara Kitchenham and Shari Lawrence Pfleeger. Software quality: the elusive target [special issues section]. *IEEE software*, 13(1):12–21, 1996.

[28] Leanne M Kmet, Linda S Cook, and Robert C Lee. Standard quality assessment criteria for evaluating primary research papers from a variety of fields. 2004.

[29] Yair Levy and Timothy J Ellis. A systems approach to conduct an effective literature review in support of information systems research. *Informing Science*, 9:181–212, 2006.

[30] Charity Majors, Liz Fong-Jones, and George Miranda. *Observability Engineering*. " O'Reilly Media, Inc.", 2022.

[31] José P Miguel, David Mauricio, and Glen Rodríguez. A review of software quality models for the evaluation of software products. *arXiv preprint arXiv:1412.2977*, 2014.

[32] Hidenori Nakai, Naohiko Tsuda, Kiyoshi Honda, Hironori Washizaki, and Yoshiaki Fukazawa. Initial framework for software quality evaluation based on iso/iec 25022 and iso/iec 25023. In *2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 410–411. IEEE, 2016.

[33] Alison Nightingale. A guide to systematic literature reviews. *Surgery (Oxford)*, 27(9):381–384, 2009.

[34] Philipp Offermann, Olga Levina, Marten Schönherr, and Udo Bub. Outline of a design science research process. In *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, pages 1–11, 2009.

[35] H Panduwiyasa, M Saputra, ZF Azzahra, and AR Aniko. Accounting and smart system: functional evaluation of iso/iec 25010: 2011 quality model (a case study). In *IOP Conference Series: Materials Science and Engineering*, volume 1092, page 012065. IOP Publishing, 2021.

[36] Cecilia Maria Patino and Juliana Carvalho Ferreira. Inclusion and exclusion criteria in research studies: definitions and why they matter. *Jornal Brasileiro de Pneumologia*, 44:84–84, 2018.

[37] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77, 2007.

[38] Emmanuel Peters and George Kwamina Aggrey. An iso 25010 based quality model for erp systems. *Adv. Sci. Technol. Eng. Syst. J*, 5(2):578–583, 2020.

[39] Aditia Arga Pratama and Achmad Benny Mutiara. Software quality analysis for halodoc application using iso 25010: 2011. *Int. J. Adv. Comput. Sci. Appl*, 12(8):383–392, 2021.

[40] Roger S Pressman. *Software engineering: a practitioner's approach*. Palgrave macmillan, 2005.

[41] Junaid Rashid, Toqeer Mahmood, and Muhamad Wasif Nisar. A study on software metrics and its impact on software quality. *arXiv preprint arXiv:1905.12922*, 2019.

[42] Anderson Ravanello, Jean-Marc Desharnais, Luis Eduardo Bautista Villalpando, Alain April, and Abdelouahed Gherbi. Performance measurement for cloud computing applications using iso 25010 standard characteristics. In *2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement*, pages 41–49. IEEE, 2014.

[43] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14:131–164, 2009.

[44] Istiningdyah Saptarini, Siti Rochimah, and Umi Laili Yuhana. Security quality measurement framework for academic information system (ais) based on iso/iec 25010 quality model. *IPTEK Journal of Proceedings Series*, 3(2):128–135, 2017.

[45] Maximilian Speicher. What is usability? a characterization based on iso 9241-11 and iso/iec 25010. *arXiv preprint arXiv:1502.06792*, 2015.

[46] Robert E Stake. *Multiple case study analysis*. Guilford press, 2013.

[47] Vijay Vaishnavi and William Kuechler. Design research in information systems. 2004.

[48] John R Venable, Jan Pries-Heje, and Richard L Baskerville. Choosing a design science research methodology. 2017.

[49] Stefan Wagner. Software product quality control. 2013.

[50] Roel J Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014.

[51] Yu Xiao and Maria Watson. Guidance on conducting a systematic literature review. *Journal of planning education and research*, 39(1):93–112, 2019.

Appendix A

ISO 25010 Sub-characteristics

Table A.1: ISO 25010 Software Product Quality Model Sub-characteristics

Characteristic	Sub-characteristic	Definition
Functional Suitability	Appropriateness	The degree to which the software provides the appropriate functions to meet the needs of the user.
	Accuracy	The degree to which the software provides the correct and exact results.
	Interoperability	The degree to which the software can exchange information and operate with other products.
Performance Efficiency	Time Behavior	The degree to which the software executes its functions within an acceptable timeframe.
	Resource Utilization	The degree to which the software uses the appropriate amount of resources.
	Capacity	The maximum limit of the software's operational capabilities.

Continued on next page

Table A.1 – *Continued from previous page*

Characteristic	Sub-characteristic	Definition
Compatibility	Coexistence	The degree to which the software can coexist with other independent software in a common environment.
	Compatibility	The degree to which the software can run on different hardware, software, or network environments.
	Interoperability	The degree to which the software can exchange information and operate with other products.
Usability	Appropriateness Recognizability	The degree to which users can recognize whether a product or system is appropriate for their needs.
		The degree to which users can recognize whether a product or system is appropriate for their needs.
	Learnability	The degree to which the software can be learned by the user.
Reliability	Maturity	The degree to which a system, product, or component meets specified requirements.
		The degree to which a system, product, or component meets specified requirements.
	Fault Tolerance	The degree to which a system, product, or component operates as intended despite the presence of hardware or software faults.

Continued on next page

Table A.1 – *Continued from previous page*

Characteristic	Sub-characteristic	Definition
Security	Confidentiality	The degree to which a system, product, or component ensures that data are accessible only to those authorized to have access.
	Integrity	The degree to which a system, product, or component prevents unauthorized access to, or modification of, computer programs or data.
	Non-repudiation	The degree to which a system, product, or component provides the means to ensure that parties to a communication or transaction cannot deny the authenticity of their signatures or the validity of their actions.
Maintainability	Modularity	The degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.
	Reusability	The degree to which a system, product, or component can be used in more than one system, or in building other systems.
	Analyzability	The degree to which it is possible to assess the impact on a system, product, or component of an intended change to one or more of its parts.

Continued on next page

Table A.1 – *Continued from previous page*

Characteristic	Sub-characteristic	Definition
Portability	Adaptability	The degree to which a system or component can effectively and efficiently be adapted for different or evolving hardware, software, or other operational environments.
	Installability	The degree to which a system, product, or component can be successfully installed and/or uninstalled in a specified environment.
	Replaceability	The degree to which a system, product, or component can be replaced with another specified system, product, or component.

Appendix B

Focus group results

Table B.1: Focus group 1: quantitative data values

Quality Characteristic	**Metric**
Functional Suitability	Number of changes needed to fit to original requirements
	Market share
	Net Promoter Score
	Number of unsolvable issues sent to customer support
Performance Efficiency	Memory utilisation
	CPU utilisation
	Network utilisation
	Latency per user
	Latency per increasing user
	Latency difference per location
Reliability	Uptime / service level agreement uptime
	Mean time between failures
	Amount of data lost on failure
	Duration of 1% longest downtime
Usability	Number of clicks to complete task
	Product complies with accessibility standards
	Text to speech: how long to get to relevant records
Portability	Platform specific lines of code / lines of code

Continued on next page

Table B.1: Focus group 1 - quantitative data values (Continued)

Quality Characteristic	**Metric**
	Number of major platforms you can run on without code modifications
Compatibility	Product has standardised API documentation
	Number of functions exposed by API / Number of functions available for user
	Systems are connected through standardised authentication
	Number of API functions / number of documented API functions
Security	Number of persons for which data is lost
	Percentage of data encrypted
	Security controls of common chosen framework implemented
Maintainability	Cycle time of new feature
	Lead time of new feature
	Cycle time of bugfix
	Lead time of bugfix
	Code coverage
	Maturity of CI/CD

Table B.2: Focus group 2: quantitative data values

Quality Characteristic	**Metric**
Functional Suitability	Number of delivered functions / Number of required functions
	Number of delivered functions / Number of functions that contribute to main goal
Performance Efficiency	Resource costs / Expected resource costs
	Used current / Expected used current
	Cyclomatic Complexity
Reliability	Uptime / service level agreement uptime
	Distribution downtime / Max downtime

Continued on next page

Table B.2: Focus group 2 - quantitative data values (Continued)

Quality Characteristic	**Metric**
	Uptime per functionality / service level agreement uptime per functionality
Usability	Conversion rate
	Time of usage
	Number of social media tags
Portability	Number of major platforms you can run on without code modifications
	Number of features functional on platform / number of features still working for new platform
	Number of cloud agnostic services used
Compatibility	Number of connected systems / Goal of number of connected systems
	Number of data exchange formats used accepted within industry
	How actual is used technique
Security	Product complies to GDPR
	% that fits security requirements
	Security breaches / Goal
Maintainability	Percentage of intended maintainers who can maintain
	Number of config parameters required to change
	Time to end of life for toolstack

Table B.3: Focus group 3: quantitative data values

Quality Characteristic	**Metric**
Functional Suitability	Number of clicks before action is finished
	Number of delivered functions / Number of required functions
Performance Efficiency	CPU Utilisation
	Network utilisation
	Memory utilisation
Reliability	Number of reliability functionalities implemented / number of reliability functionalities required
Usability	Conversion rate

Continued on next page

Table B.3: Focus group 3 - quantitative data values (Continued)

Quality Characteristic	**Metric**
	Time to complete a task
	Number of clicks to complete task
	Number of recurrent users
Portability	Time to port to different platform
Compatibility	Number of interfaces
	Number of successful requests / number of requests
Security	Number of users with access to production / Number of users with access
Maintainability	Cyclomatic complexity
	Total Language Popularity
	Number of documented functions / number of functions

Table B.4: Focus group 4: quantitative data values

Quality Characteristic	**Metric**
Functional Suitability	Percentage of succeeding tests
	Test coverage
	Net Promoter Score
	Percentage of pixels equal to UI design
	Load Test / Target Load
	Prod test
Performance Efficiency	Mobile app battery usage
	Time to interactive
	Cost per retry
	Number of retries
	Percentage of dead code
	CPU utilisation
	Network utilisation
	Memory utilisation
Reliability	Uptime
	Number of failed events / number of successful events
	Number of requests / service level agreement number of requests
Usability	Net Promoter Score
	Product complies with accessibility standards

Continued on next page

Table B.4 – continued from previous page

Quality Characteristic	**Metric**
	Number of errors in frontend
	Number of recurrent users
Portability	Number of supported devices / Number of potential users
	Number of supported browser versions
	Number of supported mobile platforms
	Number of supported OS versions
	Number of supported clouds
	Number of supported real time versions
	Lines of code
Compatibility	Percentage of API endpoints compliant to organisation standards
	Number of data exchange formats used accepted within industry
	Number of connected systems
	Number of protocols used for sharing information
Security	Open security issues per severity code
	Open security issues for infrastructure
	Number of people with admin privilege
	Number of times break the glass procedure activated
	Number of actions performed with elevated privileges
Maintainability	Lead time of new feature
	Time spent between OPS and DEV
	Time to first code deployed to prod
	Percentage of dead code

Table B.5: Focus group 5: quantitative data values

Quality Characteristic	**Metric**
Functional Suitability	Percentage of succeeding tests
	Large language model agrees that implementation fits goal
	User acceptance test
	Team uses sprint review
	Implementation fits acceptance criteria
Performance Efficiency	Time to interactive

Continued on next page

Table B.5: Quality Characteristics and Metrics (Continued)

Quality Characteristic	**Metric**
	Number of connections between source system and other systems
	Stress test
Reliability	Number of functions that always give the same result on the same input / number of functions
	Logging per function
	Number of backup options
	Number of users
	Number of differences between software instantiations
	Number of functions working under stress test / number of functions
	Accuracy of decimal values
Usability	A/B testing
	Number of clicks to complete task
	Time to complete a task
	Time of day
	Number of users using the application each day
	Time of usage
	Number of times user expresses positive emotion (facial recognition)
Portability	Number of skills needed to change environments
	Number of functions covered by documentation / number of functions
	Number of supported clouds
	Number of code changes needed per function to port
	Number of actions needed to port to different platform
	Time to port to different platform
Compatibility	Types of fields
	Number of supported operating systems
	Number of changes to data structure
	Number of data exchange formats used accepted within industry

Continued on next page

Table B.5: Quality Characteristics and Metrics (Continued)

Quality Characteristic	**Metric**
	Number of data formats not accepted within industry
Security	Number of lines of defence
	Number of security protocols used
	Number of data breaches per year / number of attacks
	Number of points of failure
Maintainability	Number of code changes per timeframe
	Number of functions / number of classes
	Number of bad code smells
	Number of developers with knowledge of techstack

Table B.6: Focus group 6: quantitative data values

Quality Characteristic	**Metric**
Functional Suitability	Number of issues
	Number of search terms in search functionality
	Click behaviour
	Number of active users / Number of issues sent to customer support
	Uptime
Performance Efficiency	Logging
	Execution time per function
	Execution time per function before refactoring / Execution time per function after refactoring
	Memory utilisation
	CPU utilisation
	Asking users if it fits performance needs with a large sample
Reliability	Latency per user
	Uptime
	Mean response time to incidents
	Mean recovery time from incidents
Usability	User analysis: fits software implementation

Continued on next page

Table B.6: Quality Characteristics and Metrics (Continued)

Quality Characteristic	**Metric**
	Difference in task time between fastest user and slowest user
	Type of users
	Time to finish task
Portability	Number of actions needed to port to different platform
	Number of features functional on platform / number of features still working for new platform
Compatibility	Number of connected systems
	Number of packages / Number of packages that can be shared with different techniques
	Version of the software / number of use cases
Security	Software product is up to date
	Number of illegal users
	Product uses SSO for authentication
	Number of users not in a security group
	Number of security layers
	Product has internet access
	Number of security protocols
Maintainability	Used programming paradigm
	Number of developers with knowledge of techstack
	Number of domain knowledge holders
	Years of experience of maintainers
	Number of documented functions / number of functions