Software requirements in public procurement

University of Turku
Department of Computing
Master of Science (Tech) Thesis
Software Engineering
July 2026
Artturi Vehviläinen

UNIVERSITY OF TURKU
Department of Computing

Artturi Vehviläinen: Software requirements in public procurement

Master of Science (Tech) Thesis, 73 p., 4 app. p.
Software Engineering
July 2026

Public procurement is vital for fulfilling the basic functionalities of a country and every year billions of euros are spent on it. ICT is one of the biggest procurement categories, annually amounting to over billion euros. With society becoming more and more dependent on digital solutions, the importance of ICT procurement is only going to grow.

In 2020 the Ministry of Finance published a report detailing the current state of public procurement in Finland, which discovered that there is a lack of understanding on the competency of procurement. A self-assessment tool was created to combat this problem, which is only a partial solution, as it does not measure the procurement process or the artifacts created by it. This represents a research gap.

Software requirements are one of the most important artifacts for the success of an ICT project. This thesis contributes to the understanding of procurement competency in Finland with a focus on requirements. A literature review was conducted in order identify key attributes for the quality of requirements. Then, the quality indicator metrics were designed. The metrics were then used on requirements found on 19 public requests for proposals. The results show that the RFPs wary greatly in quality, which supports the need for more research on procurement competency.

Keywords: Public procurement, Software requirements, Quality of requirements

Contents

1 Introduction

Public procurement means the purchasing of products or services by a public entity like a municipality or the state. It is vital for fulfilling the basic needs of a country from healthcare to infrastructure and every year billions of euros are spent on procurement in Finland [1], [2]. Spending these euros as efficiently as possible on projects that most benefit the citizens is a goal worth seeking.

In 2020 the Ministry of Finance published a report detailing the current state of public procurement in Finland [3]. The report discovered that there is a lack of understanding on the competency of procurement in Finland [3]. Later that year ProcurFinland, an initiative involving the Ministry of Finance, the Association of Finnish Cities and Municipalities, the Confederation of Finnish Industries, the Association of Finnish Entrepreneurs and Hyvinvointialueyhtiö Hyvil Ltd, launched the national public procurement strategy. One of its goals was measuring the level of competency in public procurement [4] and a self-assessment tool was created to achieve this goal [5], which is still the initiative's only way of measuring competency [6], [7].

While a self-assessment tool measures the perceived competency of individuals, it does not measure the competency of the procurement process or the artifacts created by it. For example, we know that 1159 procurement professionals rated their competence in planning a pronouncement on average as 2,17 out of 4 [8], but we do not know how this actually reflects on the artifacts produced by this planning. The

perceived competency of procurement has also been called to question by a survey on industry professionals [9]. This represents a significant research gap.

The quality of procurement artifacts, like requests for proposals (RFP) that define the product or service being procured, is really important due to the legislation surrounding procurement. To keep the procurement process transparent and fair for all, what is written down must, more or less, hold. Once the documents have been published, it becomes more difficult to make significant altercations to them [10, Chapter 3].

ICT is one of the largest procurement categories in Finland, and on its own amounts to over billion euros annually [1], [2]. At the same time, public ICT procurement has quite a bad reputation in Finland, as headlines of failed or poorly received software projects seem all too common [11]. Stakes for success in public software procurement are high, because money spent on failed projects could have been used in other critical avenues, like healthcare. For this reason, failed publicly procured software projects face more scrutiny than those in the private sector. Therefore, it is paramount to see as many projects succeed as possible.

As ICT projects often include complex software systems, proper planning and clear project goals are critical for a project's success [12]–[14]. The request for proposal is the document that outlines the projects goals in public procurement and in software procurement it contains the project's software requirements. Software requirements document what functionalities, qualities and constraints a software is expected to fulfill, and they are paramount for a software project's success according to industry professionals [15] and procurement professionals [16].

If understanding and improving the competency of procurement is the goal, then, in the case of ICT procurement, evaluating the quality of requirements could provide valuable insights. This thesis aims to bridge the research gap in Finnish

procurement competency measurement in the context of software requirements in RFPs by answering the following research questions:

- **RQ1: How can the quality of requirements in public RFPs be measured?**
- **RQ2: What is the quality of requirements documentations in Finnish RFPs?**
- **RQ3: Are there differences in quality between functional and non-functional requirements in Finnish RFPs?**

Quality is an abstract concept that can not be simply be measured without first defining what it means. RQ1, therefore, aims to identify attributes of requirements that indicate quality and translate them to practicable metrics.

RQ2 aims to shed light on Finnish procurement competency by evaluating requirements found in Finnish RFPs.

RQ3 examines whether there are differences in quality indicators between functional and non-functional requirements. The reasons for this framing are perceived difficulties of NFRs and previously found deficiencies in NFRs in Finnish RFPs [17].

Chapter 2 provides background information on public procurement in Finland. The chapter goes over a typical ICT procurement process and the relevant legislation. It also highlights the role of requirements in ICT procurement. The aim of the chapter is to provide the context for the rest of the thesis as it focuses mostly on requirements.

A literature review is conducted in Chapter 3. It examines requirements from three different viewpoints to identify the attributes that are best suited for evaluating quality of requirements in public RFPs.

The quality indicator metrics are introduced in Chapter 4. Scoring and reason for inclusion are explained for each metric. Exclusion rationale is also explained for a few considered metrics.

Chapter 5 explains the methodology of the requirements analysis. An overview of the research design is given and data collection, sampling and analysis are further elaborated on.

Chapter 6 presents the results of the requirements quality analysis. The chapter includes multiple tables and figures of the results. Appendix A includes color coded version of the results.

The results are interpreted and discussed in Chapter 7. This includes discussion on the designed quality metrics and on the requirements analysis.

Chapter 8 summarizes the thesis.

1.1 Use of AI declaration

Artificial intelligence tools were used for search query conversion and table formatting in this thesis.

The search query was first formed for ACM Digital Library by the author, and then converted to be suitable for the other databases by an AI tool. Tables featuring vertical text (3.2, 3.3, 3.4 and 6.1) were formatted with the help of an AI tool.

All text, analysis and interpretations were produced by the author.

2 Public procurement

This chapter explains the basic concepts to Finnish public ICT procurement. This is not meant as a comprehensive deep dive into the topic but rather as an introduction that helps contextualize the rest of the thesis. Section 2.1 gives an overview of the procurement process and Section 2.2 further explains software requirements in the context of public procurement.

2.1 Public procurement process in Finland

Procurement can mean any situation where a client buys products or services from a supplier, and how this transaction is handled, is really up to the parties involved. However, when the client is a public entity, such as the state or a municipality, the transaction is heavily regulated in accordance with national and EU legislation [10, Chapter 1]. This regulation is what sets public procurement apart from private procurement and has significant ramifications on how it is done. This chapter gives an overview of the public procurement process and how it is shaped by the legislation, with a focus on software procurement[1].

The laws that govern public procurement[2] are the national "Act on Public Procurement and Concession Contracts" 1397/2016 [18] and the EU directive 2014/24/EU [19]. The aim of these regulations is to ensure that public procurement

[1]This means that any special cases, like those around construction, are not discussed.

[2]There are other laws that govern special cases in public procurement that are not discussed in this Chapter.

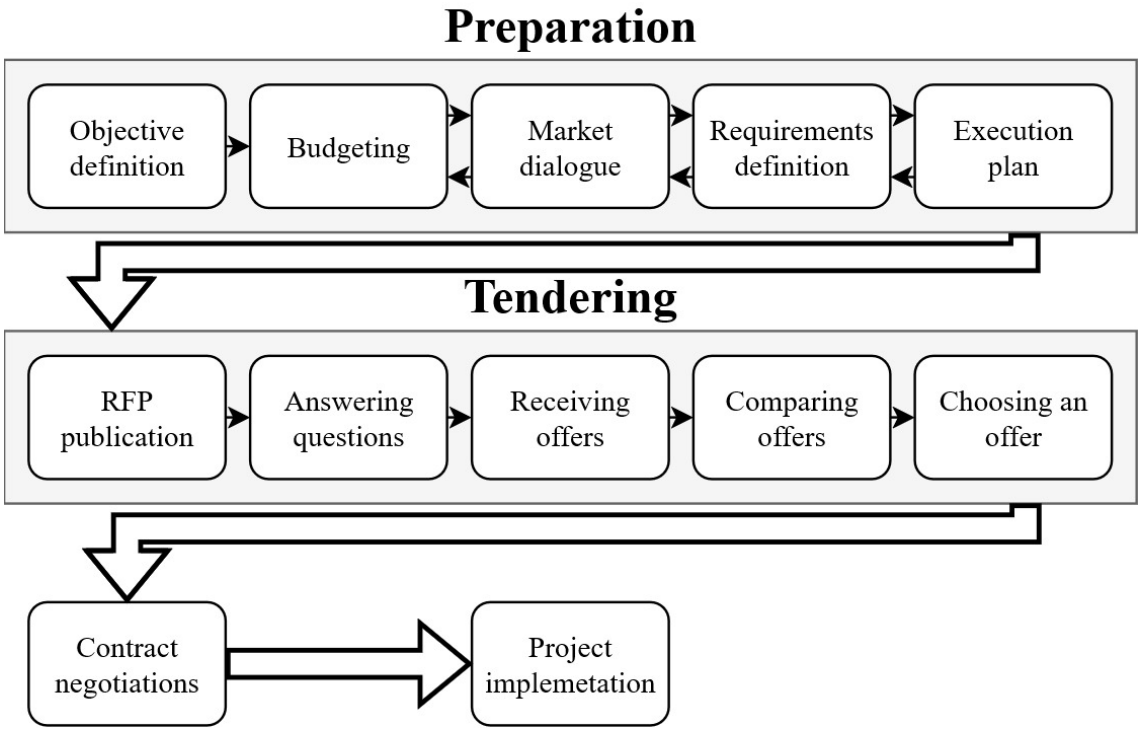

Figure 2.1: Overview of the public procurement process

is fair, nondiscriminatory and transparent and that public funds are spent efficiently [10, Chapter 1.2].

The required procurement procedure is dictated by the estimated value of the procured product or service. Procurements under 60 000 euros fall outside the procurement act and can be done in any way the client organization sees fit. Any procurements valued over 60 000 euros must be done in accordance of the national procedure and those over 140 000 or 216 000 euros, depending on the client, must be done as an EU procurement, which is a stricter procedure. [10, Chapter 1.5]

Figure 2.1 presents an overview of the procurement process with a focus on preparation and tendering. While other procurement procedures exist, only open procedure is explained in this chapter. This is done because understanding the intricacies between different procurement procedures is not necessary for this thesis.

Open procedure was selected because it is the most common among the projects analyzed in Chapter 6; 18 out of the 19 projects used open procedure.

Every procurement process starts with a need or an objective and the first step is defining this need [16], [20, Chapter 4.2]. In other words, the procuring organization has a problem they are trying to solve with procurement and first they need to clearly define what the problem is and what counts as solving it. This step is critical as every subsequent step relies on it.

Next the procurement needs to be budgeted [10, Chapter 2.1], [20, Chapter 4.2]. Both the procurement process and the procured system need to be budgeted and the breadth of the procurement process can be reliant on the budget [10, Chapter 2.1].

The preparation phase may include market dialogue or technical dialogue, which means communicating with potential suppliers [10, Chapter 2.1], [16]. Market dialogue helps the procuring organization understand what kinds of solutions and capabilities are available on the market and allows the suppliers to give feedback on the planned procurement [16]. This can lead to better procurement outcomes as procurers gain a better understanding on the market and the suppliers get familiar with the objectives of the procurement [16].

After the project objectives and market capabilities have been understood, it is time to define the system that is being procured. This is done through requirements, which describe the system's functionalities, constraints and qualities [16], [20, Chapter 4.3]. This is usually the most labor intensive preparation stage [20, Chapter 4.3]. At the same time it is vital for a successful software projects, as implementing a high quality software system from poor requirements is difficult [16], [20, Chapter 4.3]. Section 2.2 elaborates more on this stage of preparation.

Finally, the execution of the procurement needs to be planned and prepared. This includes scheduling and preparing the request for proposal and choosing the procurement procedure [20, Chapter 4.6]. The request for proposals is a document

that outlines the procured system and its requirements along with the comparison criteria for offers [20, Chapters 5.2-5.3]. The RFP is the culmination of the preparation phase and it aims to give suppliers as clear of an understanding of the procured system as possible so that the following offers will be the highest possible quality. The procurement procedure determines how the supplier is selected. This chapter focuses on open procedure.

The tendering starts with the publication of the RFP, which is done on HILMA[3]. The RFP is publicly available, but the procuring organization may also send it directly to suitable suppliers [10, Chapter 4.2.5]. If a supplier has questions about the RFP, they can ask them during the tendering process. All questions and answers are public for all participants to ensure that the process is transparent and fair. [10, Chapter 5.8]

Interested suppliers respond to the RFP with their offers that include their references, pricing and answers to the system requirements [20, Chapter 5.5]. The procuring organization then confirms that the offers fulfill the mandatory requirements and score them based on the declared comparison criteria [20, Chapter 5.7].

The winning offer is either the cheapest offer or one with the best price-quality ratio [10, Chapter 5.5], [20, Chapter 5.7]. Robust comparison criteria are key in this phase as the winner has to be based on them and no arbitrary decisions are allowed in the name of transparency and fairness.

Once the winner has been decided, contract negotiations can begin between the parties, after which implementation can begin. This chapter does not elaborate on those phases as the focus of this thesis is on the preparation phase and how it can affect the subsequent phases.

[3]https://www.hankintailmoitukset.fi

2.2 Requirements in procurement

Requirements define a given systems desired functionalities, constraints and qualities. In other words, requirements describe what the system needs to do, how it needs to do it and in what kind of an environment. Well made requirements can not only be used for implementation but also for verification and evaluating scope and risks [20, Chapter 4.3]. Requirements need to be clear and comprehensive so that the supplier fully understands the procured for system. This helps suppliers to make the best possible offers and poor requirements might even scare potential suppliers away, as they do not feel like they understand what they are signing up for [20, Chapter 4.3].

Comprehensively describing the system is, however, not sufficient as the system will most likely be a part of a larger whole and describing this environment is just as important [16]. Integration and architecture requirements are often necessary for the solution's implementation in the environment it is used in, and their early identification and definition can prevent future procurement and maintenance costs [16], [20, Chapter 4.4].

The role of requirements in procurement is slightly different compared to traditional software development. This due to the fact that when developing a software, requirements engineering can be the step when many details of the software are decided and codified in the requirements, but requirements for procurement need to let room for different kinds solutions that the suppliers might offer [16], [20, Chapter 4.3]. This is because it is not in the client's interest to disqualify innovative solutions with overly specific requirements [16], [20, Chapter 4.3] and because too strict requirements might be against the fairness principle of the procurement act [10, Chapter 5.3].

This makes requirements engineering in public procurement a delicate balancing act where the requirements need to be comprehensive enough to give the suppliers a

good idea of the desired system but not so specific that it restricts potential offers. This is why competency is important when it comes to ICT procurement and why special attention for requirements in RFPs is warranted.

3 Literature review

This literature review is one part of answering RQ1: "How can the quality of requirements be measured". The goal of this literature review is to identify attributes of requirements that can be used in the creation of requirements quality indicator metrics in Chapter 4. The metrics are intended to be used on requirements in public requests for proposals, which affects the framing of this literature review.

Since the requirements analysis is done on public RFPs, only the written requirements can be analyzed and not the requirements engineering process as a whole. For this reason the literature review focuses on requirements specification, documentation, validation and verification, and is not focused on other aspects of requirements engineering like requirements elicitation. In other words; the focus is on "What are good software requirements like?" and not on "How to get good software requirements?".

The literature review was conducted using three scientific databases: ACM Digital Libray, IEEE Xplore and ScienceDirect. The literature review was limited to articles published in 2020 and after to get an understanding of the current state of software requirements. The following search query was modified to fit each database and used:

```
Title: requirement? AND (software OR system? OR engineering
OR specification? OR document* OR verif* OR validati*)
```

The query consists of three parts. The first part is "requirement?" on one side of the AND operator, which has to be present in every search result. The wildcard operator is used because, while the plural "requirements" is more common, the singular "requirement" is also used. The second part are the three 'catch all' terms after AND: "software OR system? OR engineering". These are used to find all papers discussing software requirements or requirements engineering more holistically because they might also contain valuable insights on the areas this thesis is focused on. The final part includes all the more specific areas of requirements engineering that are of particular interest.

Quoted search terms, like "software requirements", were avoided for two reasons. The databases used do not support wildcard operators inside quoted terms. The terms show up in different permutations. For example: "requirements verification" and "verification of requirements" are both possible. While quoted terms would have limited the amount of articles irrelevant to this thesis, they were not used to maximize the amount of relevant articles.

To reduce the number of irrelevant articles, filtering tools, provided by the databases, were used. The following filters were used:

```
ACM:
Content Type: Research Article

IEEE:
Journals, Conferences
Requirements Engineering, Software Requirements,
Software Development, Software Engineering,
Non-functional Requirements, Functional Requirements

ScienceDirect:
```

```
Review articles, Research Articles
Engineering, Computer Science, Decision Sciences
```

The query returned 1900 articles: 272 from ACM, 1135 from IEEE and 493 from ScienceDirect. Applying the filters reduced the number of articles to 1045. Relevant articles were selected in three stages: based on title, based on abstract and based on full article.

A number of inclusion and exclusion criteria were defined to guide study selection. The criteria were informed by the goals of the literature review and are listed in Table 3.1.

Of the 1045 articles, a total of 162 were selected based on title, 59 based on abstract and finally, 40 articles were selected based on the full text. This process is visualized in Figure 3.1.

Table 3.1: literature review inclusion and exclusion criteria

Inclusion criteria	**Exclusion criteria**
Focus on software requirements	Focus on requirements not related to software e.g. industrial requirements
Focus on RE stages related to written requirements e.g. requirements documentation	Focus on RE stages not related to written requirements e.g. requirements elicitation
Discussion on characteristics of requirements e.g. ambiguity	
Discussion on requirements methodologies e.g. Use Case	
	Not in English
	Not accessible through University of Turku library agreements

3.1 Themes

40 studies were analyzed, and 34 relevant themes were identified. These themes were divided into 3 distinct viewpoints through which requirements can be examined: requirements engineering stages, requirements engineering methodologies and

Figure 3.1: literature review process

characteristics of requirements. Requirements engineering stages are the different phases of the RE process, requirements engineering methodologies are the different ways requirements can be made, and characteristics of requirements are the different qualities of written requirements that can be analyzed and evaluated.

Each viewpoint was evaluated for how well they fit the goal of thesis of measuring the quality of requirements found in public RFPs. As the documentation found in public RFPs does not contain information on the different requirements engineering stages that lead to final the requirements, it is not a suitable viewpoint to base the quality metrics on. Section 3.2 gives an overview of the different RE stages to serve as context for this thesis.

Requirements engineering methodologies are not a suitable viewpoint either, due to a combination of two factors. Firstly, no methodology is inherently good or bad and the quality in methodologies comes down to how suitable they are for the task and how well they are applied. Secondly, the methodologies actually used in the

RFPs are very homogeneous. For these reasons, Section 3.3 serves as a frame of reference for the analysis as to what methodologies exist.

This means that characteristics of requirements is chosen as the basis for the quality indicator metrics in Chapter 4 and for the requirements analysis in Chapter 6. Characteristics of requirements can be analyzed and evaluated with access to just the requirements and they can be determined as either good or bad for the quality of requirements. While, both of these statements are simplifications of the reality and have caveats in this thesis, this still makes characteristics the best viewpoint overall for quality evaluation. Section 3.4 is a deep dive into the different characteristics a requirement might possess.

In addition to the articles selected for the literature review, the ISO/IEC/IEEE 29148 standard [21] is used in this chapter. ISO/IEC/IEEE 29148 is an internationally recognized standard that outlines the best practices for writing and managing requirements in software engineering.

3.2 Requirements engineering stages

Requirements engineering progresses in stages. Different sources use different distinctions and names for these processes [21]–[29] but, for the most part, the following four broad categories are represented: identifying stakeholder needs, transforming needs into requirements, validating requirements and managing requirements. These stages are usually done in the listed order, and each stage supports the subsequent stages [22], [23]. This Section aims to give an overview of the different stages of requirements engineering with a focus on stages relevant to the analysis in Chapter 6.

Identifying stakeholder needs involves processes like business analysis, requirements elicitation and requirements analysis [21], [27]. The purpose of this phase is to create an understanding of the problem and its possible solutions. This stage

directly supports the following two stages. A proper understanding of stakeholder needs is obviously mandatory if said needs are to be codified into requirements. Later, the identified needs are used in validating the requirements. Ultimately, any software is made to satisfy some needs, thus understanding those needs is pivotal. This thesis does not focus on this phase of requirements engineering as the data analyzed in Chapter 6 does not include sufficient information on the used elicitation and analysis practices.

Transforming needs into requirements involves processes like documentation, refinement, modeling and specification. The aim of this process is to transform the stakeholder needs gathered in the previous stage into various requirements artifacts [21], [27]. The artifacts can be abstract models like UML-diagrams or formally defined requirements. The primary artifact that is created during this process is the requirements specification, which is a document containing the requirements of the developed system [30], [31]. This stage is important because unclear documentation makes implementing the software more difficult and risks errors in the implementation [24], [30]–[32].

Validating requirements involves processes like validation, verification and testing. This process is intended to prove that the requirements fulfill stakeholder needs (validation) and that the system implementation is in accordance with the requirements (verification). In other words, making sure the software works as intended. This helps keep maintenance costs down because more defects caught now means less to be dealt with later [33].

Managing requirements refers to the processes of handling change in the requirements engineering process [21]. Small changes to requirements can have unexpected consequences if not properly managed [23]. Requirements management is not a focus in this thesis because the requirements analyzed in Chapter 6 are static and there is no change or its management to analyze.

The most relevant stages to this study are transforming needs into requirements and validating requirements, because they more explicitly deal with requirements artifacts. Themes relating to these stages identified in the literature review are listed in Table 3.2.

Table 3.2: Themes concerning the stages of requirements engineering relevant to this thesis

Articles	Documentation	Refinement	Modeling	Specification	Testing	Validation	Verification
Akinnuwesi et al. [34]						x	
Alhazmi & Huang [27]	x					x	
Alshareef et al. [33]						x	x
Amaral et al., [35]	x						
Arenas et al. [36]		x		x			
Arif et al. [28]			x				
Awan et al. [24]			x	x			
Bacquet et al. [37]				x		x	x
Behutiye et al. [38]	x						
Behutiye, Seppänen, et al. [39]	x					x	x
Bruel et al. [40]						x	x
Bugayenko [41]				x	x		
Frattini & Frattini [42]				x			
Gérançon & Trudel [43]				x			
Hagal & Saeid [25]						x	
Jarzębowicz & Weichbroth [44]	x						
Karolita et al. [29]						x	
Kasauli et al. [45]	x						
Kravari et al. [22]						x	
Mashkoor et al. [46]						x	
Medeiros et al. [47]				x			
Mokos & Katsaros [48]						x	x
Nakamichi et al. [49]				x			
Nasir et al. [50]	x						x
Ramesh & Reddy [51]				x			
Ribeiro & Berry [30]				x			
Silva [31]				x			

3.3 Requirements engineering methodologies

Requirement engineering methodologies are the various techniques that can be used when writing requirements. As can be seen from Table 3.3, many different methodologies exist and there is not any one methodology that is over represented. Indeed, different projects call for different methodologies, which makes understanding the strengths and weaknesses of the given techniques important for choosing the ones that fit the project. The purpose of this section is to give an overview of the different methodologies identified in the literature review.

The themes in this section are grouped into three different categories that get their own subsections. The subsection Software development practices delves into the differences in requirements practices when it comes to Agile and other practices, Formalization of requirements is about the different levels and techniques of formalization, and Requirements artifacts discusses the different artifacts that requirements can be documented as.

Software development practices

Software development practices are overarching principles that guide the entire software development process. While requirements are a part of all mainstream practices, there are differences in how they interface with requirements [27]. In this literature review, 9 studies were identified that researched requirements in Agile development.

With the exception of one study that compared Agile and "traditional methods" [27], Agile was the only software development practice as a subject for research in this literature review. This can be partly explained by the literature review only including studies from the 2020s, as Agile is currently the most popular practice, but even then the result is surprising. Another reason for the prevalence of Agile seemed to be its unique challenges with requirements engineering.

Table 3.3: Themes concerning requirements engineering methodologies

Articles	Agile	Boilerplates	CNL	Formal methods	Natural language	NLP	Notation	Object-oriented	Ontology	Patterns	Personas	Semantics	UML	Use Case	User story
Alhazmi & Huang [27]	x														x
Alrumaih et al. [52]									x						
Amaral et al., [35]							x								
Amorndettawin & Senivongse [53]	x									x					x
Arenas et al. [36]											x				
Arif et al. [28]													x		
Awan et al. [24]				x									x		
Bacquet et al. [37]		x		x	x										
Behutiye et al. [38]	x														
Behutiye, Seppänen, et al. [39]	x														x
Bruel et al. [40]				x	x							x			
Bugayenko [41]			x		x			x					x		
Frattini & Frattini [42]														x	
Gérançon & Trudel [43]					x	x							x	x	x
Izhar et al. [54]						x									
Jarzębowicz & Weichbroth [44]	x														
Karolita et al. [29]											x				
Kasauli et al. [45]	x														x
Kato & Tsuda [32]						x									
Keshk et al. [55]	x														x
Khalid et al. [23]		x			x										
Kravari et al. [22]		x				x						x			
Lorch et al. [56]				x											
Mashkoor et al. [46]				x											
Medeiros et al. [47]	x														x
Mokos & Katsaros [48]		x	x	x	x				x			x			
Muzammel et al. [57]											x				
Nasir et al. [50]	x														x
Silva [31]			x		x					x			x	x	
ul Hasan & Ali Rana [26]						x									
Wang et al. [58]											x				

Documentation and specification of requirements were seen as difficult in Agile due to its focus on face to face communication and lack thorough documentation [27], [38], [39], [50], [53], [55] in addition to the shortcomings of User Stories [27], [39], [45], [47], [50], [53], [55]. User Story is the main format used for requirements in Agile methods, but they were deemed insufficient for more complex and technical projects and many studies proposed improvements to the format [47], [53], [55]. Short iteration cycles and the continuous change were also cited as problematic

characteristics for requirements [27], [38], [53]. The challenging features of Agile were seen as especially difficult for non-functional requirements [27], [38], [39], [44], [50], [53].

Formalization of requirements

Requirements can be either written in a natural or formalized language. Natural language means plain English (or any other language) sentences, like how this thesis is written. The strength of natural language is that almost anyone can create and understand it. These strengths make natural language a widely used method for creating requirements [22], [23]. The downfall of natural language requirements is their inherent vagueness and impreciseness making them ambiguous [22]–[24], [30], [31], [37], [41], [43], [48] and difficult to verify [37], [48].

Formal methods, on the other hand, have the opposite problem. Formal methods employ ontologies that strictly define the "vocabulary" of the requirements [48], [52] and semantics that specify the operations described in the requirements into a mathematically provable form [22], [40], [48]. This makes formal methods very precise, but hard to understand and create without specific training [24], [31], [37], [40], [41].

Semi-formal methods, like controlled natural languages (CNL), are a middle-ground between the two aforementioned methodologies. They aim to aim to keep the ease-of-use and understandability of natural languages, while constricting them for improved preciseness [24], [31], [37], [40], [41]. One way to do this is restricting the syntax by using patterns, templates and boilerplates, which are predefined ways to format requirements [22], [23], [31]. Semi-formal methods are, however, ultimately a compromise as they lose some of the benefits of formal methods [37], [40], [48] and are never quite as easy-to-use as natural language [53].

Requirements artifacts

Requirements artifacts are the different documents related to requirements engineering. Four artifacts were identified in this literature review, and while they are far from the only artifacts, their inclusion speaks to their prevalence in contemporary software development.

One of the primary concerns of requirements engineering is satisfying end-user needs. However, software developers do not always have ready access to end-users, which is why fictional users called personas are used [29]. Personas model the different intended users of a software and help in, for example, requirements elicitation [29], [57]. Developers can, for example ask: "Does our software satisfy the needs of our personas?", which can give valuable insights, if the personas truly represent the real end-users [29], [57]. Making sure the personas are representative of real end-users is a challenge [29], [57] and a reason why some practitioners decide no to use them [58].

Use cases and user stories are both used to represent requirements but differ in scope and level of detail. User story was the more prevalent of the two with eight studies covering it, compared to the three of use cases. There was also a difference in the types of studies that covered the two artifacts. Studies focusing on use cases discussed how to best use them [31], [42], while studies on user stories were largely about its deficiencies and how to ameliorate them [47], [53], [55].

The unified modeling language (UML) is not one type of artifact, but is instead a visual modeling language that can be used to create a multitude of different visual artifacts. The common theme that arose from the different studies was that UML, among with other visual artifacts, improve the understandability of requirements documentation [24], [31], [41], [43].

3.4 Characteristics of requirements

This section aims to identify and define the characteristics of written requirements that can then be used in the analysis in Chapter 6. Characteristics of requirements refer to the different qualities that written requirements possess such as ambiguity or consistency. Each subsection covers one characteristic with the aim to define the characteristic, discuss why it is important and how to manage it.

The defined characteristics affect the value and usefulness of requirements and thus impact the quality of the developed software. Twelve characteristics were identified and are listed in Table 3.4. Three clusters stand out when examining the number of studies that cover each characteristics. Ambiguity and non-functional have by far the most studies with 14 and 10 respectively, traceability, understandability and verifiability have 6 studies each, and the rest have 4 studies or less.

The high number of studies covering non-functional requirements is explained by it stretching the definition of a "characteristic", as it is more of a category of requirements. Other than being a broader topic, it fits with the other characteristics and is covered as such in this literature review for simplicity's sake.

The reason for the high number of studies covering ambiguity, traceability, understandability and verifiability might be due to the relative lack of complexity of covering them. With the exception of traceability, all of the characteristics can be studied by just examining the requirements themselves without the context of the project as a whole. This does not mean that studying these characteristics is in any way easy or that more context would not improve these studies, but simply that it is possible. This makes studying these characteristics more feasible; familiarizing oneself with a whole system can be really laborious and researchers might not always have access to data required for it. The one outlier of traceability does not disprove this theory, as the context required for understanding traceability in a given project

is usually well defined, meaning that, once again, an understanding of the whole system is not required.

This difference can be demonstrated by comparing ambiguity and completeness. Looking at a single requirement, one can draw conclusions about whether it is ambiguous or not. While a greater understanding of the system as a whole might inform that conclusion, it is not necessary, and an argument could be made about requirements needing to stand on their own, meaning that a greater context could even be detrimental for the analysis. In contrast, it is nigh impossible to judge the completeness of a requirement without first understanding what the system is and what it is trying to achieve.

Characteristics marked with **M1-7** in Table 3.4 are part of the quality metrics discussed in Chapter 4 and used in Chapter 6. Inclusion and exclusion rationale is elaborated in Chapter 4.

Ambiguity

The ISO/IEC/IEEE 29148 standard [21] states various characteristics that requirements should have. One such characteristic is "Unambiguous" with the following definition: "The requirement is stated in such a way so that it can be interpreted in only one way. The requirement is stated simply and is easy to understand."[21]. The value of unambiguity is clear. If the person implementing a requirement is not the same as the one writing it, any ambiguity in the requirement risks misinterpretations, resulting in an implementation that differs from the original intent. This could lead to costly reworks or even project failure [23].

What is not clear, however, is how to quantify ambiguity. After all, what is ambiguous to one, might be straightforward to another [30]. The combination of significance and elusiveness of ambiguity makes it a heavily discussed topic in the requirements engineering field. This holds true in scientific literature and among

Table 3.4: Themes concerning the characteristics of requirements. **M1-7** are part of the quality metrics in Chapter 4.

Articles	**M1 Ambiguity**	**M2 Completeness**	Consistency	Correctness	**M3 Level of detail**	Necessity	**M4 Non-functional**	**M5 Prioritization**	Technical debt	Traceability	**M6 Understandability**	**M7 Verifiability**
Alhazmi & Huang [27]							x	x		x		
Amorndettawin & Senivongse [53]							x					
Awan et al. [24]	x										x	
Bacquet et al. [37]											x	x
Behutiye et al. [38]							x		x			
Behutiye, Seppänen, et al. [39]					x		x					x
Bruel et al. [40]							x			x	x	x
Bugayenko [41]	x										x	
Gérançon & Trudel [43]	x		x							x	x	
Hagal & Saeid [25]		x	x	x		x						
Izhar et al. [54]	x											x
Jaffe [59]		x			x		x			x		
Jarzębowicz & Weichbroth [44]							x					
Kato & Tsuda [32]	x											
Khalid et al. [23]	x							x		x		
Kravari et al. [22]	x											
Kuengjai & Ramingwong [60]								x				
Melo et al. [61]	x				x		x		x			
Mokos & Katsaros [48]	x											x
Nakamichi et al. [49]	x											
Nasir et al. [50]							x					x
Ramasamy & Lim [62]	x						x					
Ramesh & Reddy [51]	x	x	x	x						x	x	
Ribeiro & Berry [30]	x											
Silva [31]	x		x									
ul Hasan & Ali Rana [26]					x							

industry professionals. Ambiguity was the most discussed topic in this literature review and in one done by Ramasamy & Lim [62]. Requirements clarity ranked third in a review of what hinders the quality RE processes by Khalid et al. [23]. In a survey conducted by Khalid et al. [23] ambiguity was ranked as the most prevalent problem in requirements elicitation and analysis by industry professionals.

Traditional ambiguity detection methods involve manual review by trained professionals. This scales poorly to larger projects as it takes a lot of time and relies on the reviewer's understanding of the project [54]. Consequently there has been much effort made towards automatic ambiguity identification in requirements. Many attempts employed natural language processing (NLP) to flag requirements that contain certain cue words that are deemed to cause ambiguity [30], [32], [49], [54]. These methods focused on different kinds of words and phrases, but what they all had in common was, that the requirements needed to be manually verified after flagging. This means that no single word inherently causes ambiguity and that context is ultimately key in defining ambiguity. For example: Nakamichi et al. [49] found that the word "sufficient" was prone to causing ambiguity, but only when, what is sufficient, was not clearly defined in the requirement.

This means that while these NLP-based methods had good recall i.e. they found most ambiguities present in the requirements, they had poor precision due to having many false positives that needed to be weeded out via manual review. Izhar et al. [54] employed machine learning to improve their framework's precision. While the model performed well in their tests, the framework's performance was depended on the rules used in NLP and training data used for machine learning [54]. This poses a threat for their method's generalizability as different contexts and domains might require specific rules and training data.

Bugayenko [41] had a much more straightforward method for measuring ambiguity: the percentage of ambiguity is the percentage of informally defined methods in a requirements specification document. Overall, it was a common view that ambiguity inherently stems from natural language [22]–[24], [30], [31], [41], [43], [48]. To this end, many formal and semi-formal methodologies were proposed. These methods aim to reduce ambiguity by having a strictly enforced form for written requirements.

This is achieved by the use of predefined semantics [22], [24], [31], [41], [43], [48] and ontologies [24], [48].

Interestingly, there seems to be a juxtaposition between ambiguity and understandability, as understandability was often described as a weakness of formal methodologies [24], [31], [40], [41]. Because formal methods follow strict rules, one has to understand said rules if they want to understand the requirements. A perfectly unambiguous requirement is of no use, if only a select few with specialized training can decipher what it means.

This links to a bigger topic of domain knowledge (DK), which plays a big part in ambiguity. The requirements engineering process involves stakeholders of different levels of DK, which can affect how they understand given requirements. Ribeiro & Berry [30] call this process, where a person reads an ambiguous requirement and unknowingly comes to a conclusion about what it means, subconscious disambiguation and it can lead to positive or negative outcomes. On one hand in a team of people of similar levels of DK, possible ambiguities might solve themselves, as every team member disambiguates them the same way [30]. On the other hand, in situations where DK varies greatly, such as in procurement where the customer has a vast DK of their field e.g. healthcare and the supplier has technical DK, subconscious disambiguation might lead to a situation where the customer and the supplier have different understandings of the requirements [30], [49].

Ambiguity prevention can take many forms per the literature review. One approach is using formal methods [22], [24], [31], [41], [43] while ensuring understandability [24], [31], [41]. More about understandability Section 3.4. Second approach is using natural language but paying close attention to cue words and phrases that are prone to causing ambiguity [30], [32], [49], [54] and dissolving those ambiguities by using quantifiable definitions and metrics [49], [54], while keeping differences in domain knowledge in mind [30], [49]. Third option, proposed by Ribeiro & Berry

[30], is giving ambiguities no special attention. They pose that most issues with ambiguities solve themselves as long as there is enough discussion among stakeholders [30].

Completeness

The ISO/IEC/IEEE 29148 standard [21] defines completeness separately for individual requirements and sets of requirements. The definition is the same in practice: "The artifact contains all necessary information to convey its purpose fully on its own.", but varies in scope.

For individual requirements the purpose to be conveyed can be a single functionality, quality metric or other part of a larger whole that requirements are used to communicate. So, if a requirement represents a single functionality, it is considered complete, if this functionality can be fully understood via this single requirement alone [21]. Whether a requirement is complete can be up to interpretation. Ramesh & Reddy [51] proposed a template for requirements in which completeness was measured by adherence to the template i.e. does the requirement contain information defined in the template.

Comparatively, for sets of requirements completeness means, that the whole entity, e.g. a piece of software, described in the requirements can be fully understood via the set of requirements alone [21]. In practice this means two things: the set has to contain all necessary requirements [25], [59] and the individual requirements have to be complete [51]. There were no concrete metrics for when a set of requirements has all the necessary requirements. Hagal & Saeid [25] suggest using UML activity diagrams to help gauging completeness and Jaffe [59] petitioned for better guidelines for completeness and pointed out that other characteristics, like robustness, are reliant on completeness. Ramesh & Reddy [51] measured a set's completeness as a percentage of individual requirements that are complete in the set.

Only three articles covered completeness. Out of them one covered it in both contexts and two in the context of a set of requirements. No article covered both aspects of a set's completeness and one article requested more guidance on the matter. This lack of research on the topic can mean that it is not seen as a serious problem to be studied or that there is a research gap. Jaffe's [59] plea for better guidance and the inconsistent coverage on the completeness of sets suggest that a research gap exists.

Consistency

Consistency is defined for a set of requirements in the ISO/IEC/IEEE 29148 standard [21]. A set is consistent if none of the requirements conflict or overlap and the units, measurements and terms used are the same in each requirement [21]. Ramesh & Reddy [51] extend this with the concept of internal consistency, which means that a requirement does not conflict with itself.

The different dimensions of inconsistency have different ramifications for a project. Conflicting and overlapping requirements can lead to costly reworks or wasted hours if they get implemented. If two conflicting requirements are implemented, at least one of them needs to be reworked or scrapped. Similarly implementing overlapping requirements means that time and effort gets wasted on the overlapping feature when it gets developed multiple times. It is, therefore, important to understand how requirements interact with one-another and with the system as a whole. Hagal & Saeid [25] suggest using dependency tables to track the relationships between requirements and activity diagrams to track how requirements link to the functionalities of the larger system. In this way consistency relates to traceability: knowing how requirements were derived will help in detecting and preventing conflicts and overlaps.

The other dimension is term consistency, which means that the terms, units and measurements are the same from one requirement to another. Inconsistent use of terms can cause confusion when one has to wonder if "Operator" in one requirement refers to the same thing as "User-Operator" in another [31]. This can also make spotting overlapping and conflicting requirements harder. If two requirements use different terms to mean the same thing, the overlap between them is more difficult to detect than, if they had used identical language. Limiting the vocabulary of terms [31] and controlling the semantics of requirements [31], [43] can help improving consistency.

Correctness

Correctness of requirements mean that they are accurate representations of stated needs that the project intends to fulfill [21], [25], [51]. An incorrect requirement provides no value to the project as it fulfills no needs, despite how, for example, unambiguous or understandable it might be.

Correctness cannot be quantified per se as it is an amalgamation of different characteristics like traceability, necessity and consistency [25], [51]. Only two articles discussed correctness, which might be explained by its patchwork nature; other articles simply discussed the characteristics that correctness amounts to.

Level of detail

Level of detail refers to the level of abstraction and granularity of requirements [26], [39], [59]. The ISO/IEC/IEEE 29148 standard [21] instructs that the level of detail of requirements should be "appropriate". This might seem frustratingly ambiguous but there is a good reason for this: one correct level of details does not exist [26], [39], [59]. The appropriate level depends on a multitude of factors like the stage of development or if the requirement is safety critical [39], [59].

Most requirements start with a high level of abstraction, i.e. low level of detail, and get refined to lower levels of abstraction [39], [59]. For example, in agile software development it is common that a requirement starts as an epic and gets refined into multiple user stories and sub-tasks [39]. This process is important because each refinement step is an opportunity to discover errors in the requirements [59]. It is also conducive to a healthy distribution of level of detail in requirements. As the higher level (low level of detail) requirements get refined to more lower level (high level of detail) requirements, the end result is that the number of requirements correlates with their level of detail. [26]

The consequences for an inappropriate level of detail vary based on if the level is too high or too low. Specifying requirements at a higher level of detail than necessary is inefficient [39] and hinders reusability [26]. Both of these issues are costly as they waste valuable development time. Requirements of too low level of detail risk being incomplete and ambiguous [61].

Necessity

The ISO/IEC/IEEE 29148 standard [21] that necessary requirements are such that their absence would cause a deficiency in the system, and that this deficiency cannot be fulfilled with other requirements. Hagal & Saeid [25] suggest tracking necessity with a table that has all requirements and their sources. Source in this case means a need of the system.

This highlights how correctness and necessity complement one another. Both relate to fulfilling system needs and both are required for those needs to be adequately represented. This also ties to traceability as it is difficult to evaluate necessity if the need that the requirement intends to fulfill is not known.

Non-functional

Non-functional requirements (NFR), sometimes also known as quality requirements, determine the different qualities of the system. NFRs govern qualities from numerous different categories like: usability, performance, reliability and security. [21] Where functional requirements dictate what the system should do, non-functional requirements tell how it should do it. For example: functional requirements determine what an "Add to cart"-button should do on a online shopping platform and NFRs determine how much latency is acceptable in the operation and what the button should look like. The line between functional and non-functional requirements, however, is not always as clear cut [59].

Even though NFRs describe no functionalities, they are important to a project's success [27], [38], [39], [50], [61]. Lack of or poor NFRs accrues technical debt [38], [61], which can lead to poor or even incorrect implementation of functionalities. This, in turn, increases maintenance costs and the need for reworks. NFRs also affect the quality of the development process as developers can use NFRs to determine when they are ready with a functionality and they support a robust validation process. [38]

There is no consensus on a single correct way to handle non-functional requirements, rather there is a myriad of different methods that are employed depending on the context [38], [39], [44], [50], [62]. Many of the same virtues apply to NFRs as do to functional requirements, e.g. unambiguity, traceability and understandability. One characteristic, however, that is more emphasized for NFRs is re-usability [50], [53]. While some functional requirements might be reused in similar projects, many NFRs are far more likely to apply to a greater variety of different projects.

Non-functional requirements are often discussed specifically in the context of agile software development. In this literature review, out of the 10 articles discussing NFRs, 6 did so in the context of agile. This can be partly explained with the general

rising interest towards agile, but additionally, NFRs are seen to be uniquely challenging for agile due to minimal documentation, focus on functionality, shortcomings of User Stories and short iteration cycles [27], [38], [44], [50], [53].

Prioritization

Requirements prioritization means ranking them by importance [27], [60]. Prioritization allows development teams to focus on the requirements that bring most value to the project [23], [27], [60]. Prioritization is crucial for the success and quality of a software project [23].

Prioritization is often associated with agile software development [27], [60]. Requirements are prioritized continuously throughout the development process to maximize stakeholder value and as a way to validate requirements [27]. However, prioritization is also important at the very first stages of development such as in procurement. Proper prioritization of requirements in RFPs gives potential suppliers a better understanding of the clients needs and results in better offers.

Technical debt

The future development costs caused by suboptimal design or implementation in software development is called technical debt (TD). While technical debt is usually associated with coding activities, it naturally extends to requirements in two ways: the debt of the requirements themselves [61] and debt caused by insufficient requirements [38]. These are clearly linked as TD of requirements will translate into general TD if it is not managed before the requirements are implemented. The rest of this chapter focuses on the TD of requirements.

The definition for TD of requirements used by Melo et al. [61] is: "the distance between the optimal requirements specification and the actual system implementation". Melo et al. [61] identified the leading causes of TD in requirements and

low level of detail and ambiguity were the most notable. Overall, while all stages of requirements engineering were represented in the causes, causes relating to documentation and specification were most prominent [61]. This result supports this thesis' focus on those stages.

Technical debt can be intentional or unintentional [61]. Unintentional debt occurs due to mistakes and inexperience and intentional debt is strategically taken to gain short-term benefits [61]. An example of intentional technical debt in requirements is leaving ambiguity in the requirements so that the details can be specified later, once the needs are better understood [30]. Intentional debt is unproblematic and often a vital part of software development, as long as it is managed and repaid [61].

Melo et al. [61] describe a lack of tools and research on TD of requirements. The latter is corroborated by this thesis as only 2 of the studies in this literature review discussed TD of requirements.

Traceability

Requirements very rarely come to be in their final form. Usually requirements start as a need to be met by the system that is then refined to the final requirement [59]. This is not a process in which objective truths are being discovered, rather it is a process that involves subjective choices, judgment calls and design decisions. The ability to trace these different stages of a given requirement's life cycle is called traceability and is often achieved with documentation [21].

Proper traceability is important in software development for two main reasons. Traceability is crucial for validation because it is difficult to validate requirements if it is not known what is the need they are trying to fulfill and what influenced the decision behind their design [21], [25]. The other avenue where traceability is key, is making changes to requirements [23]. Changing a single requirement can have knock on effects on the whole system due to dependencies between requirements.

It is, therefore, important to understand all dependencies before making changes so that they can be taken into account.

Traceability relies on documentation. For this reason, software development practices, like agile, that rely on light documentation can struggle with it [27]. Traceability matrices [23] and expansions to the User Story format [25] were suggested for documenting traceability.

Understandability

Requirements need to be understood for them to bring any value for the project. To this end, writing requirements in natural language, i.e. the way we normally talk and write, seems like the best choice, as all stakeholders are most likely proficient in communicating that way. This innate understandability of natural language is also its pitfall because the understandability stems from inherent ambiguity and impreciseness [22]–[24], [30], [31], [37], [41], [43], [48]. This ambiguity and impreciseness can be an insurmountable issue in safety critical systems.

Formal methods, on the other hand, are very unambiguous and precise but suffer from poor understandability [24], [31], [37], [40], [41]. Formal methods utilize mathematical concepts, which makes them difficult to understand for stakeholders who do not possess the required domain knowledge [24], [40]. This is especially problematic for requirements due to the variety of stakeholders that need to understand them. For example, it is not necessarily a problem if a client does not understand the software's code as long as the software performs as they expect, but they need to understand the requirements so that they can verify wether the requirements represent their needs accurately. For this reason, most research around understandability focuses on formal methods.

One way to breach the gap between natural language and formal methods is using semi-formal methods like controlled natural languages (CNL). Requirements

written in a CNL resemble natural language but adhere the structural and semantical rules of the given CNL. [24], [31], [37], [40], [41] While improving understandability, semi-formal methods compromise some of the benefits of formal methods [37], [40], [48].

Model transformations offer a way to aid understandability, while keeping all the benefits of formal languages. As the name implies, model transformations change requirements from one model to another [24], [37], [40], [41]. This is done to attain the benefits of the more understandable informal models and the precision of formal models. The transformation can be implemented in either direction: taking formal requirements and presenting them in a more understandable model like an UML diagram [24], [41] or writing requirements in an easier-to-use semi-formal language and transforming them to a formal language [37].

Verifiability

Verification refers to the activity of determining wether the system satisfies the specified requirements [21]. In other words, does the system have all the functionalities defined in the functional requirements and does it meet all the quality attributes defined in the non-functional requirements. Verifiability refers to how 'easy' a requirement is to verify.

Verification can be manual or automated [37], [40], [50]. Manual verification includes code reviews, tests and checklists [39], [50]. The chosen verification technique depends on the type and abstraction level of the requirement [39], but verification can be made easier at any level by writing the requirement in a way that ensures verifiability.

Concrete and measurable metrics enhance verifiability [54]. It is, for example, much easier to verify that an interface loads in 0,2 seconds than it is to verify that the interface is "fast and responsive". In practice any method that clearly conveys

the expected behavior of the requirement makes it more verifiable. These methods include Definition of Dones [39] and verification rules [50].

Manual verification can become unsustainable in larger projects [40] and it has the potential for human errors [37]. Automatic verification aims to rectify these issues. For requirements to be automatically verifiable, they need to be interpretable by a machine, which in practice means using formal methods [37], [48]. However, using a formal method might not guarantee automatic verifiability for a given project as defining a domain-specific ontology might be required [48].

4 Quality indicator metrics for requirements

This chapter introduces the quality indicator metrics for requirements that were used to analyze requirements in public RFPs in Chapter 6. The metrics were created based on the preceding literature review and aim to evaluate the quality of software requirements.

Section 4.1 elaborates on each selected metric, seen in Table 4.1, explaining their reason for inclusion, scope and application. Section 4.2 goes over metrics that were considered, but ultimately excluded from the final quality metrics.

Scope refers to what the unit of assessment is. Metrics that have requirement as their scope are applied to each requirement individually and metrics with the RFP scope are applied to the RFP as a whole. For example, the scope for AM1 "No ambiguous terms" is requirement, which means that each requirement is individually scored for if they contain ambiguous terms and correspondingly the scope of UN2 "Visual artifacts" is RFP, which means that the whole RFP gets a single score for if it includes visual artifacts.

Each metric has individual scoring criteria, but all scores are between 0 and 1, where 1 means that the requirement or RFP abides by the metric and 0 means that the requirement or RFP is in conflict with the metric. For metrics of requirement scope, all individual scores are summed and divided by the number of requirements

and then multiplied by 100 as seen in equation 4.1. For metrics of RFP scope, the score is multiplied by 100 as seen in equation 4.2. This gives a final score between 0 and 100 for each metric, where a score of 100 would mean that the RFP fully abides by the metric. The overall quality score of the RFP is the average of the scores of the different metrics.

$$Metric\ score\ of\ req\ scope = \frac{\sum Requirement\ scores}{Num\ of\ requirements} \times 100 \tag{4.1}$$

$$Metric\ score\ of\ RFP\ scope = RFP\ score \times 100 \tag{4.2}$$

$$Quality\ score = \frac{\sum Metric\ scores}{Num\ of\ metrics} \tag{4.3}$$

While the intention of this metric is to evaluate the quality of the requirements in RFPs, it is to be noted that the resulting score does not directly measure quality. Rather, the score measures indicators of quality. This means that the metrics do not intend to show that an RFP with a quality score of 80 is objectively better than one with the score of 65, but that the former is most likely better as it has more indicators of quality.

This is an important distinction because requirements are complex and varied and cannot be made to fit one mold of "quality". Sometimes a little ambiguity is necessary to leave room for different solutions and other times it is justified to not have a self-contained requirement. These are, however, exceptions and the metrics aim to capture the frequency of 'best practices' in the requirements.

4.1 Included metrics

The section details reason for inclusion, scope and scoring for each metric in Table 4.1.

Table 4.1: Software Requirements Quality Metrics Framework

Category	ID	Metric	Scope	Description
Ambiguity	**AM1**	No ambiguous terms	Req	Requirements avoid vague or subjective wording
	AM2	Clear sentence structure	Req	Sentences are unambiguous in regards to e.g. modifiers and references
	AM3	Single interpretation	Req	Requirement can be reasonably interpreted in only one way
Completeness	**COM**	Requirements are self-contained	Req	Requirement can be understood without reference to external requirements or documents
Level of Detail	**LOD1**	Requirements are atomic	Req	Requirements describe a single functionality or constraint
	LOD2	Sufficient specificity	Req	Requirements contain enough detail to guide implementation
Non-functional requirements	**NFR**	NFRs are defined	RFP	NFRs are explicitly defined in the requirements specification
Prioritization	**PRI**	Requirements are prioritized	RFP	Requirements have different weights
Understandability	**UN1**	Requirements are understanable	Req	Requirements understandable
	UN2	Visual artifacts	RFP	Visual artifacts like UML-diagrams are used
	UN3	Domain knowledge taken into account	RFP	Technical and domain specific language and terms are explained
Verifiability	**VER**	Requirements are verifiable	Req	Requirements can be verified via testing

Ambiguity metrics

AM1 "No ambiguous terms" evaluates requirements on their use of ambiguous terms. Ambiguous terms are words that are prone to causing ambiguity. Several studies in the literature review flagged specific words that were deemed ambiguous [30], [32], [49], [54]. While the use of an ambiguous term does not inherently cause ambiguity, it is still worthwhile to pay attention to them, as they make requirements prone to it.

AM1's scope is individual requirements and it is scored in the following manner:

$$AM1 = \begin{cases} 1 \Leftrightarrow \text{The requirement does not contain ambiguous term(s).} \\ 0 \Leftrightarrow \text{The requirement contains ambiguous term(s).} \end{cases}$$

An example of a requirement that would score a 0 from AM1 is: "The quality of the software is sufficient". The word "sufficient" is an ambiguous term as it does not really tell anything substantive about the expected quality of the software. Note, however, that if the requirement went on to explain what it meant by "sufficient", it would still score a 0 from AM1, as the metric is only interested in the use of ambiguous terms and not on the ambiguity of the requirement.

AM2 "Clear sentence structure" evaluates the requirements on the ambiguity of their sentence structures. Ambiguity stemming from an unclear sentence structure can be more difficult to spot than one stemming from ambiguous terms, making it more prone to subconscious disambiguation [30].

AM2's scope is individual requirements and it is scored in the following manner:

$$AM2 = \begin{cases} 1 \Leftrightarrow \text{The requirement's sentence structure is clear.} \\ 0 \Leftrightarrow \text{The requirement's sentence structure is unclear.} \end{cases}$$

An example of a requirement that would score a 0 from AM2 is: "Confidential data or patient information shall not be displayed to unauthorized users". It is not clear if the modifier "Confidential" refers only to data or also to patient information. In other words, does the restriction apply to all patient information or only to confidential patient information. Again, AM2 is only about if there is an unclear sentence structure and not if it leads to the requirement being ambiguous.

AM3 "Single interpretation" evaluates requirements on if they can be reasonably interpreted only in one way. Ambiguity in requirements of RFPs can cause issues throughout the procurement process. Ambiguities that are caught early cause extra work for the client and supplier alike as the ambiguities need to be ironed out, ambiguities that are caught after suppliers have left their proposals can lead to unnecessary rejections or restarts of the whole process and ambiguities that are uncaught can lead to unsuitable software and difficult contract negotiations.

AM3's scope is individual requirements and it is scored in the following manner:

$$AM3 = \begin{cases} 1 \Leftrightarrow \text{The requirement has only one reasonable interpretation.} \\ 0 \Leftrightarrow \text{The requirement has multiple reasonable interpretations.} \end{cases}$$

Example requirements given for AM1 and AM2 would score 0s for AM3 also, but AM3 is does not always match the other metrics. For example, a requirement that contains an ambiguous term that is then specified in the same requirement could differ in rating for AM1 and AM3.

Completeness metrics

COM "Requirements are self-contained" evaluates requirements on if they can be understood without reference to external requirements or documents. Self-contained requirements are easier to understand and verify as all the required information exists in the requirement itself.

COM's scope is individual requirements and it is scored in the following manner:

$$COM = \begin{cases} 1 \Leftrightarrow \text{The requirement does not rely on external information.} \\ 0 \Leftrightarrow \text{The requirement relies on external information.} \end{cases}$$

An example of a requirement that would score a 0 from COM is: "The software's data must be handled in accordance with Appendix A". There is no way to understand the requirement without reading Appendix A. Other examples include using domain specific terminology that cannot be assumed to be known or referencing functionalities or constraints of other requirements. These types of requirements can often be justified as, for example, referencing an accessibility standard is far more practical than explaining it fully in a single requirement. However, an over-

reliance of external referencing can make a requirements document difficult to read and comprehend.

Level of detail metrics

LOD1 "Requirements are atomic" evaluates requirements on if they describe only one functionality or constraint. Non-atomic requirements that describe multiple functionalities or constraints are common in RFPs as the projects are in the earliest day of development and requirements tend to be general level. The more functionalities a single requirement describes, the less detail any one functionality has. This can introduce ambiguity to the requirements.

LOD1's scope is individual requirements and it is scored in the following manner:

$$LOD1 = \begin{cases} 1 \Leftrightarrow \text{The requirement describes only one functionality or constraint.} \\ 0 \Leftrightarrow \text{The requirement describes multiple functionalities or constraints.} \end{cases}$$

An example of a requirement that would score a 0 from LOD1 is: "The system collects usage data and creates daily reports based on the data". The requirement clearly has two functionalities and separating them into two distinct requirements would make them atomic.

LOD2 "Sufficient specificity" evaluates requirements on if they have enough detail to reasonably guide implementation. Low level of detail is understandable for requirements in RFPs due to the early stage of development, but overly vague requirements might leave out details that causes the suppliers to submit unsuitable solutions.

LOD2's scope is individual requirements and it is scored in the following manner:

$$LOD2 = \begin{cases} 1 \Leftrightarrow \text{The requirement contains enough detail} \\ \quad \text{to guide implementation.} \\ 0 \Leftrightarrow \text{The requirement does not contain enough detail} \\ \quad \text{to guide implementation.} \end{cases}$$

This metric was scored by evaluating if a software engineer could be reasonably expected to start implementing the feature based on the given level of detail. For example the requirement "The system supports customer support in their daily work" would score a 0, as there is no mention of how the tools should support or really even in what. Comparatively the requirement "The system displays pending customer support tickets ranked oldest to newest" has enough detail for an engineer to start their work. Obviously the requirement still does not give the full picture of how exactly the functionality should work, but it has enough detail to guide the implementation.

Non-functional requirements metrics

NFR evaluates RFPs by if they have recognized different categories of non-functional requirements. Not only are NFRs important for a projects success [27], [38], [39], [50], [61], there are also many different categories of them. Ten of the most common categories of NFRs were identified of the collected RFPs and the RFPs were scored based on how many of the categories were explicitly recognized. Accounting for different categories in the requirements indicates that the quality of the software is a consideration for the procurer in addition to its functionality.

NFR's scope is RFP and it is scored in the following manner:

$$NFR = \frac{Num\ of\ categories\ in\ RFP}{10}$$

For a category to be counted towards explicitly recognized, it has to be named as a category in the requirements document. An RFP with no categories would score a 0 and one with integration, usability and data protection categories would score a 0,3. The ten categories of non-functional requirements identified in the RFPs were: information security, integration requirements, technical requirements, data protection, availability, accessibility, usability, performance, architecture and scalability. A service level agreement was also counted for availability as many RFPs had availability requirements in the SLA rather than in the main requirements document.

Prioritization metrics

PRI "Requirements are prioritized" evaluates RFPs by if the requirements have different weights. Prioritizing requirements guides development as the most valuable features can be implemented first. In procurement prioritization typically means having mandatory and optional requirements and possibly having different weights for the optional requirements. Weighted optional requirements give the clients better tools to evaluate proposed solutions and offers suppliers better insight into what is important for the solution.

PRI's scope is RFP and it is scored in the following manner:

$$PRI = \begin{cases} 1 \Leftrightarrow \text{Optional requirements are weighted.} \\ 0,5 \Leftrightarrow \text{Optional requirements exist, but are not weighted.} \\ 0 \Leftrightarrow \text{All requirements are mandatory.} \end{cases}$$

Requirements documents in RFPs mention whether the requirements are mandatory or optional. Some RFPs have only mandatory requirements and score a 0 from

PRI. RFPs that have optional requirements that are unweighted i.e. score no points or all score the same points score 0,5 and ones with differing points score a 1.

Understandability metrics

UN1 evaluates requirements on if they are understandable. Understandability is essential for requirements to bring any value to the project. For RFPs poor understandability can result in a redoing of the whole procurement process because supplier cannot provide a good solution if they do not understand what is being asked.

UN1's scope is individual requirements and it is scored in the following manner:

$$UN1 = \begin{cases} 1 \Leftrightarrow \text{The requirement is understandable.} \\ 0 \Leftrightarrow \text{The requirement is not understandable.} \end{cases}$$

Understandability for this metric means that the reviewer has a general idea of what the requirement means. UN1 is not about if the idea is correct, but just that the requirement is coherent and legible enough to convey some idea. This makes UN1 very subjective to score, for what is understandable to one, might incoherent to another. This, however, is not as troublesome as it might seem, because the vast majority of the requirements present in RFPs should be understandable to most people and any significant portion of an RFPs requirements not being understandable should be cause for concern regardless of the reviewer.

UN2 "Visual artifacts" evaluates RFPs on if they include visual artifacts to aid understandability. Visual artifacts mean visual representations of the requirements, procured system or its intended use cases. These include UML-diagrams of the desired architecture or Use Case diagrams. Visual artifacts aid understandability an thus help suppliers create better proposals.

UN2's scope is RFP and it is scored in the following manner:

$$UN2 = \begin{cases} 1 \Leftrightarrow \text{The RFP contains visual artifacts.} \\ 0 \Leftrightarrow \text{The RFP does not contain visual artifacts.} \end{cases}$$

UN2 scores visual artifacts simply for their existence and not for their quality or comprehensiveness. This was done to make the quality metrics more manageable to use, as UN2 is just one of twelve metrics, and because the mere existence of visual artifacts is an indicator of quality, that not all RFPs pass.

UN3 "Domain knowledge taken into account" evaluates RFPs on if domain specific language and terms are explained. Procurement efforts usually end up with experts of two or more different domains working together and they cannot be expected be familiar with one the other domains at least from the get-go. For software procurement the supplier is usually an expert in software development while the client is an expert on their respective field and the requirements are full of terms and language specific to both domains. Including an appendix that explains all the terms can help bring both parties on the same page on what the requirements mean.

UN3's scope is RFP and it is scored in the following manner:

$$UN3 = \begin{cases} 1 \Leftrightarrow \text{The RFP explains domain specific terms.} \\ 0 \Leftrightarrow \text{The RFP does not explain domain specific terms.} \end{cases}$$

UN3 is scored for the existence of a vocabulary instead of its quality for similar reasons as UN2.

Verification metrics

VER evaluates requirements on if the are verifiable. Verifying means checking if the system abides by the requirements. In software development verifying lets the team know that they are building the software right and in procurement it allows the suppliers to better evaluate if their proposal fits what the clients are looking for.

VER's scope is individual requirements and it is scored in the following manner:

$$VER = \begin{cases} 1 \Leftrightarrow \text{The requirement is verifiable.} \\ 0 \Leftrightarrow \text{The requirement is not verifiable.} \end{cases}$$

An example of a requirement that would score a 0 from VER is: "The authorization system is resilient brute force attacks". The requirement lacks any concrete ways to verify wether the system completes the requirement or not. The requirement "The system locks a user account for 30 minutes after 5 consecutive failed login attempts" has the quantifiable metrics of "30 minutes" and "5 consecutive attempts" that can be tested to verify if the system works as intended.

4.2 Excluded metrics

This section briefly discusses metrics that were considered but discarded.

Metrics pertaining to correctness and necessity were not included because scoring them would have required a thorough understanding of the projects and the needs driving them. While most RFPs contain documents that aim to explain those factors, they do not satisfactorily meet the need for two reasons. The documents greatly vary in detail and thoroughness and because of the nature of errors regarding correctness and necessity. If the requirements contain a requirement that is not necessary for solving the underlying problem, the same error is likely to be duplicated in the accompanying documents because the people creating the documents do not understand the problem.

Traceability metrics were excluded as projects that are being procured are in the earliest stages of development so there is little to trace. In addition the selected RFPs do not include any traceability documentation.

Consistency and completeness metrics could not be properly analyzed due to the sampling approach detailed in Chapter 5. For completeness, only completeness of

individual requirements could be analyzed because completeness of a set would require analyzing each requirement in the set. Consistency could in theory be analyzed within the sample, but in practice that did not yield meaningful results. Initially the analysis included three consistency metrics: "consistency of terms", "no overlapping requirements" and "no conflicting requirements", which were ultimately excluded.

A third level of detail metric was considered and scrapped. LOD3 was meant to measure if requirements were too specific as to unnecessarily restrict possible solutions. The issue with this metric was in the word "unnecessarily". Without a thorough understanding of the system, its goals and future environment, it would be nigh impossible to make such judgment calls. A possible solution was to forgo the judgment and simply score based on if the requirement restricted the "how" and not only the "what". The issue with this approach was that certain categories of requirements, like integration, would have almost all scored low, and the randomness of how many requirements of those categories were in the sample would have had a large part in deciding the final score.

5 Methodology

This thesis aims to contribute to the understanding of procurement competency in Finland by designing a framework for evaluating quality indicators in requirements and by applying the framework on public RFPs. Thus, this thesis represents applied research with a deductive research logic and the research purpose is evaluation. This thesis has elements of design research and case study.

The evaluated requirements are natural language documents that are manually transformed into quantifiable data through the quality indicator analysis. This makes the research process a mix of qualitative and quantitative with a positivist research approach.

Data collection, sampling and analysis processes are further elaborated on in the following Sections.

5.1 Data collection

The data used in this study is requirements specifications from Finnish public requests for proposals. The data is collected from RFPs found in the procurement platform HILMA. HILMA is owned by the Ministry of Finance and maintained by Hansel [63]. Hansel is a centralized procurement body owned by the Ministry of Finance and Kuntaliitto, an organization compromised of the municipalities in Finland [64]. As HILMA is owned and operated by the public administration, it falls under domestic and EU legislation governing the transparency of data in the public

sector [65]–[67]. These laws, in addition with the procurement act that demands public procurement be transparent [68], permit the use of data found in HILMA for research purposes.

The HILMA database can be filtered with Common Procurement Vocabulary (CPV) codes. CPV is a classification system for public procurement, which consists of nine digits. The first two digits define the division, which are broad categories that can be further specified with the following digits. For example, 42000000 is the code for "Industrial machinery" and it can be further specified for "Cooling and ventilation equipment" with 42500000.

The CPV division that is of interest for this thesis is 72000000: "IT services: consulting, software development, Internet and support". Division 72 contains various IT services from IT support to data classification. Of those services, this thesis is interested in software development, which can be found under, for example, 72230000: "Custom software development services". However, the search is conducted using the main division code of 72000000 due to the inconsistent CPV usage in the RFPs found in HILMA.

HILMA also offers other filtering options, like the region or scope of the RFP, but those are left empty to get a wide sampling of different projects and because they are not relevant for the outcomes of this study. The two types of RFPs that are left outside this study are dynamic purchasing systems (DSP) and expired RFPs. DSPs are not included because they are used for ongoing client-supplier relationships and handle requirements in a different manner. Expired RFPs are not included due to most of them containing deprecated links, making data gathering impossible.

The data collection process progressed in steps. First, all of the RFPs not related to software development were left out based on their name and description in HILMA. Second, only RFPs with clearly defined requirements were selected. While, RFPs with vague or non-existent requirement specifications would make for an in-

teresting research, they are out of scope for this thesis as it is about quality of the requirements.

The data collection was conducted in 9.-10.5.2026 and yielded 19 RFPs. The selected RFPs represent a varied sample of typical Finnish RFPs including both smaller domestic projects and larger EU projects. Although the RFPs are publicly available, they are anonymized in this thesis as the goal is to evaluate the quality of requirements in procurement as a whole and not to critique specific procurement organizations. The RFPs are identified as RFP1-RFP19 in this thesis.

The selected RFPs had separate requirements documents that were Excel sheets in 17 of the RFPs and PDF documents in 2.

5.2 Data sampling

The total number requirements was determined using the COUNTIF-function for requirement in Excel sheets and manual counting for requirements in PDF documents. All in all, 4511 requirements were identified accross the 19 RFPs.

Due to the high number of requirements and quality metrics that would be applied on the requirements, it was determined that the analysis would be conducted on a sampled dataset. This was done so that the number of examined RFPs could be kept high in order to form a more comprehensive understanding of the overall quality of ICT procurement. A low number of analyzed RFPs would have compromised the general applicability of this thesis.

The sampling strategy was designed to fit the research questions. **RQ2** needs the sampling to be done in a way that allows comparing the RFPs with one another and **RQ3** requires that functional and non-functional requirements can be compared. To achieve this a two level stratified sampling was conducted in which the requirements were first stratified based on the RFP and then stratified based on if requirement category.

A stratified sample means dividing the data population into separate subpopulations that are then sampled. So instead taking a random sample out of the whole pool of 4511 requirements, the requirements were divided into 38 pools (19 RFPs times 2 requirements categories) that were sampled individually.

The data was already stratified based on RFP as the requirements were in separate documents, but they also needed to be stratified based on requirements category. This was done using the categorizations present in the requirements documents. Every RFP's requirements documents were categorized and the categories were analyzed. Three main categories were identified: functional requirements, non-functional requirements and organizational requirements. Organizational requirements involve, for example, employee training and project delivery and they were excluded from the analysis due to the thesis' focus on software requirements.

Each RFP categorized the requirements in different ways. Most RFPs had a functional requirements category, but some had the functional requirements categorized under different components of the procured system. Some RFPs had a non-functional requirements category, but most used the subcategories listed in Table 6.2.

Once the relevant requirements were identified, they were collected in Excel to perform the sampling. Requirements from other Excel sheets needed no processing, but those from PDF documents needed to be cleaned to make them suitable for Excel. The data cleaning involved removing unnecessary information and line-breaks.

A sample size of 10% was selected to provide a large enough dataset for meaningful analysis, while keeping it practical for manual review. The sampling was done proportionally for the RFPs and equally for the requirements categories. This means that 10% of each RFP's requirements were included in the sample and the 10% was equally sampled from functional and non-functional requirements. If the

Table 5.1: Number of requirements in RFPs and their samples

	Total			**Sample**		
RFP	All	Func	NFR	All	Func	NFR
RFP1	257	99	134	26	13	13
RFP2	399	258	137	40	20	20
RFP3	113	14	66	11	5	6
RFP4	235	123	84	24	12	12
RFP5	339	223	68	34	17	17
RFP6	158	39	47	16	8	8
RFP7	360	182	111	36	18	18
RFP8	167	39	111	17	8	9
RFP9	144	74	71	14	7	7
RFP10	66	23	16	7	4	3
RFP11	53	18	26	5	2	3
RFP12	175	84	38	18	9	9
RFP13	621	175	38	62	31	31
RFP14	238	143	45	24	12	12
RFP15	67	36	31	7	4	3
RFP16	60	15	9	6	3	3
RFP17	243	204	39	24	12	12
RFP18	209	113	73	21	11	10
RFP19	607	264	180	61	31	30
Total	4511	2117	1324	453	227	226

10% could not be equally divided by two, the category with more requirements got the larger sample.

Table 5.1 lists all RFPs and their total number of requirements and sample sizes. As can seen from the Table, the total sample is a bit over 10% at 453 requirements due to rounding. Organizational requirements are included in the Total column, which is why functional and non-functional requirements do not amount to the amount of all requirements.

The sampling was conducted using Excel's RAND function by giving each requirement in a stratified pool a distinct random number and by ordering them based on the number. This randomizes the order of the requirements, after which the top n requirements were selected as a part of the sample, where n is the appropriate number shown in Table 5.1.

5.3 Data analysis

The total number of requirements and the number of mandatory requirements were determined using Excel's COUNTIF-function for requirements found in Excel sheets and manual counting for those in PDF documents.

The quality analysis was conducted according to the metrics defined in Chapter 4. The sampled requirements were collected in an Excel sheet where each row was a requirement and each column a metric. Every requirement was manually scored against each metric, with the exception of RFP metrics that were scored once for each RFP. Excel was then used to count the final scores for each RFP according to Equations 4.1, 4.2 and 4.3. A combined quality score was also calculated.

6 Requirements analysis results

This chapter goes over the results of the analysis conducted on the requirements of the selected RFPs. Section 6.1 gives an overview of the general characteristics of the RFPs like number of requirements. Section 6.2 presents the results of the quality indicator analysis. The results are interpreted in Chapter 7.

6.1 Overview

Table 6.1 shows the general characteristics of the chosen RFPs. As the table shows, the size of the RFPs, in terms of number of requirements varies greatly. The largest RFP (RFP13) with 621 requirements was over ten times the size of the smallest (RFP11) with 53 requirements.

The average number of requirements was 237 and the median was 209. Figures 6.1 and 6.2 show that over half (10) of the RFPs had between 100 and 300 requirements. Just over third of the RFPs fall within 100 requirements of the most common group with four RFPs having under 100 requirements and three having between 300 and 400 requirements. Two outliers had over 600 requirements.

As can be seen from Table 6.1, the majority of RFPs used the EU procedure with only four using the national procedure. Out of the five smallest RFPs, by number of requirements, four used national procedure and one used EU.

The ratio of mandatory requirements varied form 62,7% to 100%. Five RFPs had only mandatory requirements. The average ratio was 87,5% and the median was 94%.

As can be seen from Table 6.1, the number of identified non-functional requirements categories varied from 2 to 8 with the average being 5,1 and median 5. Table 6.2 shows the identified categories and the number of RFPs they were identified in. The most common NFR categories were information security and integration requirements and the least common were architecture and scalability requirements.

Table 6.1: Characteristics of the chosen RFPs

RFP	Procedure	Requirements	Mandatory	Mandatory ratio	NFR categories
RFP1	EU	257	256	99,6%	6
RFP2	EU	399	375	94,0%	7
RFP3	FI	113	94	83,2%	7
RFP4	EU	235	195	83,0%	4
RFP5	EU	339	260	76,7%	5
RFP6	EU	158	158	100%	5
RFP7	EU	360	278	77,2%	7
RFP8	EU	167	167	100%	3
RFP9	EU	144	109	75,7%	4
RFP10	EU	66	61	92,4%	2
RFP11	FI	53	53	100%	3
RFP12	EU	175	175	100%	5
RFP13	EU	621	466	75,0%	6
RFP14	EU	238	188	79,0%	4
RFP15	FI	67	42	62,7%	8
RFP16	FI	60	60	100%	2
RFP17	EU	243	234	96,3%	5
RFP18	EU	209	206	98,6%	7
RFP19	EU	607	572	94,2%	7
Total	-	4511	3949	87,5%	-
Average	-	237	208	87,5%	5,1

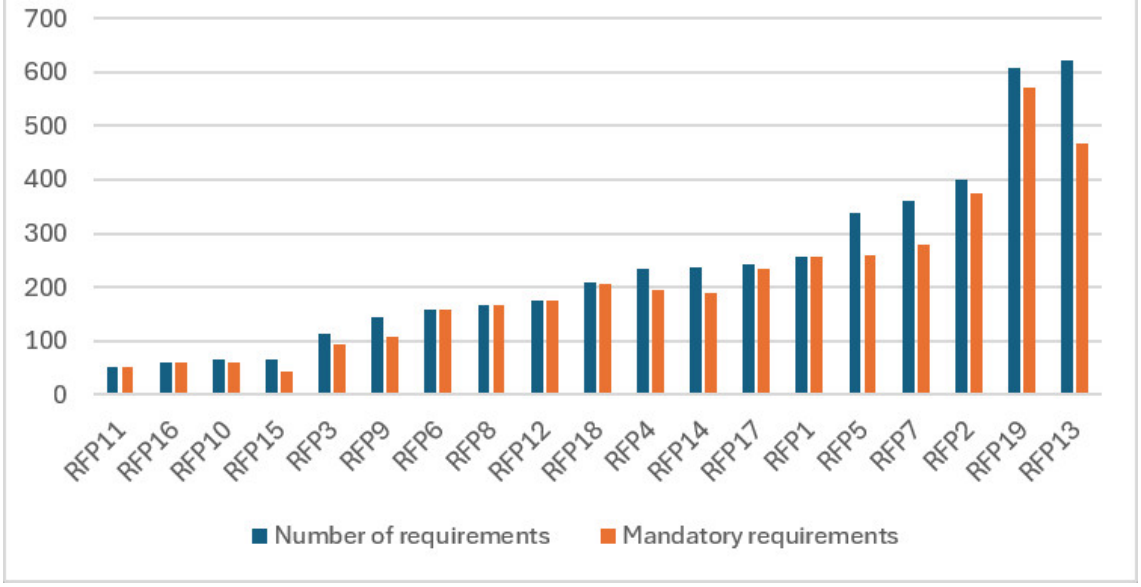

Figure 6.1: Number of requirements in RFPs

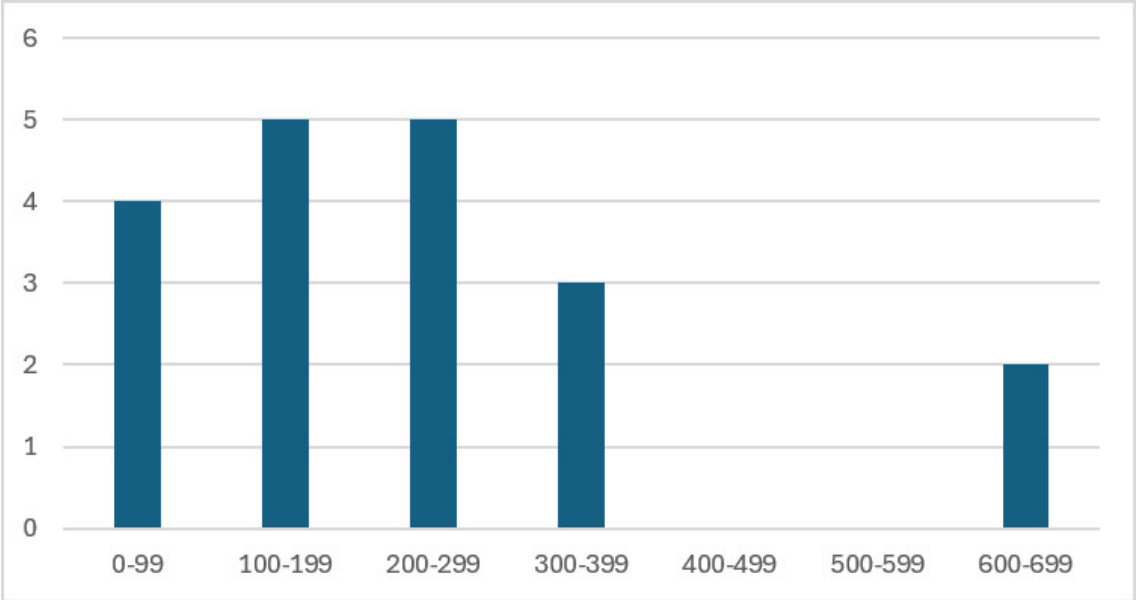

Figure 6.2: Number of RFPs by ranges requirements amounts

Table 6.2: Categories of non-functional requirements identified in RFPs, ranked by how many RFPs included them

Category	Prevalence
Information security	17
Integration requirements	15
Technical requirements	14
Data protection	13
Availability	11
Accessibility	7
Usability	7
Performance	6
Architecture	3
Scalability	1

6.2 Quality indicators

The results of the quality indicator analysis are shown in Figures 6.3, 6.4 and 6.5. The same results are color coded in Appendix A. Each column represents a category introduced in Chapter 4 and each row represents the scores the RFPs scored in the categories. All categories are scored from 0 to 100 where the higher number represents a higher number of quality indicators. The last row represents the average scores of all RFPs put together.

The quality indicator scores varied from 40,8 to 81,1. Figure 6.6 shows the quality indicator scores and normalized sizes of all RFPs sorted from smallest to largest. Normalized size means the number of requirements divided by 621, which is the number of requirements in the largest RFP. Larger RFPs tended to have better quality indicator scores.

The overall quality indicator score for all requirements was 65,6, for functional requirements it was 68,8 and for non-functional requirements it was 69,1. The best performing metric for all categories was UN1 "Requirements are understandable" and the worst performing was VER "Requirements are verifiable".

Out of the 453 requirements, 143 contained ambiguous terms i.e. scored a 0 from AM1. Out of those 143, 101 scored a 0 from AM3 "Single interpretation".

This means that, in the analyzed dataset, a requirement that had an ambiguous term, was ambiguous 70,6% of the time. The same figures for AM2 "Clear sentence structure" were 44 and 26, which gives an ambiguity rate of 59,1% for AM2. If a requirement got a 0 from both AM1 and AM2, the probability that it was ambiguous was 94,1%, with only one of the 17 requirements being unambiguous. Out of the 131 requirements that were ambiguous per AM3, 20 scored a 1 from AM1 and AM2. This means that there was a 15,3% chance that a requirement was ambiguous even if it did not contain ambiguous terms and had a clear sentence structure.

Functional requirements contained less ambiguous terms and had less requirements with multiple interpretations, but had more requirements with unclear sentence structures than non-functional requirements. Overall, functional requirements had more quality indicators when it comes to ambiguity. When all three metrics are summed together, functional requirements had a combined ambiguity score of 79,2 and non-functional requirements had 74,0.

As shown by Figure 6.7 and Table 6.3, the metrics with the largest difference between functional and non-functional requirements were AM1 and LOD2. The two metrics were also the only ones where the difference was over 10 points and overall the difference was only 0,2 points in favor of NFRs. Ambiguity, overall, favored functional requirements while level of detail fared better with NFRs. Functional requirements were slightly more self-contained (COM) and verifiable, while NFRs were a bit more understandable.

Four out of the nineteen RFPs had no prioritization of requirements. This is contradictory with Table 6.1 that shows five RFPs with 100% mandatory requirements. This discrepancy is due to RFP8 having only mandatory requirements, but also marking some of those requirements as "Possibly optional". This is why it has a mandatory rate of 100% in Table 6.1 and a PRI score of 50. Five RFPs had optional requirements with no or equal weights and ten had differing weights.

RFP	AM1	AM2	AM3	COM	LoD1	LoD2	PRI	UN1	UN2	UN3	VER	NFR	All
1	73,1	92,3	65,4	69,2	69,2	61,5	50,0	92,3	100	100	61,5	60,0	74,6
2	70,0	82,5	92,5	77,5	52,5	82,5	50,0	93,8	100	100	67,5	70,0	78,2
3	45,5	81,8	63,6	54,5	45,5	63,6	100	86,4	100	0,0	45,5	70,0	63,0
4	62,5	95,8	62,5	79,2	45,8	50,0	100	79,2	100	0,0	33,3	40,0	62,4
5	88,5	84,6	80,8	88,5	61,5	61,5	100	96,2	100	100	61,5	50,0	81,1
6	62,5	100	68,8	75,0	81,3	56,3	0,0	100	100	100	37,5	50,0	69,3
7	61,1	88,9	72,2	75,0	50,0	58,3	100	91,7	100	0,0	47,2	70,0	67,9
8	58,8	100	70,6	64,7	64,7	76,5	50,0	94,1	100	100	41,2	30,0	70,9
9	71,4	92,9	71,4	92,9	35,7	57,1	100	92,9	100	0,0	28,6	40,0	65,2
10	85,7	100	85,7	100	42,9	42,9	100	100	0,0	0,0	28,6	20,0	58,8
11	60,0	100	60,0	80,0	40,0	20,0	0,0	80,0	0,0	0,0	20,0	30,0	40,8
12	55,6	88,9	72,2	66,7	38,9	72,2	0,0	88,9	0,0	0,0	22,2	50,0	46,3
13	75,8	93,5	66,1	72,6	53,2	67,7	100	95,2	100	100	32,3	60,0	76,4
14	79,2	95,8	79,2	79,2	62,5	70,8	100	95,8	0,0	0,0	37,5	40,0	61,7
15	28,6	85,7	28,6	85,7	42,9	42,9	100	100	0,0	0,0	14,3	80,0	46,8
16	100	66,7	83,3	33,3	16,7	50,0	0,0	83,3	100	0,0	0,0	20,0	46,1
17	75,0	95,8	66,7	75,0	50,0	50,0	50,0	100	0,0	100	33,3	50,0	62,2
18	66,7	76,2	57,1	66,7	38,1	57,1	50,0	85,7	0,0	100	42,9	70,0	59,2
19	62,3	91,8	68,9	82,0	50,8	47,5	100	91,8	0,0	100	27,9	70,0	66,1
All	68,4	90,3	71,1	75,7	52,3	60,9	65,8	92,7	57,9	47,4	40,0	51,1	64,5

Figure 6.3: Quality indicator scores for all requirements

Just over half of the RFPs contained visual artifacts (UN2) and just under half contained a vocabulary of domain specific terms (UN3) with eleven and nine respectively. Six RFPs contained both and five neither.

RFP	AM1	AM2	AM3	COM	LoD1	LoD2	UN1	VER	All
1	76,9	84,6	38,5	76,9	61,5	46,2	92,3	61,5	67,3
2	85,0	80,0	90,0	70,0	55,0	80,0	92,5	75,0	78,4
3	40,0	100	60,0	80,0	20,0	40,0	100	40,0	60,0
4	83,3	100	91,7	83,3	50,0	50,0	75,0	58,3	74,0
5	94,1	82,4	88,2	82,4	58,8	58,8	94,1	70,6	78,7
6	62,5	100	75,0	62,5	75,0	50,0	100	50,0	71,9
7	77,8	77,8	77,8	77,8	44,4	50,0	83,3	55,6	68,1
8	62,5	100	75,0	62,5	62,5	62,5	87,5	62,5	71,9
9	85,7	85,7	71,4	100	14,3	42,9	85,7	14,3	62,5
10	100	100	100	100	25,0	25,0	100	0,0	68,8
11	50,0	100	50,0	100	50,0	0,0	100	0,0	56,3
12	66,7	88,9	88,9	77,8	33,3	66,7	100	33,3	69,4
13	67,7	90,3	51,6	74,2	45,2	51,6	90,3	25,8	62,1
14	83,3	100	83,3	91,7	58,3	66,7	91,7	50,0	78,1
15	50,0	75,0	50,0	75,0	50,0	50,0	100	25,0	52,8
16	100	66,7	100	66,7	33,3	100	100	0,0	70,8
17	83,3	100	66,7	91,7	41,7	25,0	100	25,0	66,7
18	63,6	54,5	45,5	72,7	18,2	36,4	72,7	27,3	48,9
19	83,9	90,3	80,6	83,9	54,8	45,2	90,3	32,3	70,2
All	77,1	87,7	72,7	79,3	48,0	52,0	90,5	43,2	68,8

Figure 6.4: Quality indicator scores for functional requirements

RFP	AM1	AM2	AM3	COM	LoD1	LoD2	UN1	VER	All
1	69,2	100	92,3	61,5	76,9	76,9	92,3	61,5	78,8
2	55,0	85,0	95,0	85,0	50,0	85,0	95,0	60,0	76,3
3	50,0	66,7	66,7	33,3	66,7	83,3	75,0	50,0	61,5
4	41,7	91,7	33,3	75,0	41,7	50,0	83,3	8,3	53,1
5	70,6	82,4	76,5	88,2	58,8	70,6	100	47,1	74,3
6	62,5	100	62,5	87,5	87,5	62,5	100	25,0	73,4
7	44,4	100	66,7	72,2	55,6	66,7	100	38,9	68,1
8	55,6	100	66,7	66,7	66,7	88,9	100	22,2	70,8
9	57,1	100	71,4	85,7	57,1	71,4	100	42,9	73,2
10	66,7	100	66,7	100	66,7	66,7	100	66,7	79,2
11	66,7	100	66,7	66,7	33,3	33,3	66,7	33,3	58,3
12	44,4	88,9	55,6	55,6	44,4	77,8	77,8	11,1	56,9
13	83,9	96,8	80,6	71,0	61,3	83,9	100	38,7	77,0
14	75,0	91,7	75,0	66,7	66,7	75,0	100	25,0	71,9
15	0,0	100	0,0	100	33,3	33,3	100	0,0	40,7
16	100	66,7	66,7	0,0	0,0	0,0	66,7	0,0	37,5
17	66,7	91,7	66,7	58,3	58,3	75,0	100	41,7	69,8
18	70,0	100	70,0	60,0	60,0	80,0	100	60,0	75,0
19	40,0	93,3	56,7	80,0	46,7	50,0	93,3	23,3	60,4
All	59,7	92,9	69,5	72,1	56,6	69,9	94,9	36,7	69,1

Figure 6.5: Quality indicator scores for non-functional requirements

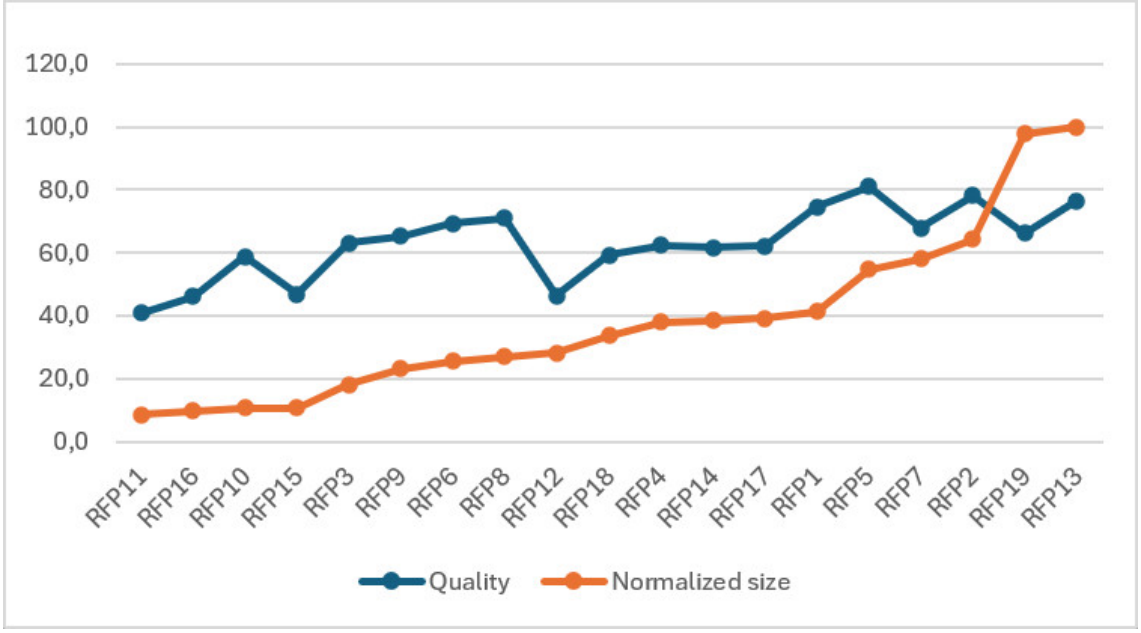

Figure 6.6: Quality of requirements and normalized number of requirements

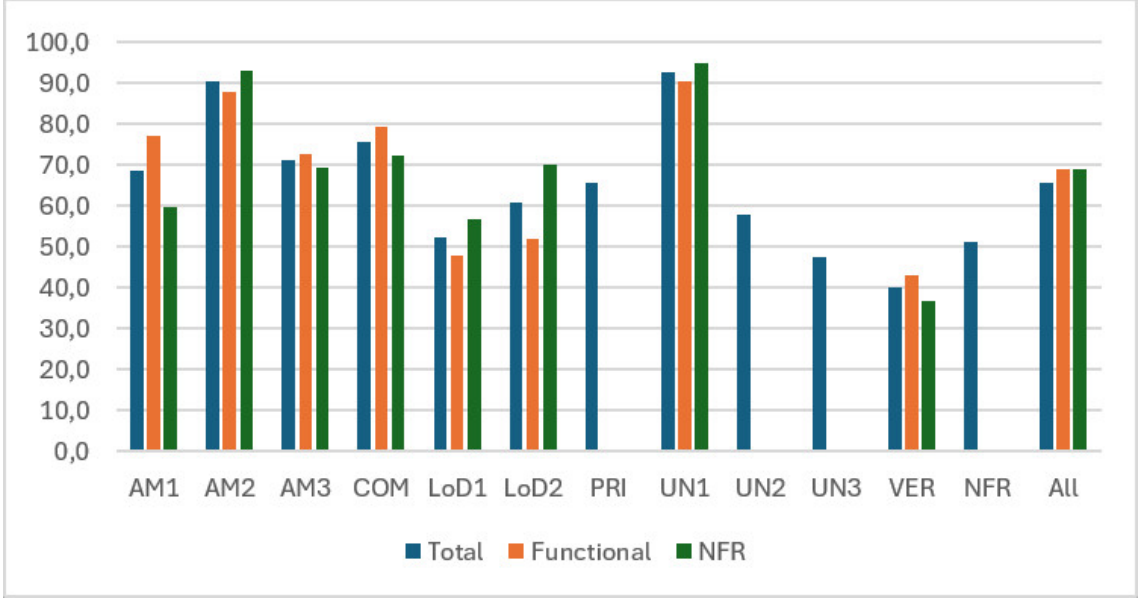

Figure 6.7: Quality indicator scores by metric

Table 6.3: Quality indicator metrics for functional and non-functional requirements and their differences

Metric	**Func**	**NFR**	**Diff**
AM1	77,1	59,7	17,4
AM2	87,7	92,9	-5,2
AM3	72,7	69,5	3,2
COM	79,3	72,1	7,2
LOD1	48,0	56,6	-8,6
LOD2	52,0	69,9	-17,9
UN1	90,5	94,9	-4,4
VER	43,2	36,7	6,5
All	68,8	69,1	-0,3

7 Discussion

7.1 Requirements quality indicator metrics

The drawbacks of the requirements quality metrics, designed in Chapter 4, were caused by the split focus of this thesis. This thesis aimed to contribute to the measuring of requirements competency by designing a framework for evaluating requirements and by measuring the current state of procurement competency. One of these goals pushed the metrics to be more comprehensive and granular, while the other pushed for breadth at the cost of depth. This can be most plainly seen in the excluded metrics discussed in Section 4.2. Entire categories of metrics had to be cut in order to accommodate the sampling approach that allowed for more RFPs to be included in the analysis.

This is not to say that these decision were made in poor judgment or that they hurt the thesis, quite the contrary. While applying the metrics to a wide range of RFPs and requirements limited their design, it was necessary to demonstrate the need for such tools. The requirements analysis showed that there indeed are differing levels of competency in Finnish ICT procurement.

However, if the metrics were to be further developed, they would benefit from a tighter focus. The two avenues where the metrics could be valuable are: a tool that procuring organizations can use to evaluate their project's requirements and a tool for en masse analysis of requirements in RFPs. The first would help create more

competent procurement artifacts and the second would create better understanding on procurement competency.

If the metrics were to be developed as a tool for procurement organizations to evaluate their own metrics, they would need to be more comprehensive. The excluded metrics discussed in Section 4.2 would be included and more than the publicly available documentation could be incorporated. The scoring system could also be expanded to allow for more expressive and actionable results. For example, metrics like PRI and UN2 that score solely based on the existence of prioritization and visual artifacts could be expanded to include the quality and validity of those elements. These additions would not have the same feasibility issues as the metrics in this thesis, because only one set of requirements would be analyzed.

The metrics descriptions and scoring criteria would also need to be expanded for them to be easier to apply for a varied user base of different levels of domain knowledge. The metric most in need of further guidance is AM2 "Clear sentence structure". The metric is based on an article by Ribeiro & Berry [30] where they introduce multiple different forms of ambiguity caused by sentence structure. The article provides multiple examples but they do not always translate well and a list of examples in Finnish would be helpful.

Unified quality metrics for requirements in Finnish public procurement could provide useful on multiple fronts. The competency of procurement documents and individuals would increase as the metrics could be used to verify the quality of requirements in RFPs as well as used for training. They would also create knowledge on the importance of different elements of procurement as the quality scores more and less successful projects could be compared.

Developing the metrics for mass analysis of requirements would require either further simplification of the metrics or automatization of the process. Manually scoring each requirement on twelve metrics, some of which require interpretation

and judgment, is unsustainable for any larger analysis. Natural language processing approaches could be developed to automate scoring of some or all metrics. This type of mass analysis of requirements in RFPs would provide a comprehensive view of the state of competency in ICT requirements, which could be then used to target training and guidelines on areas that seem lacking.

Ultimately, the first approach seems more promising. A tool that directly helps procurers create better quality requirements has more potential to positively impact procurement competency. Additionally, if the tool was widely used, it would also create data on the overall competency of procurement in Finland.

This would not replace the self-assessment tool already in use, but support it. Combining the self-assessment and requirements quality data would bring a new layer of depth to the understanding of procurement competency. Currently the self-assessment tool data is used in focusing training on areas that are ranked low, but a lack of training might not always be the reason for failed procurement.

For example, cases where the self-assessment shows high competency, but the quality indicators are low, could mean that there are organizational or procedural deficiencies that make producing quality difficult even for competent professionals. This would allow for much more targeted responses to issues, ultimately improving the quality of public procurement in Finland.

7.2 The state of ICT procurement in Finland

The requirements in Finnish RFPs varied greatly in amount and quality. Interestingly, as Figure 6.6 shows, the RFPs with more requirements also tended to have better quality scores. This validates the findings in two ways. First, a larger amount of requirements shows that the project is important and that effort has been put into defining its requirements, and the fact that the quality indicator scores align with this speak to them really measuring what they claim. Second, due to the sam-

pling strategy, some of the sample sizes were really small and it could have produced volatile results, which did not happen. This indicates that the metrics are robust and reliable even in smaller sample sizes.

Even so, the sampling strategy would have benefited from reconsideration. The idea behind it was to make sure that the RFPs had a proportional impact on the overall quality scores, but the smallest sample sizes ended up too small. This is a threat to the validity of this thesis even though the results did not show any anomalies.

As mentioned in Section 6.2, ambiguous terms (AM1) lead to ambiguity 70,6% of the time and an unclear sentence structure 59,1% of the time. This means that both factors are worth paying attention to when defining requirements, but it also shows that neither make requirements necessarily ambiguous. This means that now words or sentence need to be forbidden from requirements engineers, but that they are to be used intentionally, knowing their tendency to ambiguity and how to avoid it.

Simply avoiding unclear sentence structures and ambiguous terms is not sufficient as 15,3% of the ambiguous requirements had neither. 15,3% might seem low but considering the average RFP in this thesis had 237 requirements, 15,3% amounts to roughly 36 ambiguous requirements. 36 requirements are enough to potentially cause a lot of headaches in contract negotiations.

Overall, ambiguity was lower for functional requirements. This may show that the functionalities of the procured systems are what client organizations tend to focus on and have a more concrete understanding of, while NFRs are left on a general level. This could be explained with domain knowledge, as the clients are experts on what they want to do with the system, but not so much on how the system should do it. The exception was AM2 "Clear sentence structure" where

NFRs had functional requirements beat. Functional requirement were often more complex and especially prone to modifier ambiguity (explained in Section 4.1).

This more complex nature of functional requirements could also explain the lower score in understandability. Functional requirements also relied more on domain specific language and concepts that made some of the requirements difficult to understand.

Overall, UN1 "Requirements are understandable" was the highest scoring metric in the analysis. This is partly because the bar is quite low, simply comprehending a natural language sentence, but at the same time, the score is not to be overlooked. The requirements deal with complex system and the fact that the vast majority of the requirements were understandable, speaks to understandability being something that is considered in the procuring organizations.

NFRs scored higher on both metrics related to level of detail. Functional requirements tended to bundle multiple functionalities into single requirements, which shows in the low LOD1 "Atomic requirements" score. Though, the score was not much better for NFRs either. This is to be expected and not too alarming for requirements at the earliest levels of development. Requirements tend to start on a more abstract level and get refined further in the development process.

The starkest difference between functional and non-functional requirements in the analysis was with LOD2 "Sufficient specificity". One of the reasons for this was that NFRs referenced other documents, like usability standards, more than functional requirements did. This can also be seen in the lower COM "Requirements are self-contained" score for NFRs.

Another reason for the low specificity score for functional requirements is fear of over-specification. Many requirements described a functionality in such a general manner that there really is no way to even start developing it and this is generally how the requirements are supposed to be. Requirements in public procurement

should not restrict the possible solution too much, unless there is good reason for it. This is yet another example why these scores are not to be taken as objective metrics for quality, but as indicators of quality that need to be interpreted.

Verifiability was the lowest scoring metric for both categorizes of requirements, with NFRs being the worse performer. This is in line with a previous study from 2010 that found NFRs in Finnish RFPs to be largely unverifiable [17]. Writing requirements in a verifiable manner is difficult in public procurement, because you do not yet know all of the details of the system. Even so, verifiability was the metric with most room for improvement.

All in all, functional and non-functional requirements got nearly identical total scores with only 0,2 separating the two. This means that there does not seem to be that much of a difference in the overall quality of functional requirements and NFRs. The differences in the metric scores show, however, that the two have different pitfalls and can not be handled the same way. For example ambiguity needs to be more closely considered when defining NFRs and level of detail more of an issue for functional requirements. This is yet another example as to why this kind of evaluation is useful.

Prioritization scored the highest for the RFP-wide metrics with over half of the RFPs scoring a full points with having optional requirements with differing weights. Still, the overall mandatory requirement rate was quite high at 87,5%. This means that while prioritization is utilized in a majority of the RFPs, the utilization is still quite shallow overall.

Visual artifacts were slightly more prominent than domain specific vocabularies. This result was surprising, because visual artifacts require more time, effort and expertise to create than term explanations. This could indicate that visual artifacts are either more well known or seen as more important.

The most frequent NFR category, as shown in Table 6.2, was information security. This is hardly surprising, as it is the category with the most potential damages on poor execution. The low frequency of performance, architecture and scalability requirements could be explained by some RFPs including them under integration, technical and availability requirements. The number of usability requirements is lower than in a previous study [17], but the counting methods were different. Lehtonen et al. [17] counted requirements with specific keywords, while this thesis counted stated categories of the requirements.

The state of ICT procurement could be described as 'varied'. The RFPs varied in size and quality. The most uniform feature of the requirements was that they were presented in an Excel sheets and even that was not universal, as two RFPs had them in PDF documents. The fact that there were such big differences, not only between RFPs, but also between different metrics, shows that there is a need for more research on the competency of procurement processes and artifacts.

8 Conclusion

The goal of this thesis was to contribute to the understanding of public ICT procurement competency in Finland. Procurement competency is a topic where a lack of understanding has been identified [3] and improving this understanding has been named as one of the goals of a national initiative [4]. This was chosen as the topic of this thesis due to its societal relevance and because the means by which procurement competency knowledge is currently created is insufficient.

Currently procurement competency is measured solely with a self-assessment tool [5], which is good for measuring the perceived competency of procurement but only hints at the competency of the actual procurement processes and artifacts. A survey on industry professionals has also shown that vendors have a conflicting perception on the competency of procurers [9]. For these reasons, this thesis poses that the procurement processes and artifacts should also be measured to get a comprehensive understanding of procurement competency.

This thesis contributes to the measuring of procurement competency by proposing software requirements quality indicator metrics and by conducting a quality indicator analysis on requirements found in public RFPs. Software requirements were chosen due to their importance on a projects success and because of the poor reputation of ICT procurement in Finland.

The thesis answered the following research questions:

- **RQ1: How can the quality of requirements in public RFPs be measured?**
- **RQ2: What is the quality of requirements documentations in Finnish RFPs?**
- **RQ3: Are there differences in quality between functional and non-functional requirements in Finnish RFPs?**

To answer RQ1, a literature review was conducted to determine the factors that could be used to measure the quality of requirements in public RFPs. Three viewpoints were identified: requirements engineering stages, requirements engineering methodologies and characteristics of requirements. Ultimately, the characteristics of requirements were selected for the basis of the metrics and seven characteristics were selected that were included in the metrics.

The selected characteristics were ambiguity, completeness, level of detail, non-functional, prioritization, understandability and verifiability. Requirements quality indicator metrics were designed to measure the selected characteristics.

19 RFPs were selected and analyzed with the quality indicator metrics to answer RQ2. The analysis showed that the RFPs varied highly in size and quality indicators. The larger RFPs tended to have higher quality scores. The results show that there is a need for more research on procurement competency.

To answer RQ3, functional and non-functional requirements were compared and, while the results different between metrics, the overall scores were similar. This shows that neither category is neglected in public RFPs, but also that their quality is dependent on different factors.

References

[1] "Hankintojen arvo", Tutkihallintoa.fi. Section: Sivu, Accessed: Jul. 4, 2026. [Online]. Available: https://www.tutkihallintoa.fi/julkiset-hankinnat/hankintojen-arvo/.

[2] "Tutki hankintoja", Accessed: Jul. 4, 2026. [Online]. Available: https://tutkihankintoja.fi/.

[3] *Suomen julkisten hankintojen tilannekuva*. Ministry of Finance, Apr. 14, 2020, ISBN: 978-952-367-312-0. Accessed: Jul. 3, 2026. [Online]. Available: https://julkaisut.valtioneuvosto.fi/handle/11111/11619.

[4] "Kansallinen julkisten hankintojen strategia 2020", Sep. 9, 2020. Accessed: Jul. 3, 2026. [Online]. Available: https://julkaisut.valtioneuvosto.fi/handle/11111/11618.

[5] "Hankintaosaamisen itsearviointi", Valtiokonttori. Section: Palvelu, Accessed: Jul. 3, 2026. [Online]. Available: https://www.valtiokonttori.fi/palvelu/hankintaosaamisen-itsearviointi/.

[6] "Vaikuttavien julkisten hankintojen ohjelma Hankinta-Suomi", [Online]. Available: https://vm.fi/hankinta-suomi.

[7] "Hankinta-suomi-toimenpideohjelman strategian toteutumista tukevat toimenpiteet", [Online]. Available: https://vm.fi/hankinta-suomi.

[8] "Hankintaosaamisen seuranta", Tutkihallintoa.fi. Section: Sivu, Accessed: Jul. 3, 2026. [Online]. Available: https://www.tutkihallintoa.fi/julkiset-hankinnat/hankintaosaamisen-seuranta/.

[9] "Tietojärjestelmien hankinta Suomessa 2013", TIVIA, Accessed: Jul. 5, 2026. [Online]. Available: https://tivia.fi/toimiala/tutkimukset/tietojarjestelmien-hankinta-suomessa-2013.

[10] P. Karinkanta, *Julkiset hankinnat yrityksille käytännönläheisesti*, 1. painos., in collab. with T. Lahtinen. Helsinki: Kauppakamari, 2017, ISBN: 978-952-246-433-0.

[11] V. Luoma-aho. "Kilpailutukset | Julkisilla it-hankinnoilla on surkea maine eikä aivan syyttä: "Alan tapa on, että kilpailutukseen laitetaan huippuosaajien cv:t, ja sitten harjoittelijat tekevät työt."", Helsingin Sanomat. Section: HS Visio, Accessed: Jul. 5, 2026. [Online]. Available: https://www.hs.fi/visio/art-2000008571439.html.

[12] D. Utama and B. Purwandari, "Critical success factors of software projects: The perspective a software company from indonesia", in *2020 6th International Conference on Computing Engineering and Design (ICCED)*, Oct. 2020, pp. 1–5. DOI: 10.1109/ICCED51276.2020.9415817. Accessed: Jan. 26, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/9415817.

[13] B. Binboga and C. Altin Gumussoy, "Factors affecting agile software project success", *IEEE Access*, vol. 12, pp. 95 613–95 633, 2024, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2024.3384410. Accessed: Jan. 26, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/10488401.

[14] A. Gonçalves, J. Varajão, P. Moura Oliveira, and I. Moura, "Success factors for public sector information systems projects", *Digit. Gov.: Res. Pract.*, vol. 6,

no. 4, 53:1–53:20, Dec. 18, 2025. DOI: 10.1145/3761819. Accessed: Jan. 26, 2026. [Online]. Available: https://dl.acm.org/doi/10.1145/3761819.

[15] P. Mohagheghi and M. Jørgensen, "What contributes to the success of IT projects? success factors, challenges and lessons learned from an empirical study of software projects in the norwegian public sector", in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, May 2017, pp. 371–373. DOI: 10.1109/ICSE-C.2017.146. Accessed: Jan. 26, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/7965362.

[16] "ICT-hankintojen pelikirja ohjeistaa onnistuneisiin ICT-hankintoihin", Valtiovarainministeriö, Accessed: Jul. 3, 2026. [Online]. Available: https://vm.fi/-/ict-hankintojen-pelikirja-ohjeistaa-onnistuneisiin-ict-hankintoihin.

[17] T. Lehtonen, J. Kumpulainen, T. N. Liukkonen, and T. Jokela, "To what extent usability truly matters? a study on usability requirements in call-for-tenders of software systems issued by public authorities", in *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, ser. NordiCHI '10, New York, NY, USA: Association for Computing Machinery, Oct. 16, 2010, pp. 719–722, ISBN: 978-1-60558-934-3. DOI: 10.1145/1868914.1869013. Accessed: Mar. 26, 2026. [Online]. Available: https://dl.acm.org/doi/10.1145/1868914.1869013.

[18] "Laki julkisista hankinnoista ja käyttöoikeussopimuksista | 1397/2016 | lainsäädäntö | finlex", Accessed: Mar. 28, 2026. [Online]. Available: https://www.finlex.fi/fi/lainsaadanto/2016/1397.

[19] *Directive 2014/24/EU of the european parliament and of the council of 26 february 2014 on public procurement and repealing directive 2004/18/EC*

text with EEA relevance, Feb. 26, 2014. Accessed: Mar. 28, 2026. [Online]. Available: http://data.europa.eu/eli/dir/2014/24/oj.

[20] P. Forselius, *Onnistunut tietojärjestelmän hankinta*, 3. uud. p., in collab. with Tietotekniikan liitto. Helsinki: Talentum, 2013, ISBN: 978-952-14-2085-6.

[21] "ISO/IEC/IEEE international standard - systems and software engineering – life cycle processes – requirements engineering", *ISO/IEC/IEEE 29148:2018(E)*, pp. 1–104, Nov. 2018. DOI: 10.1109/IEEESTD.2018.8559686. Accessed: Feb. 18, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/8559686.

[22] K. Kravari, C. Antoniou, and N. Bassiliades, "Towards a requirements engineering framework based on semantics", in *Proceedings of the 24th Pan-Hellenic Conference on Informatics*, ser. PCI '20, New York, NY, USA: Association for Computing Machinery, Mar. 4, 2021, pp. 72–76, ISBN: 978-1-4503-8897-9. DOI: 10.1145/3437120.3437278. Accessed: Jan. 23, 2026. [Online]. Available: https://dl.acm.org/doi/10.1145/3437120.3437278.

[23] S. Khalid et al., "Ensuring quality in software requirement engineering process: A comparative study", *Egyptian Informatics Journal*, vol. 31, p. 100 754, Sep. 1, 2025, ISSN: 1110-8665. DOI: 10.1016/j.eij.2025.100754. Accessed: Jan. 26, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1110866525001471.

[24] M. M. Awan, F. Azam, M. W. Anwar, and Y. Rasheed, "Formal requirements specification: Z notation meta model facilitating model to model transformation", in *Proceedings of the 9th International Conference on Software and Information Engineering*, ser. ICSIE '20, New York, NY, USA: Association for Computing Machinery, Jan. 5, 2021, pp. 61–66, ISBN: 978-1-4503-7721-8.

DOI: 10.1145/3436829.3436845. Accessed: Jan. 23, 2026. [Online]. Available: https://dl.acm.org/doi/10.1145/3436829.3436845.

[25] M. A. Hagal and F. F. M. Saeid, "A framework for improving the process of discovering potential errors at the requirements engineering stage", in *2022 International Conference on Engineering & MIS (ICEMIS)*, Jul. 2022, pp. 1–7. DOI: 10.1109/ICEMIS56295.2022.9914311. Accessed: Jan. 23, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/9914311.

[26] S. A. ul Hasan and Z. Ali Rana, "Determining the level of detail of software requirements", in *2022 International Conference on Frontiers of Information Technology (FIT)*, Dec. 2022, pp. 13–17. DOI: 10.1109/FIT57066.2022.00013. Accessed: Jan. 23, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/10043069.

[27] A. Alhazmi and S. Huang, "Survey on differences of requirements engineering for traditional and agile development processes", in *2020 SoutheastCon*, ISSN: 1558-058X, Mar. 2020, pp. 1–9. DOI: 10.1109/SoutheastCon44009.2020.9397492. Accessed: Jan. 23, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/9397492.

[28] M. Arif, C. W. Mohammad, and M. Sadiq, "Software requirements modeling: A systematic literature review", in *2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON)*, Oct. 2020, pp. 194–200. DOI: 10.1109/GUCON48875.2020.9231058. Accessed: Jan. 23, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/9231058.

[29] D. Karolita, J. McIntosh, T. Kanij, J. Grundy, and H. O. Obie, "Use of personas in requirements engineering: A systematic mapping study", *Information and Software Technology*, vol. 162, p. 107 264, Oct. 1, 2023, ISSN: 0950-5849.

DOI: 10.1016/j.infsof.2023.107264. Accessed: Jan. 26, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584923001180.

[30] C. Ribeiro and D. Berry, "The prevalence and severity of persistent ambiguity in software requirements specifications: Is a special effort needed to find them?", *Science of Computer Programming*, vol. 195, p. 102 472, Sep. 1, 2020, ISSN: 0167-6423. DOI: 10.1016/j.scico.2020.102472. Accessed: Jan. 27, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167642320300824.

[31] A. R. D. Silva, "Linguistic patterns, styles, and guidelines for writing requirements specifications: Focus on use cases and scenarios", *IEEE Access*, vol. 9, pp. 143 506–143 530, 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3120004. Accessed: Jan. 23, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/9570334.

[32] T. Kato and K. Tsuda, "A method of ambiguity detection in requirement specifications by using a knowledge dictionary", *Procedia Computer Science*, Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 26th International Conference KES2022, vol. 207, pp. 1482–1489, Jan. 1, 2022, ISSN: 1877-0509. DOI: 10.1016/j.procs.2022.09.205. Accessed: Jan. 27, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050922010882.

[33] S. F. Alshareef, A. M. Maatuk, T. M. Abdelaziz, and M. Hagal, "Validation framework for aspectual requirements engineering (ValFAR)", in *Proceedings of the 6th International Conference on Engineering & MIS 2020*, ser. ICEMIS'20, New York, NY, USA: Association for Computing Machinery, Sep. 14, 2020, pp. 1–7, ISBN: 978-1-4503-7736-2. DOI: 10.1145/3410352.3410777. Accessed:

Jan. 23, 2026. [Online]. Available: https://dl.acm.org/doi/10.1145/3410352.3410777.

[34] B. A. Akinnuwesi, S. G. Fashoto, E. Mbunge, P. Mashwama, and P. A. Owate, "A SWOT analysis of software requirement validation techniques", *International Journal of Software Innovation*, vol. 10, no. 1, 2022, ISSN: 2166-7160. DOI: https://doi.org/10.4018/IJSI.297132. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2166716022000303.

[35] L. M. Amaral, F. L. Siqueira, and A. A. F. Brandão, "A survey on requirements notations in software engineering research", in *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC '22, New York, NY, USA: Association for Computing Machinery, May 6, 2022, pp. 1291–1298, ISBN: 978-1-4503-8713-2. DOI: 10.1145/3477314.3507253. Accessed: Jan. 23, 2026. [Online]. Available: https://dl.acm.org/doi/10.1145/3477314.3507253.

[36] C. Arenas, L. Garcés, M. J. C. Carmona, and C. M. Simões, "Requirements specification of a software-intensive system in the health domain: An experience report", in *Proceedings of the XIX Brazilian Symposium on Software Quality*, ser. SBQS '20, New York, NY, USA: Association for Computing Machinery, Mar. 6, 2021, pp. 1–10, ISBN: 978-1-4503-8923-5. DOI: 10.1145/3439961.3439996. Accessed: Jan. 23, 2026. [Online]. Available: https://dl.acm.org/doi/10.1145/3439961.3439996.

[37] C. Bacquet, P. Marangé, É. Bonjour, and A. Kerbrat, "Requirements structure for system requirements formal modelling, verification and validation", *IFAC-PapersOnLine*, 18th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2024, vol. 58, no. 19, pp. 289–294, Jan. 1, 2024, ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2024.09.194. Accessed: Jan. 26, 2026.

[Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896324016136.

[38] W. Behutiye, P. Rodríguez, M. Oivo, S. Aaramaa, J. Partanen, and A. Abhervé, "Towards optimal quality requirement documentation in agile software development: A multiple case study", *Journal of Systems and Software*, vol. 183, p. 111 112, Jan. 1, 2022, ISSN: 0164-1212. DOI: 10.1016/j.jss.2021.111112. Accessed: Jan. 27, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121221002090.

[39] W. Behutiye, P. Seppänen, P. Rodríguez, and M. Oivo, "Documentation of quality requirements in agile software development", in *Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '20, New York, NY, USA: Association for Computing Machinery, Apr. 17, 2020, pp. 250–259, ISBN: 978-1-4503-7731-7. DOI: 10.1145/3383219.3383245. Accessed: Jan. 23, 2026. [Online]. Available: https://dl.acm.org/doi/10.1145/3383219.3383245.

[40] J.-M. Bruel, S. Ebersold, F. Galinier, M. Mazzara, A. Naumchev, and B. Meyer, "The role of formalism in system requirements", *ACM Comput. Surv.*, vol. 54, no. 5, 93:1–93:36, May 25, 2021, ISSN: 0360-0300. DOI: 10.1145/3448975. Accessed: Jan. 23, 2026. [Online]. Available: https://dl.acm.org/doi/10.1145/3448975.

[41] Y. Bugayenko, "Combining object-oriented paradigm and controlled natural language for requirements specification", in *Proceedings of the 1st ACM SIGPLAN International Workshop on Beyond Code: No Code*, ser. BCNC 2021, New York, NY, USA: Association for Computing Machinery, Oct. 17, 2021, pp. 11–17, ISBN: 978-1-4503-9125-2. DOI: 10.1145/3486949.3486963. Accessed: Jan. 23, 2026. [Online]. Available: https://dl.acm.org/doi/10.1145/3486949.3486963.

[42] J. Frattini and A. Frattini, "Adopting use case descriptions for requirements specification: An industrial case study", in *2025 IEEE 33rd International Requirements Engineering Conference (RE)*, ISSN: 2332-6441, Sep. 2025, pp. 179–191. DOI: 10.1109/RE63999.2025.00026. Accessed: Jan. 23, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/11190096.

[43] B. Gérançon and S. Trudel, "Improving quality of software requirements by using a triplet structure", in *2022 International Conference on Computational Science and Computational Intelligence (CSCI)*, ISSN: 2769-5654, Dec. 2022, pp. 1884–1888. DOI: 10.1109/CSCI58124.2022.00339. Accessed: Jan. 23, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/10216449.

[44] A. Jarzębowicz and P. Weichbroth, "A qualitative study on non-functional requirements in agile software development", *IEEE Access*, vol. 9, pp. 40 458–40 475, 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3064424. Accessed: Jan. 23, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/9371679.

[45] R. Kasauli, E. Knauss, J. Horkoff, G. Liebel, and F. G. de Oliveira Neto, "Requirements engineering challenges and practices in large-scale agile system development", *Journal of Systems and Software*, vol. 172, p. 110 851, Feb. 1, 2021, ISSN: 0164-1212. DOI: 10.1016/j.jss.2020.110851. Accessed: Jan. 27, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121220302417.

[46] A. Mashkoor, M. Leuschel, and A. Egyed, "Validation obligations: A novel approach to check compliance between requirements and their formal specification", in *Proceedings of the 43rd International Conference on Software Engineering: New Ideas and Emerging Results*, ser. ICSE-NIER '21, Virtual Event, Spain: IEEE Press, Dec. 17, 2021, pp. 1–5, ISBN: 978-0-7381-3324-9.

DOI: 10.1109/ICSE-NIER52604.2021.00009. Accessed: Jan. 23, 2026. [Online]. Available: https://dl.acm.org/doi/10.1109/ICSE-NIER52604.2021.00009.

[47] J. Medeiros, A. Vasconcelos, C. Silva, and M. Goulão, "Requirements specification for developers in agile projects: Evaluation by two industrial case studies", *Information and Software Technology*, vol. 117, p. 106 194, Jan. 1, 2020, ISSN: 0950-5849. DOI: 10.1016/j.infsof.2019.106194. Accessed: Jan. 27, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584919302010.

[48] K. Mokos and P. Katsaros, "A survey on the formalisation of system requirements and their validation", *Array*, vol. 7, p. 100 030, Sep. 1, 2020, ISSN: 2590-0056. DOI: 10.1016/j.array.2020.100030. Accessed: Mar. 10, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590005620300151.

[49] T. Nakamichi, K. Nishiura, M. Sasakura, and A. Monden, "An empirical study on ambiguous words in software requirements specifications of local government and library systems", in *2024 IEEE/ACIS 22nd International Conference on Software Engineering Research, Management and Applications (SERA)*, ISSN: 2770-8209, May 2024, pp. 206–211. DOI: 10.1109/SERA61261.2024.10685614. Accessed: Jan. 23, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/10685614.

[50] S. Nasir, E. Guerra, L. Zaina, and J. Melegati, "An exploratory study about non-functional requirements documentation practices in agile teams", in *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC '23, New York, NY, USA: Association for Computing Machinery, Jun. 7, 2023, pp. 1009–1017, ISBN: 978-1-4503-9517-5. DOI: 10.1145/3555776.

3577605. Accessed: Jan. 23, 2026. [Online]. Available: https://dl.acm.org/doi/10.1145/3555776.3577605.

[51] M. R. R. Ramesh and C. S. Reddy, "Metrics for software requirements specification quality quantification", *Computers and Electrical Engineering*, vol. 96, p. 107 445, Dec. 1, 2021, ISSN: 0045-7906. DOI: 10.1016/j.compeleceng.2021.107445. Accessed: Jan. 27, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790621004043.

[52] H. Alrumaih, A. Mirza, and H. Alsalamah, "Domain ontology for requirements classification in requirements engineering context", *IEEE Access*, vol. 8, pp. 89 899–89 908, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2993838. Accessed: Jan. 23, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/9091131.

[53] M. Amorndettawin and T. Senivongse, "Non-functional requirement patterns for agile software development", in *Proceedings of the 2019 3rd International Conference on Software and e-Business*, ser. ICSEB '19, New York, NY, USA: Association for Computing Machinery, Jan. 28, 2020, pp. 66–74, ISBN: 978-1-4503-7649-5. DOI: 10.1145/3374549.3374561. Accessed: Jan. 23, 2026. [Online]. Available: https://dl.acm.org/doi/10.1145/3374549.3374561.

[54] R. Izhar, S. N. Bhatti, and S. A. Alharthi, "Bridging precision and complexity: A novel machine learning approach for ambiguity detection in software requirements", *IEEE Access*, vol. 13, pp. 12 014–12 031, 2025, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2025.3529943. Accessed: Jan. 23, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/10843220.

[55] N. Keshk, M. El-Ramly, and A. Salah, "A proposal for enhancing agile requirements engineering with prototyping and enriched user stories", in *Proceedings of the Federated Africa and Middle East Conference on Software En-*

gineering, ser. FAMECSE '22, New York, NY, USA: Association for Computing Machinery, Jul. 17, 2022, pp. 59–63, ISBN: 978-1-4503-9663-9. DOI: 10.1145/3531056.3542773. Accessed: Jan. 23, 2026. [Online]. Available: https://dl.acm.org/doi/10.1145/3531056.3542773.

[56] R. Lorch et al., "Formal methods in requirements engineering: Survey and future directions", in *Proceedings of the 2024 IEEE/ACM 12th International Conference on Formal Methods in Software Engineering (FormaliSE)*, ser. FormaliSE '24, New York, NY, USA: Association for Computing Machinery, Jun. 6, 2024, pp. 88–99, ISBN: 979-8-4007-0589-2. DOI: 10.1145/3644033.3644373. Accessed: Jan. 23, 2026. [Online]. Available: https://dl.acm.org/doi/10.1145/3644033.3644373.

[57] C. S. Muzammel, M. Spichkova, and J. Harland, "Towards using personas in requirements engineering: What has been changed recently?", in *2025 IEEE 33rd International Requirements Engineering Conference Workshops (REW)*, ISSN: 2770-6834, Sep. 2025, pp. 115–122. DOI: 10.1109/REW66121.2025.00019. Accessed: Jan. 23, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/11190262.

[58] Y. Wang et al., "Who uses personas in requirements engineering: The practitioners' perspective", *Information and Software Technology*, vol. 178, p. 107 609, Feb. 1, 2025, ISSN: 0950-5849. DOI: 10.1016/j.infsof.2024.107609. Accessed: Jan. 26, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584924002143.

[59] M. Jaffe, "Levels of requirements, robustness, unicorns, and other semi-mythical creatures in the requirements engineering bestiary: Why "types" of software requirements are often misleading", in *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, ISSN: 2155-7209, Oct. 2021, pp. 1–8.

DOI: 10.1109/DASC52595.2021.9594323. Accessed: Jan. 23, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/9594323.

[60] Y. Kuengjai and L. Ramingwong, "A pilot study of requirement prioritization techniques in agile software development.", in *Proceedings of the 4th International Conference on Computer Science and Software Engineering*, ser. CSSE '21, New York, NY, USA: Association for Computing Machinery, Dec. 20, 2021, pp. 146–151, ISBN: 978-1-4503-9067-5. DOI: 10.1145/3494885.3494912. Accessed: Jan. 23, 2026. [Online]. Available: https://dl.acm.org/doi/10.1145/3494885.3494912.

[61] A. Melo, R. Fagundes, V. Lenarduzzi, and W. B. Santos, "Identification and measurement of requirements technical debt in software development: A systematic literature review", *Journal of Systems and Software*, vol. 194, p. 111 483, Dec. 1, 2022, ISSN: 0164-1212. DOI: 10.1016/j.jss.2022.111483. Accessed: Jan. 27, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121222001650.

[62] R. Ramasamy and T. Y. Lim, "Exploring non-functional requirements in requirements engineering: A systematic review", in *2024 IEEE International Conference on Computing (ICOCO)*, Dec. 2024, pp. 491–496. DOI: 10.1109/ICOCO62848.2024.10928262. Accessed: Jan. 23, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/10928262.

[63] "HILMA | Hankinnat", Accessed: Apr. 13, 2026. [Online]. Available: https://www.hankinnat.fi/eu-hankinta/ilmoittaminen/hilma.

[64] "Hansel - hansel in brief", Accessed: Apr. 13, 2026. [Online]. Available: https://www.hansel.fi/en/about-us/hansel-brief/.

[65] "Laki viranomaisten toiminnan julkisuudesta | 621/1999 | lainsäädäntö | finlex", Accessed: Apr. 13, 2026. [Online]. Available: https://www.finlex.fi/fi/lainsaadanto/1999/621.

[66] "Laki julkisen hallinnon tiedonhallinnasta | 906/2019 | lainsäädäntö | finlex", Accessed: Apr. 13, 2026. [Online]. Available: https://www.finlex.fi/fi/lainsaadanto/2019/906.

[67] *Directive (EU) 2019/1024 of the European Parliament and of the Council of 20 June 2019 on open data and the re-use of public sector information (recast)*, Jun. 20, 2019. Accessed: Apr. 13, 2026. [Online]. Available: http://data.europa.eu/eli/dir/2019/1024/oj.

[68] "Laki julkisista hankinnoista ja käyttöoikeussopimuksista | 1397/2016 | lainsäädäntö | finlex", Accessed: Apr. 13, 2026. [Online]. Available: https://www.finlex.fi/fi/lainsaadanto/2016/1397.

Appendix A Color coded quality indicator results

RFP	AM1	AM2	AM3	COM	LoD1	LoD2	PRI	UN1	UN2	UN3	VER	NFR	All
1	73,1	92,3	65,4	69,2	69,2	61,5	50,0	92,3	100	100	61,5	60,0	74,6
2	70,0	82,5	92,5	77,5	52,5	82,5	50,0	93,8	100	100	67,5	70,0	78,2
3	45,5	81,8	63,6	54,5	45,5	63,6	100	86,4	100	0,0	45,5	70,0	63,0
4	62,5	95,8	62,5	79,2	45,8	50,0	100	79,2	100	0,0	33,3	40,0	62,4
5	88,5	84,6	80,8	88,5	61,5	61,5	100	96,2	100	100	61,5	50,0	81,1
6	62,5	100	68,8	75,0	81,3	56,3	0,0	100	100	100	37,5	50,0	69,3
7	61,1	88,9	72,2	75,0	50,0	58,3	100	91,7	100	0,0	47,2	70,0	67,9
8	58,8	100	70,6	64,7	64,7	76,5	50,0	94,1	100	100	41,2	30,0	70,9
9	71,4	92,9	71,4	92,9	35,7	57,1	100	92,9	100	0,0	28,6	40,0	65,2
10	85,7	100	85,7	100	42,9	42,9	100	100	0,0	0,0	28,6	20,0	58,8
11	60,0	100	60,0	80,0	40,0	20,0	0,0	80,0	0,0	0,0	20,0	30,0	40,8
12	55,6	88,9	72,2	66,7	38,9	72,2	0,0	88,9	0,0	0,0	22,2	50,0	46,3
13	75,8	93,5	66,1	72,6	53,2	67,7	100	95,2	100	100	32,3	60,0	76,4
14	79,2	95,8	79,2	79,2	62,5	70,8	100	95,8	0,0	0,0	37,5	40,0	61,7
15	28,6	85,7	28,6	85,7	42,9	42,9	100	100	0,0	0,0	14,3	80,0	46,8
16	100	66,7	83,3	33,3	16,7	50,0	0,0	83,3	100	0,0	0,0	20,0	46,1
17	75,0	95,8	66,7	75,0	50,0	50,0	50,0	100	0,0	100	33,3	50,0	62,2
18	66,7	76,2	57,1	66,7	38,1	57,1	50,0	85,7	0,0	100	42,9	70,0	59,2
19	62,3	91,8	68,9	82,0	50,8	47,5	100	91,8	0,0	100	27,9	70,0	66,1
All	68,4	90,3	71,1	75,7	52,3	60,9	65,8	92,7	57,9	47,4	40,0	51,1	64,5

Figure A.1: Color coded quality indicator scores for all requirements

RFP	AM1	AM2	AM3	COM	LoD1	LoD2	UN1	VER	All
1	76,9	84,6	38,5	76,9	61,5	46,2	92,3	61,5	67,3
2	85,0	80,0	90,0	70,0	55,0	80,0	92,5	75,0	78,4
3	40,0	100	60,0	80,0	20,0	40,0	100	40,0	60,0
4	83,3	100	91,7	83,3	50,0	50,0	75,0	58,3	74,0
5	94,1	82,4	88,2	82,4	58,8	58,8	94,1	70,6	78,7
6	62,5	100	75,0	62,5	75,0	50,0	100	50,0	71,9
7	77,8	77,8	77,8	77,8	44,4	50,0	83,3	55,6	68,1
8	62,5	100	75,0	62,5	62,5	62,5	87,5	62,5	71,9
9	85,7	85,7	71,4	100	14,3	42,9	85,7	14,3	62,5
10	100	100	100	100	25,0	25,0	100	0,0	68,8
11	50,0	100	50,0	100	50,0	0,0	100	0,0	56,3
12	66,7	88,9	88,9	77,8	33,3	66,7	100	33,3	69,4
13	67,7	90,3	51,6	74,2	45,2	51,6	90,3	25,8	62,1
14	83,3	100	83,3	91,7	58,3	66,7	91,7	50,0	78,1
15	50,0	75,0	50,0	75,0	50,0	50,0	100	25,0	52,8
16	100	66,7	100	66,7	33,3	100	100	0,0	70,8
17	83,3	100	66,7	91,7	41,7	25,0	100	25,0	66,7
18	63,6	54,5	45,5	72,7	18,2	36,4	72,7	27,3	48,9
19	83,9	90,3	80,6	83,9	54,8	45,2	90,3	32,3	70,2
All	77,1	87,7	72,7	79,3	48,0	52,0	90,5	43,2	68,8

Figure A.2: Color coded quality indicator scores for functional requirements

RFP	AM1	AM2	AM3	COM	LoD1	LoD2	UN1	VER	All
1	69,2	100	92,3	61,5	76,9	76,9	92,3	61,5	78,8
2	55,0	85,0	95,0	85,0	50,0	85,0	95,0	60,0	76,3
3	50,0	66,7	66,7	33,3	66,7	83,3	75,0	50,0	61,5
4	41,7	91,7	33,3	75,0	41,7	50,0	83,3	8,3	53,1
5	70,6	82,4	76,5	88,2	58,8	70,6	100	47,1	74,3
6	62,5	100	62,5	87,5	87,5	62,5	100	25,0	73,4
7	44,4	100	66,7	72,2	55,6	66,7	100	38,9	68,1
8	55,6	100	66,7	66,7	66,7	88,9	100	22,2	70,8
9	57,1	100	71,4	85,7	57,1	71,4	100	42,9	73,2
10	66,7	100	66,7	100	66,7	66,7	100	66,7	79,2
11	66,7	100	66,7	66,7	33,3	33,3	66,7	33,3	58,3
12	44,4	88,9	55,6	55,6	44,4	77,8	77,8	11,1	56,9
13	83,9	96,8	80,6	71,0	61,3	83,9	100	38,7	77,0
14	75,0	91,7	75,0	66,7	66,7	75,0	100	25,0	71,9
15	0,0	100	0,0	100	33,3	33,3	100	0,0	40,7
16	100	66,7	66,7	0,0	0,0	0,0	66,7	0,0	37,5
17	66,7	91,7	66,7	58,3	58,3	75,0	100	41,7	69,8
18	70,0	100	70,0	60,0	60,0	80,0	100	60,0	75,0
19	40,0	93,3	56,7	80,0	46,7	50,0	93,3	23,3	60,4
All	59,7	92,9	69,5	72,1	56,6	69,9	94,9	36,7	69,1

Figure A.3: Color coded quality indicator scores for non-functional requirements