

UNIVERSITY OF TWENTE

MASTER'S THESIS

Design and validation of a Software Requirements Specification evaluation checklist

Author:

Martin DE LAAT

Supervisors:

dr. M. Daneva (University of Twente)

Prof. N. Madhavji (University of Western Ontario)

Graduation Committee:

dr. M. Daneva (University of Twente)

dr. Faiza Allah Bukhsh (University of Twente)

*A thesis submitted in fulfilment of the requirements
for the degree of Master of Science*

Services and Cybersecurity (SCS)

Electrical Engineering, Mathematics and Computers Science

7th July 2019

Declaration of Authorship

I, Martin DE LAAT, declare that this thesis titled, 'Design and validation of a Software Requirements Specification evaluation checklist' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UNIVERSITY OF TWENTE

Abstract

Electrical Engineering, Mathematics and Computer Sciences

Master of Science

Design and validation of a Software Requirements Specification evaluation checklist

by Martin DE LAAT

The quality of a Software Requirements Specification (SRS) can be a major contributing factor to the overall success of the software project, possibly more so for (Government) IT procurement and/or when tendering is involved. SRS evaluation via inspection techniques can be an effective method for quality assurance purposes. An artefact is developed to support the RE practitioner during the Software Requirements Specification evaluation process with the goal to improve both the process and the outcome of the evaluation. The artefact is developed in an iterative manner based on results from a literature study, feedback from supervisors Prof. Nazim Madhavji and Dr. Maya Daneva and the results from an interview sessions with RE practitioners. The format chosen is that of a checklist combined with a handbook consisting of supportive materials to be used during the SRS evaluation process. The artefact is validated using a mixed method approach during a Live Study [21] at the REFSQ 2018 conference in Utrecht. The artefact is found to be a comprehensive tool in evaluating SRSs and, based on a series of 8 quality attributes, is regarded to be of good quality. Indicated challenges are its usability, clarity and applicability to a variety of contexts. Application of the artefact in practice and analysing its effects is seen as the next step in understanding the effect of applying the artefact under different contexts, enhancing the artefact accordingly and proving its merit in practice.

Preface

Before you lies the thesis ‘Design and validation of a Software Requirements Specification evaluation checklist’. This research is an attempt to contribute to better software quality by making the job of evaluating (and potentially writing-) requirements specification a more manageable and tangible experience, hopefully improving the quality of the evaluation and the resulting software product alike. The opportunity provided by Prof. Nazim Madhavji, that is to develop a practical tool in collaboration with a renowned software consultancy firm, seemed like the perfect opportunity to combine my intrinsic goals and values with the graduation requirements of the university. This thesis marks the end of this research and my journey at the university of Twente, and is the final milestone towards completing the Masters degree Computer Science: Information and Software Engineering.

The unique environment of the university has allowed me to not only discover my interests and develop my technical background but also to discover and develop myself as a person. It has been a place with not only academic, but also social possibilities. After finishing the bachelor Business & IT, it became clear to me that I enjoy being in a position that allows me to provide value to people and found I could make this a reality bringing stakeholders and software developers closer together. To properly do so, I felt I needed to improve my social skills as well as my technical skills. The university has allowed me to combine my studies with achieving valuable work experience via various positions at the university and my work at Nedap, to gain management experience as chairman of the board of the climbing association TSAC and to gain valuable experience as a climbing instructor.

Now, after a long journey with many ups and downs (figuratively and literally), the time has come to fully transcend into my role as a contributing member to society.

Po: Maybe I should just quit and go back to making noodles.

Oogway: Quit, don’t quit? Noodles, don’t noodles? You are too concerned about what was and what will be. There is a saying: yesterday is history, tomorrow is a mystery, but today is a gift. That is why it is called the present.

Po & Master Oogway, Kung Fu Panda

I would like to express my immense gratitude to my project supervisors Maya Daneva and Nazim Madhavji, as well as my parents and my girlfriend. It is not an understatement to say that without their continuous support and understanding my graduation would not have been a possibility. Furthermore, I’d like to thank Faiza Allah Buksh for offering to be part of the graduation committee and Lilian Spijker, Anne-Marie Hoogland, Barbara Spikker-Sieverink and Marten van Sinderen for their guidance and help in navigating the regulatory waters. Finally, I’d like to thank Nedap for their flexibility and investment in me.

Martin de Laat
Enschede, July 2019

Contents

Declaration of Authorship	iii
Abstract	v
Preface	vii
1 Introduction	1
1.1 About the author	1
1.2 Research initiation	1
1.3 Motivation and Significance	1
1.4 Scope of study	2
1.5 Research Objective	2
1.6 Approach	2
1.7 Results	2
1.8 Structure of the report	2
2 Research Design	5
2.1 Research Methods	5
2.1.1 Method Overview	5
2.1.2 Design Science and the design cycle	6
2.2 Research Goals	8
2.2.1 Technical Research Problem Statement	8
2.2.2 Design science Research Goals	8
3 Investigation	11
3.1 The Social context	11
3.1.1 Problem context	11
3.1.2 Social context goals	11
3.1.3 Requirements	13
3.2 The Knowledge context	13
3.2.1 Requirements in the product life cycle[KQ 1]	13
3.2.2 The importance of the requirements specification	15
3.2.3 Context-dependency	16
3.2.4 SRS quality [KQ 2]	16
3.2.5 SRS challenges [KQ 3]	17
3.2.6 SRS evaluation [KQ 4]	17
3.2.7 SRS inspection techniques [KQ 5]	18
3.2.8 Checklist design [KQ 6]	19
3.2.9 Checklist validation [KQ 7]	20

4	Instrument Design	21
4.1	Instrument versioning overview	21
4.2	Design process summary	21
4.3	Instrument structure	22
4.3.1	Anticipated flow of the evaluation process	22
4.3.2	Division of sections	23
4.4	Design decisions	23
4.4.1	Format	24
4.4.2	Variability and acknowledging the reviewer's expertise	24
4.4.3	Rating scheme and judgement	24
5	Semi-structured interviews with expert practitioners	27
5.1	Introduction	27
5.1.1	Research goals and Research Questions	27
5.2	Methods	28
5.2.1	Interview Design	28
5.2.2	Logistics	28
5.2.3	Materials	29
5.2.4	Participants	29
5.2.5	Procedure	29
5.2.6	Analysis	30
5.3	Results	32
5.3.1	Participants' background	32
5.3.2	RG 1: To better understand the Problem Context	33
5.3.3	RG 2: To assess and improve the quality of the checklist	38
5.3.4	Answering the Research Questions	39
5.4	Discussion & Conclusion	42
5.4.1	Trade-off between comprehensiveness and usability	42
5.5	Changes incorporated to the instrument resulting the interview sessions	44
6	Live Study Validation at REFSQ'18	45
6.1	Background	45
6.1.1	About REFSQ	45
6.2	Study design	45
6.2.1	Research objectives and research questions	45
6.2.2	Methodology	46
6.2.3	Logistics	46
6.2.4	Course of the live study	47
6.3	Results and analysis	48
6.3.1	Gathering of the results and handling of the data	48
6.3.2	Categorisation of collected data and their purpose	48
6.3.3	Usage and purpose of the data collected	49
6.3.4	Participants' experience within the domain	49
6.3.5	Quality of the Instrument	49
6.3.6	Anticipated effect of instrument application in the problem context	52
6.3.7	Improving the instrument	52
6.4	Discussion	53
6.5	Conclusion	55
6.5.1	RQ1: What is the quality of the artefact? (KQ 8)	55

6.5.2	RQ2: What is the anticipated effect of applying the artefact in practice? (PG 1)	55
6.6	Changes to the instrument	56
7	Recommendations, conclusion & future work	59
7.1	Reflection on the research process	59
7.2	Challenges faced during the research	59
7.3	Recommendations for practitioners	60
7.3.1	State of the checklist	60
7.3.2	Adjusting the checklist to context	60
7.3.3	Required expertise to use the checklist	61
7.4	Future research	61
A	SRS Evaluation Checklist	63
A.1	Evaluation sheet	64
A.2	Handbook	65
B	Interview sessions - analysis	77
B.1	Interview Scheme	77
B.2	Code Skeletons	78
B.2.1	High-level	78
B.2.2	Complete	79
B.3	Code Tree	80
B.3.1	Problem Investigation	80
B.3.2	Instrument Validation	81
B.4	Code Table	82
C	Live Study Materials	87
C.1	Checklist sheet versions	87
C.1.1	Version 1	88
C.1.2	Version 2	89
C.1.3	Version 3	90
C.1.4	Version 4	91
C.2	Post-Use Questionnaire	92
D	Live Study results	93
D.1	Checklist sheet fault detection	93
	Bibliography	97

List of Figures

2.1	A framework for design science	6
2.2	The Engineering Cycle	6
2.3	Goal structure of a design science research project	8
3.1	Requirements in the product life cycle	13
5.1	Positively perceived content during the interview sessions	40
5.2	Negatively perceived content during the interview sessions	41
5.3	Suggested improvements during the interview sessions	41
6.1	Distribution of participants' indicated experience measured in years	49
6.2	Box plot of the results displayed in Table 6.1	50
6.3	Bubble plot of the net strength and mean score	51
6.4	Box plot of the results displayed in Table 6.3	52

List of Tables

5.1	Interview details	29
5.2	Materials used during the interview sessions	29
6.1	Results of Critical Feedback Survey questions set 1-a	50
6.2	Net strength score	51
6.3	Results of Critical Feedback Survey questions set 2	52
6.4	Top 10 checks with most indicated faults	53
6.5	Recommendations from the Questionnaire	57

List of Abbreviations

Comlist	Criteria Of Merit checklist.
DBR	Default- Based Reading
FR	Functional Requirement
NFR	Non- Functional Requirement
PBR	Process- Based Reading
RE	Requirements Engingeering
RFP	Request For Proposal
SIG	Software Improvement Group
SRS	Software Requirements Specification
UBR	Usage- Based Reading
WMU	Western Michigan University

Chapter 1

Introduction

1.1 About the author

The researcher, Martin de Laat, has a bachelor degree in *Business & IT* and this thesis is the final milestone towards completing the Master *Computer Science* with sub-track *Information and Software Engineering* at the University of Twente in The Netherlands. Next to his studies the author has over over two years experience as Product Developer / Product Owner / Software Developer in an agile work environment at Nedap Health care¹ and developing SaaS solutions in the healthcare domain.

1.2 Research initiation

This research started as a research project for the course *Advanced Requirements Engineering* at the University of Twente. This project was initiated by -then visiting Professor- Nazim Madhavji in collaboration with the Software Improvement Group (SIG), a software consultancy firm located in Amsterdam, the Netherlands. The goal was simple: to design an instrument to help their expert practitioners in evaluating Software Requirement Specifications. The company wished to improve the cost and quality of their evaluations whilst still leaning on the experience of their experts to set them apart from competitors. After initial meeting and discussions it was realised that the scope of this project went beyond the scope of the course and the decision was made by the use this research as part of the Master's graduation project.

1.3 Motivation and Significance

The author is interested in the field of Requirements Engineering and the challenges that it faces with the industry's embracement of agile practices. Idealistically speaking, better communication between the software developer and the users should make parts of the extensive Requirements Engineering phase unnecessary. The researcher is under the opinion that preferably requirements should only be written down when they serve a purpose and are going to be used. In many scenarios, such as governmental projects, public contracts and tenders, specifications and intermediary parties are a 'necessary evil' to convey system requirements and contextual information between system beneficiaries and builders. The quality of these specifications is an important factor in making sure not only that the right product is build, but also that it is built right. Writing these specifications and making sure they are of sufficient quality is a very time-consuming process. Chances are that stakeholders

¹<https://nedap.com/>

do not have the right experience in determining whether their demands are correctly written down, and the parties responsible for writing and interpreting the requirements often miss the domain knowledge to make sure that what they're building corresponds to the stakeholders' actual needs. By creating an instrument that will hopefully not only lessen the burden of conducting the evaluation process but also make it more accessible to non-professionals, the researcher hopes to shorten the bridge between domain experts and technical experts.

1.4 Scope of study

The scope of the study is limited to the design and validation of an instrument to assist a reviewer with the Software Requirements Validation process. The instrument has not been evaluated in practice.

1.5 Research Objective

The objective of the research is to design and validate an artefact to be used during-, and that will have a positive effect on the Software Requirements Specification evaluation process. The Technical Research Problem Statement and further sub-goals are described in section 2.2.

1.6 Approach

Following the Design Cycle methodology of Wieringa [75], the problem is investigated, a treatment is designed and the treatment was validated. A literature review was conducted and interview sessions with practitioners were held to familiarise the author with the problem context and solution landscape. A combination of scientific-, technical- and professional literature as well as interview sessions with practitioners formed the basis for the treatment design (the instrument). The instrument was validated by 1) reading inspection and semi-structured interview sessions with practitioners (Section 5) and 2) an empirical validation [21] during a workshop-session at the REFSQ 2018 conference.

1.7 Results

The artefact is found to be a comprehensive tool in evaluating SRSs and, based on a series of 8 quality attributes, is regarded to be of good quality. Indicated challenges are its usability, clarity and applicability to a variety of contexts. A detailed analysis can be found in section 6.3

1.8 Structure of the report

In chapter 2 the Research Design and Methods used in designing and validating the artefact are discussed. A series of Research Goals -the Artefact design goal, Knowledge Goals and Prediction Goals- are formulated in section 2.2.

In chapter 3 the social context as well as the knowledge context are investigated. The problem context where artefact will operate in is described, a series of social context goals which the artefact aims to accomplish as well as a series of stakeholder

requirements the artefact should confirm to. The knowledge context is explored in section 3.2 where the knowledge questions are answered.

Chapter 4 describes the design process of the artefact. The structure of the checklist is described as well as the rationale behind the design decisions made during development.

Chapter 5 entails the design, execution and analysis of the interview sessions with practitioners from the Company. The interview sessions contributed towards understanding the social context, provided valuable input in designing the instrument and identified strong and weak points of the version 1 of the artefact.

Chapter 6 describes the design, execution and analysis of the live study validation session at the REFSQ 2018 conference in Utrecht, the Netherlands. Based on a mixed method checklist validation approach where Strengths, Weaknesses and missing items are identified and the quality of the artefact is assessed based on 8 quality criteria. Furthermore, the anticipated effect of applying the artefact in the problem context is determined.

Finally, chapter 7 reflects on the research process and the current state of the artefact. Recommendations are made for future research as well as using the instrument in practice.

Chapter 2

Research Design

This chapter describes the goals of the research and the methodology applied in the design and validation of our artefact: the Software Requirements Specification Evaluation Checklist plus accompanying documents.

2.1 Research Methods

This section serves as an overview of the different methods used and applied during this study and the delineation of research goals and -questions.

2.1.1 Method Overview

- As the overarching methodology the Design Cycle by [75] was followed (Section 2.1.2).
- In answering the Knowledge Questions the Empirical Cycle by the same author was followed. (Section 2.1.2)
- In designing and validating the checklist the guidelines for developing evaluation checklists by [67] were followed.
- Validation was done in four stages, the last two validation stages were based on Expert Opinion.
 1. (internal) Written and oral feedback by project supervisor
 2. (internal) Validation by comparing checklist to The Evaluation Centre's¹ validation standards [68]
 3. (external) Semi-structured interview with 2 expert practitioners
 4. (external) Live study at REFSQ'18 [22]
- In designing the materials during the live study the guidelines by [67] was followed.
- In analysing the results of the live studies the methodology proposed by [54] was followed.

¹A Research Center at the Western Michigan University, USA (<https://www.wmich.edu/evaluation/about>)

2.1.2 Design Science and the design cycle

This subsection serves as an introduction to Wieringa's Design Cycle Methodology and the prescribed Design Cycle.

Design Science centres around the study of an artefact in context and its activities consist of the design and investigation of this artefact in context. As illustrated in Figure 2.1, these activities interact between the social context (where the artefact will make a contribution) and the knowledge context (which will help shape the design).

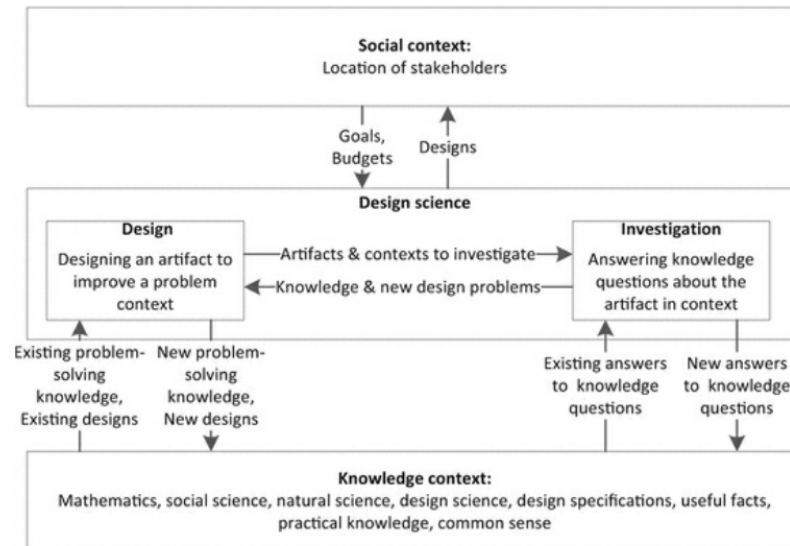


FIGURE 2.1: A framework for design science [75, Fig. 1.3]

The Design Cycle is part of a larger cycle -the engineering cycle as illustrated in Fig 2.2- in which the result is a validated treatment. In this illustration, the question marks indicate knowledge questions and the exclamation marks indicate design problems.

The design cycle only consists of the first three tasks of the engineering cycle: problem investigation, treatment design and treatment validation. The cycle emphasizes an iterative approach in developing instruments and typically in a research project you iterate over design and validation tasks multiple times.



FIGURE 2.2: The Engineering Cycle ([75, Fig 3.1])

Artefacts and treatments

Artefacts are creations made by humans for humans that serve a practical purpose. The term will henceforward be used interchangeably with the word 'instrument'. In designing an artefact, one is essentially envisioning this artefact to be used as a treatment for a given problem context. Many engineers speak of *solutions* but Wieringa avoids this term as 'it may blind us for the possibility that an artefact may solve a problem only partially or maybe not at all' and defines a treatment as 'the interaction between the artefact and the problem context'.

In this research the SRS Evaluation Checklist is considered the artefact and the use of the artefact during the validation of a SRS is seen as the treatment.

Treatment validation v.s. -evaluation

Depending on the context, slight nuances exists for defining the concepts of verification, validation and Evaluation. In the context of validating c.q. evaluating an artefact (c.q. 'treatment'), validation is the investigation whether-, and justifications that the treatment will contribute to the stakeholder goals if implemented. Evaluation is the investigation of the implementation when implemented by stakeholders in the field. The distinction becomes clearer by asking the following questions:

Treatment validation Would this treat the problem c.q. contribute to stakeholder goals if implemented?

Treatment evaluation How successful has the treatment been?

See section 3.2.1 regarding the distinction between verifying, validating and evaluating requirements (specifications).

2.2 Research Goals

The goal of this research is to develop an artefact in the form of a Checklist to help during the SRS evaluation process. As we are dealing with a design problem, instead of describing a series of Research Questions, a series of goals and knowledge questions are defined as described by [75]. As illustrated in Figure 2.3, the design science research aim contribute to a given problem context. The goals are thus categorised into Design science research goals (Section 2.2.2) and Social context goals 3.1.2 and are summarised into the Technical Research Problem Statement (Section 2.2.1). It is important to note that artefact design is an iterative, on-going process. This means that the stated goals have evolved over time whilst more insight is gained.

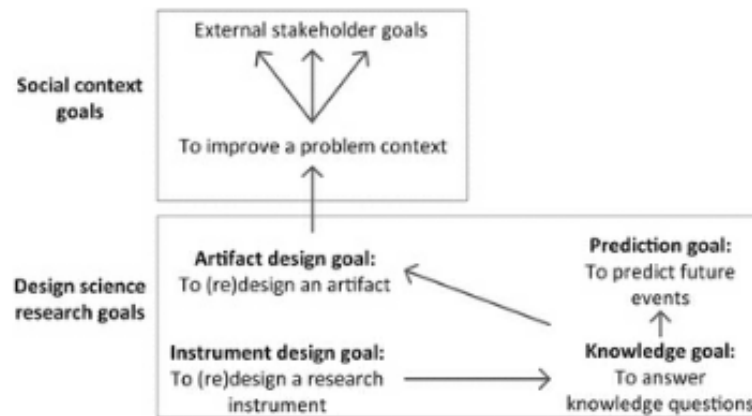


FIGURE 2.3: Goal structure of a design science research project ([75, Fig 2.1])

2.2.1 Technical Research Problem Statement

In summary, the aims for this design research effort can be summed up as follows:

- Improve the Software Requirements Specification Evaluation process
- by designing an evaluation checklist
- that satisfies the requirements listed in Section 3.1.3
- so that the stakeholder goals listed in Section 3.1.2 can be achieved
- for the context problem described in Section 3.1.1

2.2.2 Design science Research Goals

Artefact design goal

The Artefact Design Goal (ADG), or technical research goal is to improve the Software Requirement Specification evaluation process and -outcome.

- **ADG 1:** The design of an artefact that improves the SRS evaluation process and resulting outcome

Knowledge- and Prediction goals

The knowledge goals (KG) require answering a set of knowledge questions (KQ). The knowledge goals and knowledge questions are defined as follows²:

- **KG 1**: To understand the SRS evaluation process
 - **KQ 1**: How are SRSs used in the software product life-cycle?
 - **KQ 2**: What entails a quality SRS?
 - **KQ 3**: What are commonly found defects in SRSs?
 - **KQ 4**: What SRS quality assessment techniques exist?
 - **KQ 5**: What SRS evaluation/validation artefacts currently exist?
- **KG 2**: To understand the process of developing an evaluation checklist
 - **KQ 6**: How to create an evaluation checklist?
 - **KQ 7**: How to validate an evaluation checklist?
 - **KQ 8**: What is the quality of the created artefact?

Apart from the quality of the artefact, there is a desire to predict the effect of the artefact on the evaluation process. This translates into the following Prediction Goal (PG):

- **PG 1**: What is the anticipated effect of applying the artefact in practice?

²Note: by clicking on abbreviated code, you are hyper-linked to the part where the question is answered.

Chapter 3

Investigation

3.1 The Social context

3.1.1 Problem context

The process of eliciting, validating and maintaining software requirements can be a daunting task for many organisations under pressure to launch their products fast. For areas such as Government IT Procurement, the Software Requirements Specification (SRS) often forms the basis for a public procurement where its quality is of major importance. A problem in writing these specifications is that the possession of the domain knowledge is often mutually exclusive to the possession of the required technical knowledge and experience. The company SIG indicated being routinely contracted to evaluate a given Software Requirement Specification. The consultancy firm is motivated to improve their evaluation process. In their case, a third party -frequently a municipality or governmental organisation- requests the services of an independent and experienced consultancy firm to evaluate a given Software Requirements Specification. In the case of larger projects, this evaluation is done before specifications are being sent out for a public contract where interested tenders can offer their competing services based on the distributed specification(s). This evaluation can also be done in advance of contract negotiations where the specifications are an important factor in the formation of the contract. Finally, for IT projects where the client is convinced that the supplier failed to meet the stated expectations, and court battles might ensue. For these cases, an independent consultancy firm can be contracted to investigate the agreements and share their expert opinion for the court.

Having these specifications validated by an experienced, independent third party not only provides the requesting party with a way of ensuring its quality, it is also a safeguard to ensure and demonstrate that due process is followed. In the described problem context, the consultancy firm wishes to provide a cost-effective service in validating these specifications whilst upholding a certain quality standard.

If we extrapolate from this scenario, an artefact that improves the process of evaluating a Software Requirements Specification can be of high value. Not only for situations as described above, but possibly also during the specification of the requirements or different contexts such as where the applicants are less experienced in the field of Requirements Engineering.

3.1.2 Social context goals

The goal of the artefact is to achieve a certain effect or outcome in a social context, in other words: the artefact contributes to a Social Context Goal (SDG). In order to define our goals in a social context, we need to understand this context, the actors

involved and delineate any relevant stakeholder goals. Section 3.1.1 is dedicated to describing the problem context.

- **SDG:** To improve the Software Requirements Specification evaluation process

The social context goal can be divided into a series of stakeholder goals.

External stakeholder goals

From the Problem Context the following list of stakeholders were elicited:

The product beneficiary The party benefiting from the resultant information system and often the party responsible for writing the Software Requirements Specification.

The evaluation service provider The party that offers the evaluation service and who is responsible for the quality of the evaluation. This can be an external party such as the consultancy company or an in-house department.

The evaluator The person tasked with conducting the evaluation process.

The stakeholders identified above each have their own interests in the evaluation process and outcome, why is why the

- The product beneficiary
 - **SG 1:** Indication of the quality of the SRS
 - **SG 2:** Indication of faults in the SRS
- The evaluation service provider
 - **SG 3:** An improved evaluation outcome
 - (a) Improved fault detection
 - (b) Reduced evaluation costs
 - (c) Improved assessment quality
 - (d) Improved consistency in assessment quality
- The evaluator
 - **SG 4:** An improved evaluation process
 - (a) Improved Work Flow
 - (b) Reduced levels of task saturation
 - (c) Improved guidance during evaluation process

Social context goals v.s. goals researcher

It is important to acknowledge that the researcher's goals are often different than the goals of the external stakeholders. In this case, the researcher wishes to learn, explore and meet the requirements set by the University of Twente to use this research for his Master's Thesis. That said, the researcher does aim for the designed artefact to be of use by practitioners.

3.1.3 Requirements

Naturally, the artefact should meet some requirements. From the problem context, the following requirements c.q. constraints were elicited:

- Requirement 1: The artefact should be self-explanatory
- Requirement 2: The artefact should be usable in an office environment
- Requirement 3: The artefact should not overly restrict the SRS evaluator
- Requirement 4: The artefact should respect project diversity and the evaluator's experience
- Requirement 5: The artefact should not significantly affect the time it takes to evaluate an SRS in a negative way

3.2 The Knowledge context

3.2.1 Requirements in the product life cycle[KQ 1]

Requirements play a role in many parts of the product life cycle. Figure 3.1 illustrates a general outline of the involvement of requirements in a product's life cycle as described by Lauesen.

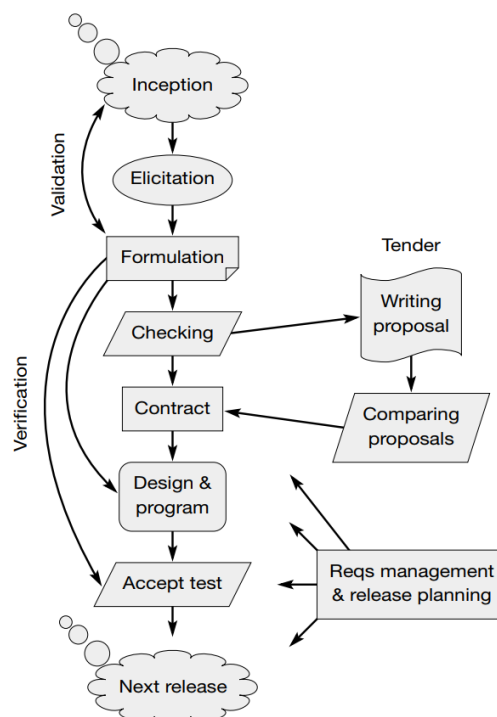


FIGURE 3.1: Requirements in the product life cycle as described by Lauesen [50, p. 291]

The development process and to which extend requirements are involved in the product's life cycle can vary per project but in developing the instrument it is envisioned that these phases are followed to at least some extend. In this document we will use to the following concepts and definitions:

The **project inception** process describes the start of the project. Here, usually the business opportunities, -goals and the project scope are identified.

Elicitation c.q. requirements gathering, describes the process of researching and discovering the requirements through consultation with stakeholders, from system documents, domain knowledge and market studies [65, p.11]. Whilst eliciting the requirements it is important to employ a variety of techniques and balance the desires of all relevant stakeholders, including the end users. Involving and engaging communication with an adequate representation of said stakeholders can be important in negotiating requirements to assure the acceptance of the resultant system. [16] It is common for the supplier to be heavily involved during the eliciting phase and share their expertise but this is not the case with tendering projects [50, p. 290]. This stresses the importance of the SRS being of adequate quality.

Formulation, or specification describes the process of writing down the requirements. Doing this in a verifiable way allows for a shared and test-able consensus between the actors involved. The outcome of this process would be the Software Requirements Specification. stakeholders' expectations[50, p. 374]

Checking determines whether the SRS is of sufficient quality and/or ready to be sent out.

Validation, Verification There seem to be some inconsistencies c.q. ambiguities regarding to what is considered as validation and verification. As Figure 3.1 shows, Lauesen sees validation as the process of evaluating whether the SRS correctly reflects what is desired of the product (i.e. the needs of the business and the stakeholders) and verification as evaluating whether the product under development meets the requirements.

IEEE standard 1012 [25], however, advocates the following definitions:

Validation: The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements (..) and satisfy intended use and user needs.[25]

Verification: (A) The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase (..) and its associated products conform to requirements (e.g., for correctness, completeness, consistency, and accuracy)(..).[25]

In this thesis, the definitions by Boehm [10] are preferred:

Validation: to establishing the truth of the correspondence between a software product and its specification. Or: 'Are we building the right product?' [10]

Verification: to establish the fitness or worth of a software product for its operational mission [9]. Or: 'Are we building the product right?' [10]

In the stated Problem Context, where an actor is tasked with evaluating a SRS, the following definition is used:

Evaluation: to assess c.q. judge the quality of an SRS.

The evaluation is seen as the result of the evaluator applying his or her experience and intuition to the results of an inspection technique (which may include verification and validation). The designed artefact does not, in fact, 'evaluate' an SRS, but is to be used during the evaluation process.

The Contract is a written agreement between two or more parties (i.e. the customer and the supplier) where the obligations of each party involved are described. In case of in-house development these parties can be two departments within the company, such as the sales- and the IT department. In a tendering process, a request for proposal (RFP) is generally sent out, requesting suppliers to send in their proposals.

Designing & developing the product should take place based on the SRS. In practice however, this does not always happen (5.3.2). Ideally speaking, the requirements should be traced into the growing product and be updated as the project goes on (see requirements management).

Acceptance testing describes the process of having the product used by the users and verifying whether the resultant product corresponds to the SRS and/or the stakeholders' expectations. One can imagine the scenario where the product conforms to the SRS but does not meet the expectations of the stakeholders.

Maintaining/managing Requirements helps keep track of changing requirements when new insights occur, stakeholders' expectations evolve and uncertainties get resolved.

Future releases Future releases serve to deliver further improvements to the product and requires further communications, requirements management and release planning. In waterfall-style processes there is usually a clear line between delivering the product and maintenance/enhancement of the product, with agile and continuous development less so. Naturally, not all software projects involve tendering and the writing of a contract, just as not all projects involve a scrupulous process of specifying and validating requirements. However, most software projects will incorporate some form of requirements elicitation which will have to be shared between actors involved in the development of the product in some way or another.

3.2.2 The importance of the requirements specification

For many software projects, requirements engineering and -specification is an integral part of the development process and crucial to project success. Requirement Specifications help to create 'a common view of what the software they are developing should do' [47].

In many large-scale IT projects there's no direct communication between the developers and the users [1]. Stakeholders' needs and relevant domain knowledge are communicated via IT consultants. This is especially problematic when SRS's are not always consolidated during the development process [8] or kept up to date [51]. There's the obvious vicious circle where an SRS is not kept up-to-date because it's not used and it's not used because it's not up-to-date. The lack of a sufficiently functioning mechanism to share what the system is supposed to do across the people involved can cause communication gaps, which in turn can lead to increased implementation cost and higher test effort [1, 8].

However, it is flawed to assess (and appreciate) the quality of the RE effort solely based on the product of the process (e.g. the SRS). In [34], Gorschek and Davis argue that the quality of the SRS is just the first level of assessing the RE process and that the effect of the RE process transcends into higher, more significant levels such as whether the project, and the resulting product was successful, and as a result,

the company and even society. To complicate matters, ‘applying good requirements practices will not guarantee you success, nor will applying bad requirements practices guarantee failure.’ ([20])

Assessing the importance of a qualitative SRS is therefore not an easy endeavour and in extension, the effect of the SRS’s quality. Furthermore, assessing the quality of an SRS is at least partly depended on subjective aspects (e.g. what is seen as ‘good RE’) and one should, in the author’s opinion, be wary of approaching the subject as an exact science. As nicely described by Davis and Zowghi in [20], we should be wary of getting too over- confident in our quest to develop or apply good requirements practices.

The SRS is seen both as a result of the act of trying to understand what the product should do, and as a communication tool to share this knowledge across stakeholders. This mechanism contributes to a common understanding and consensus across stakeholders and as a bonus that expectations are managed. The creation and existence of an SRS serves an important role in the product development life cycle but, in the end, should be viewed as a variable -or even interchangeable- method/tool. And a tool should fit the job.

3.2.3 Context-dependency

Although various approaches for gathering, writing and assuring the quality of requirement specifications exist, the degree to which the quality of said specification affects the quality of the resultant product are context-dependent. The domain, characteristics of the project and the chosen development process all affect the importance of the SRS’s quality. Effective requirement specification and quality assurance of the SRS must take contextual circumstances into consideration and should be meaningful to the engineering endeavour [56]. A more traditional, waterfall approach typically places more emphasis on the specification of detailed requirement up-front than agile approaches would, and for banking software different quality criteria would apply than for, let’s say, a web shop.

On a more general note, [56] highlights the importance of semantic quality and SRS understandability, irrespective of the context.

3.2.4 SRS quality [KQ 2]

In [19], Davis et al. define a quality SRS as one that ‘contributes to successful, cost-effective creation of software that solves real user needs.’

Writing and assessing requirements specifications is mostly based on best practices. Attempts to describe what constitutes as good requirement specification include: defining a series of qualities / criteria [19, 6, 44, 10, 9, 43] to which (parts of) the SRS should adhere to, developing standards [39, 41, 40] and even prescribing complete document templates [61] to follow. These criteria c.q. recommendations vary in scope from addressing f.e. ambiguities and phrasing, to the span of a series of requirements to the structure and content of the overall document.

What becomes clear though is that although some aspects could be standardised or even automated, the expert-opinion remains highly valuable -if not necessary- in assessing the quality of a requirement specification. But instruments to improve the quality, efficiency and enjoyment of this process remain valuable.

3.2.5 SRS challenges [KQ 3]

Requirements specifications (most notably contract-style requirements specifications) run the risk of becoming overly large. This severely limits the reader-friendliness and usability of the requirement specification and everything that comes with it. It becomes difficult to assure a mutual understanding and agreement from the stakeholders, to embrace new or changing requirements that emerge in the development process, to keep the requirement specification up-to-date and, more generally, to assure you're building the right product.

But, even a specification that correctly identifies all requirements does not guarantee those requirements will be satisfied. In [32], Gause describes the five most important sources of requirements failure as:

- failure to effectively manage conflict
- lack of clear statement of the design problem to be solved
- too much unrecognised disambiguation
- not knowing who is responsible for what, and
- lack of awareness of requirements risk.

In [72], Walia and Carver developed a taxonomy of requirements' errors in the requirements phase. They identified fourteen types of errors and classified them into *People Errors*, *Process Errors* and *Documentation Errors*. Their results highlight the significance of people and processes involved in the cause of requirement errors.

Often multiple requirement artefacts (e.g. specifications, diagrams, or user stories) are employed to support requirement-related activities. The employment of multiple different artefacts imposes 'challenges like scattering of information, incomplete translations, or inconsistencies between artefacts' [53].

Resultant requirements are generally stated in a natural language, which enables the writer of the requirement to choose the desired abstraction level and enabling all stakeholders to read and understand what shall be delivered. A major drawback is that natural languages are inherently ambiguous.

These findings indicate that for Quality Assurance, one must not only consider the requirement specification itself, but also the people, processes and context external to the document, which is why a chapter devoted to aspects external to the document under review was included in the resultant checklist.

3.2.6 SRS evaluation [KQ 4]

In the SRS evaluation process an SRS is evaluated based on its quality. Apart from the question what constitutes as a quality SRS it is worthwhile to ascertain what aspects of the SRS are empirically evaluated, by using which research method and in what environmental context. Condori-Fernandez et al. [15] provide a systematic mapping study on empirical evaluation of SRS techniques and found that understandability, followed up by completeness are the most commonly evaluated aspect of an SRS, that experiments are the most commonly used research methods and that the academic environment is where most empirical evaluations take place. A similar study performed by Pekar, Felderer and Breu also show a primary focus in research effort in Ambiguity (31%) followed by completeness (24%).

Interestingly, the results from [15], seem to implicate that what academics perceive as important does not necessarily translate into what practitioners perceive as

important and that our knowledge of these aspects might be skewed based on one-sidedly accumulated research. In their results, in the academic environment a high emphasis is placed on Understandability (84.2%) and Correctness (83.3%) whilst in the Industrial environment seems to favour Efficiency (22.2%) over Understandability (10.5%) and Correctness (0.0%). Furthermore, empirical studies were found to be rarely undertaken in government setting. One explanation might be that they're either not done, or are outsourced to academic institution or the industry. Some automated evaluation approaches [49, 38] are proposed in literature that show promise but require more experimentation.

3.2.7 SRS inspection techniques [KQ 5]

Software inspections ([24]) are generally considered 'best practice' [73] and, much like software inspections, a software requirements inspection is the process of analysing all or a part of an SRS to uncover faults c.q. weaknesses. Generally these are done by one or multiple reviewers with the aim of identifying faults early on in the development life-cycle, where they're cheaper to address. The nature of the inspection process naturally calls for a checklist to guide the reviewer in his or her process and provide the reviewer with hints and recommendations for finding defects. SRS inspections are not evaluative in nature, but their findings can be used in forming an evaluation.

Requirements Smells

In [29], Femmer et al. have transferred the concept of *Code Smells* to *Requirements Smells* as indicators of a quality violation for requirements. Based on this concept and the ISO/IEC/IEEE 29148:2011 standard [41], they developed a relatively effective [29, 28], light-weight static requirements analysis approach based on that allows for rapid requirements checking.

Defect-based reading

Defect-based reading (DBR) [58] techniques are techniques that focus on detecting specific types of faults by defining a set of scenarios that the reviewer follows during the inspection. Here, defects are classified and a set of questions is posed for each class. Scenarios, collections of procedures for detecting particular classes of defects are created and assigned to a reviewer to detect particular type of defects.

Perspective-based reading

Perspective-based reading (PBR) techniques, on the other hand, focus on examining parts of the requirements specification from different perspectives that depend upon the roles people have within the software development process. This allows reviewers to be more focused, systematic and goal-oriented whilst allowing different reviewers with different expertise to analyse the same document with less overlap.

Usage-based reading

Usage-based reading (UBR) techniques differ from the other techniques in the sense that it focuses on detecting the most critical faults, instead of attempting to uncover *all* faults. They do so with a prioritised, requirements-level use case model. In the inspection process, the reviewer is tasked with selecting the use case with the highest

priority, and verifying whether the use case is satisfactory full-filled by the document under inspection.

Checklist-based reading techniques

Surprisingly, despite Checklist-Based Reading (CBR) techniques being heavily discussed in literature (i.e. [12, 59, 50, 4, 64, 77, 2]), no mature, completed and sufficiently validated checklists were found, both in literature or elsewhere. Identified checks or checklists in literature include [61, 5, 9, 10, 52, 55, 44, 45, 27, 31, 5, 30, 6] checklists analysed in [12] and checklists found online which are used in industry or as teaching material at universities [17, 14, 74, 26, 18]. Most checklists that were found were developed before the creation of various standards [41, 40] and were developed with traditional software development in mind. For these reasons, the decision was made to develop a new checklist based on newer standards and checklist design methodology and more tailored to the work flow of the applicant.

Comparisons

Research seem to indicate [58, 59, 48, 70, 4] that DBR, PBR or UBR methods show significant improvements in fault detection rates and cost per fault found compared to checklist-based reading (CBR) methods. However, the checks applied during the experiments questions omitted many potential types of defects, were quite broad in nature and did not seem to give enough attention to the recommendations of the ISO/IEEE standards for software quality. The results are not conclusive [36, 70] and more experimentation is required to determine the most efficient inspection-based technique and whether this differs between f.e. software domains [4]. Most notably, hybrid approaches are suggested.

3.2.8 Checklist design [KQ 6]

Checklists are mnemonic devices that can present complex theory in a simple, condensed and structured way to guide applicants along a process. Checklists can be of use in a variety of applications. Be it a simple grocery list, a pre-flight checklist to see if an aircraft is cleared for take-off or as a scoring system for an annual car inspections. Checklists can implore users to apply them in certain sequence (e.g. sequentially as opposed to in-sequentially), or require users to make multiple pass-throughs, a.k.a. an iterative checklist. Diagnostic checklists typically result in some form of diagnoses or classification and evaluative checklists such as the *criteria of merit* checklist (comlist) can result in a score, or series of scores to rate an evaluand. Although a checklist can be considered less "ambitious" than alternative solutions, it is often good enough for evaluative purposes [63].

Strengths

Checklists can serve a series of benefits and have demonstrated to be very valuable in industries where the welfare of a human being is at risk and promising in other areas [35]. Checklists can reduce the chances of forgetting important steps, they are generally easier to understand, develop and apply than alternative solutions, they reduce the effect of personal biases such as the halo effect and the Rorschach effect and they can incorporate huge amounts of specific knowledge and extreme complexity in an economical format [63, 33]. Furthermore, checklists have shown in other areas such as medicine to be simple and effective safety tools for addressing patient

safety and have shown to have a positive effect on compliance with guidelines [11, 71].

Considerations

One has to be careful with expecting the success stories and design principles from other industries to translate across domains. 'A checklist is a complex socio-technical intervention that requires careful attention to design, implementation and basic skills required for the task' ([13]).

Software requirement inspections are good ways to validate requirement specifications, but it is important to note that the effectiveness of the process (and accompanying instruments) relies on the expertise of the reviewer which can impact the effectiveness of the requirement inspection [3]. Experienced reviewers on the other hand, can feel (over)confident in their ability and divert from procedure/instructions [37]. Issues may arise with strictly 'yes/no' answers on checklists with no room for supporting details as this may lead to critical problems in ambiguity of which the reviewers themselves might not even be aware [37].

Design

Western Michigan University's Evaluation Center¹ runs a project designed to provide evaluators with a growing set of useful evaluation checklists [66]. Besides these sample checklists, the group has put out several documents on checklist design [67] [62] [7] and -validation c.q. -evaluation [68] [23] [69]. In the Checklist Checklist Development Checklist (CDC) [67], Stufflebeam describes a 12-step list of designing and evaluating a checklist and is included in Section 4.2 In [12] a series of heuristics that are commonly suggested for creating effective inspection is listed (discussed in section 4.4.1)

3.2.9 Checklist validation [KQ 7]

Although critical feedback based on repeated use of the checklist should be considered the preferred method for validating c.q. evaluating and improving an evaluative checklist, Martz offers a feasible, low-cost alternative of validating a checklist by suggesting a mixed methods approach [54], which is also one of the validation methods used in this research.

¹<https://wmich.edu/evaluation/about>

Chapter 4

Instrument Design

4.1 Instrument versioning overview

Version 0.X

This is considered Work In Progress versioning for the draft version of the artefact. This version was developed based on the results from literary research and contained uncategorised series of checks and quality criteria along with informational pieces of texts.

Version 1.X

Under guidance from Prof. Nazim Madhavji and based on written and verbal feedback from both Prof. Nazim Madhavji and the contact person at the Company, the checks were organised and categorised. The resulting document similar was similar in style to what is now described as ‘the handbook’ and was approved to be used during the interview sessions (Section 5)

Version 2.X

Contains the changes (5.5) resulting from the interview sessions. The updated version was used during the live study (Section 6) at REFSQ 2018.

Version 3.X

Contains the changes (6.6) resulting from the live studies, of which v3.2 is the most current version.

4.2 Design process summary

In designing the instrument, the steps described in the Checklist Development Checklist (CDC) [67] were followed.

1. Focus the checklist task
2. Make a candidate list of checkpoints
3. Classify and sort the checkpoints
4. Define and flesh out the categories
5. Determine the order of categories
6. Obtain initial review of the checklist

7. Revise the checklist content
8. Delineate and format the checklist to serve the intended uses
9. Evaluate the checklist¹
10. Finalise the checklist
11. Apply and disseminate the checklist²
12. Periodically review and revise the checklist²

Multiple sessions with supervisor Nazim Madhavji and supervisors from SIG were held to ascertain the goals and intended uses of the instrument. Relevant literature was studied to answer the knowledge questions.

Based on the results from the literature study, a series of checks were distilled as well as a general impression of both the structure of Requirements Specifications as well as the anticipated work flow of the evaluator. The individual checks were then grouped where possible and categorised based on the parts of the Requirements Specification they would be applicable to.

The categories were then ordered and hierarchized based on the stages the evaluator would go through whilst evaluating the document. The grouping, ordering and hierarchisation of the checks was done in an iterative manner and evolved throughout the project.

Initial reviews were gathered from supervisor Prof. Nazim Madhavji and the first major version of the instrument was shown during the Interview Sessions (Chapter 5). The participants were interviewed to gain an in-depth understanding of their concerns and suggestions and these were used to improve the instrument.

A checklist sheet containing the item numbers, the names of the checks and the three validation criteria: *Satisfactory*, *Unsatisfactory* and *Not Applicable*.

Major version 2 was then used during the live study validation sessions 6 with the goals of validating the instrument as well as gain further information to improve the instrument. Based on the results the instrument was then finalised into Major version 3.

4.3 Instrument structure

The structure of the checklist is designed to emulate the work flow of the evaluator and can act as a guide. Nevertheless, the instrument is not designed to restrict the evaluator in his process.

4.3.1 Anticipated flow of the evaluation process

When first presented with the evaluand, the evaluator is expected to first observe the document and its lay-out and possibly skim through the document to get a broad picture of what is and isn't in the document. He is then expected to attempt to try to understand what product is being described. Why the product is being build, how the the product will accomplish the specified tasks and goals and who the stakeholders involved are. After this, the evaluator is expected to scan the requirements and gradually go into them more detailed, analysing them and comparing them to

¹This step is replaced by validation

²These steps are not part of the thesis

the product's description and other requirements. This process is anticipated to differ between evaluators and, in fact, type of requirements specification. Only after actually going through the document and its requirements can the evaluator conclusively say something about the overall wording in the document and criteria such as consistency and the correctness of the collective requirements. The evaluator is expected to assess whether he sees any red flags in the way the SRS is created. Whether indeed the stakeholders' relevant requirements are properly accounted for and agreed upon. Finally, any undesired elements in the SRS (e.g. implementation details) are expected to be addressed.

4.3.2 Division of sections

1. Document structure and contents
This section consists of checks to validate the SRS's structure and its contents. It checks for the presence of elements such as a table of contents, appendices and consistent numbering and referencing but also the inclusion of the necessary information to understand the specified product. The business and the environment the product will be used in, what is demanded of the product and who will benefit from the product (i.e. the stakeholders involved).
2. Requirement structure and contents
This section consists of checks for which all requirements are anticipated to either confirm to. For example, there is no need to validate for each specific requirement whether it is numbered properly. Either all requirements are numbered satisfactory or they are not.
3. Specification of individual requirements
This section consists of checks that can be performed on individual requirements. To what extent each requirement should be analysed will be left up to the applicant c.q. agreements made by the parties involved.
4. Validity of the document as a whole
This section consists of checks that validate aspects of the SRS that can only be done after having gained a thorough understanding of the SRS. Apart from a final verdict on the wording of the content of the SRS, aspects such as the validity, coverage, consistency of the requirements and feasibility of the product can be assessed. These checks are limited to information that resides in the SRS.
5. Context surrounding the document under review
This section consists of those checks that involve the world surrounding the SRS, e.g. the business and the stakeholders involved. The SRS should be a fair representation of the demands and ideally the stakeholders should agree on what is in the document.
6. Absence of unwanted elements
This section consists of checks that check for aspects and elements that should not be in the document.

4.4 Design decisions

In designing the instrument, several design decisions were made and are listed below.

4.4.1 Format

In [12], Brykczynski summarises the following heuristics that are commonly suggested for creating effective inspection checklists:

1. Checklists should be regularly updated based on defect analysis.
2. Checklists should not be longer than a single page
3. Checklist items should be phrased in the form of a question
4. Checklist items should not be too general
5. Checklist items should not be used for conventions better enforced through other means.

When developing the first iterations it was found that when phrasing the checks as questions and keeping the granularity of the checks at a level that would not over-generalise the checklist would result in an oversized document. Furthermore, based on the interview sessions (section 5) the decision was made to split the instrument into two parts: a checklist sheet that contains all the checks, and an accompanying handbook where the reviewer will find more information about how to evaluate this check. Because the checklist will be used during an evaluation process, the checks on the checklist sheet are formulated as 'judgement points' (i.e. *Completeness, Ambiguity, Feasibility*, etc.) and the 'judgement criteria' c.q. checks phrased as questions are included in the handbook ('Is the specified system feasible? A specification is feasible when: 1) All individual requirements are feasible, 2) a solution to the set of requirements exists and is achievable, ...')

4.4.2 Variability and acknowledging the reviewer's expertise

Early on the design process the decision was made that the artefact should be a supportive tool that guides the reviewer without taking away the reviewer's autonomy. There are many variables involved in evaluating a SRS, many of which are context-based and these are not always described in the SRS. What might be acceptable in some domains might not be acceptable in others. What might be acceptable in an agile development approach might not acceptable in cases where tendering is involved. These variables are seen as too broad and difficult to address. We can assist the reviewer in the process and provide data for him to base his judgement on (i.e. identification of faults), but the actual judgement should, at this point in time, be left to the reviewer.

4.4.3 Rating scheme and judgement

Continuing the discussion from the section above, when a reviewer is inspecting the SRS the following scenario's can be imagined:

- An element is present..
 - but not described 'correctly'³
 - * and the reviewer finds that acceptable.
 - * and the reviewer finds that not acceptable.

³according to best practices and standards

- and described 'correctly'
 - * and the reviewer finds that acceptable
 - * and the reviewer finds that not acceptable⁴
- An element is not present..
 - and that's acceptable
 - and that's not acceptable

Listing each option on the checklist sheet would make the sheet overly large. The decision is made to look at the possible outcomes and the following ratings are deduced:

Satisfactory For this check, what is in the SRS is satisfactory

Unsatisfactory For this check, what is in the SRS is not satisfactory

Not Applicable This check is not applicable given the context

⁴This is added for reasons of completeness

[0][0]

Chapter 5

Semi-structured interviews with expert practitioners

5.1 Introduction

Given the exploratory state and nature of the research at this moment in time and the desire for impressions regarding the feasibility and quality of the instrument thus far, interview sessions with expert practitioners were organised. The company Software Improvement Group (SIG, from here on referred to as The Company) agreed to facilitate interview sessions with two practitioners with substantial experience with both software development and the writing and evaluation of software requirements specifications. Because of the scarcity of the practitioners' availability, the interview sessions were limited to one-hour time slots per practitioner.

It is important to note that at this point of the design, the instrument solely consisted of -what is now called- the handbook. At this point in time, the handbook consisted of 13 pages with checks and small written descriptions and there was no separate checklist sheet. The handbook was considered Work In Progress and contained unfinished items but the overall structure, contents and flow of the document was similar to the current c.q. latest iteration.

5.1.1 Research goals and Research Questions

The goal of the interview was to see how the research effort compares to the opinions and experiences of the expert practitioners. In the understanding of the problem context, the way to address this, but also to get an indication of the quality and viability of the instrument. Apart from gaining insights on the problem context and the quality of the instrument, the interviews were to bring ways to improve the instrument to light. These goals are broken down as follows:

RG 1 To better understand the problem context

RG 1.1 To better understand the environment the SRS is used in

RG 1.2 To better understand the way the SRS is created and used

RG 1.3 To better understand the evaluation process performed by the practitioner

RG 1.4 To better understand which aspects in the problem context can be standardised

RG 2 To assess and improve the quality of the checklist

RG 2.1 To validate the chosen form of the instrument

RG 2.2 To validate the flow of the checklist

RG 2.3 To validate contents of the instrument

RG 2.4 To predict its effect on the problem context

RG 2.5 To gather information on ways to improve the instrument

To this end, the following Research Questions are delineated.

RQ 1 To what extent will a checklist help in evaluating SRS's?

RQ 1.1 Is a checklist a suitable form for an instrument?

RQ 1.2 What elements of the evaluation process can and can not be standardised?

RQ 2 What is the quality of the instrument as-is?

RQ 2.1 What are its strengths?

RQ 2.2 What are its weaknesses?

RQ 3 How can the instrument be improved?

5.2 Methods

5.2.1 Interview Design

The nature of the interview sessions was exploratory. The interview sessions were held to gain a better understanding of the problem context, to assess the quality of the developed instrument as well as to gather information on how to improve said instrument. The style adopted was that of a semi-structured interview with open-ended questions [46] as it suits the research context of this project. New insights were usually rebounded with a follow-up question and elaborate answers were encouraged. The involvement of the expert practitioners allowed the Researcher to draw from the participants' experience and to validate aspects of the designed instrument and the decisions made whilst creating it. A Semi-Structured interview approach was determined to be the best fit given the nature of the research, the available time and participants and the stated research goals. In designing the interviews, the underlying rationale was to take the participants along steps taken from the problem context to the designed instrument. The flow of the interview was to go from conceptual to tangible and concrete. The first part of the interview focused on the background and experience of the participants. The second part focused on the experiences of the practitioners in working with SRS's, e.g., "What is easy to assess when validating SRS's and what is difficult?". The third part consisted of giving the participants a sample SRS and asking the them to describe how they would go about validating this specification. The fourth and last part consisted of showing the participants the developed instrument and asking them for their impression of the instrument. To minimise the factor of response biases, the sample SRS and the designed instrument were introduced as late as possible. The resulting Interview Scheme can be found in Appendix B.1.

5.2.2 Logistics

General information

The interview sessions were held at the 14th of December 2014 at SIG headquarters in Amsterdam. Detailed information can be found in Table 5.1

TABLE 5.1: Interview details

Date	2014-12-14
Location	SIG hq, Amsterdam
Interview style	Semi-structured
Number of participants	2
Duration	60 minutes per interview
Language spoken	English

5.2.3 Materials

The materials involved during the interview sessions are described in Table 5.2. The SRS that was provided by The Company was a real requirements document used in a software tendering process they were involved with. As per the wishes of The Company, any details regarding the contents of the requirements document are removed from the transcriptions. The participants had not seen any documents used during the session beforehand.

TABLE 5.2: Materials used during the interview sessions

Materials	Provided for by
Printed copies of the checklist document	The Researcher
Laptop with voice recording functionality	The Researcher
A sample Software Requirements Specification	The Company
Availability of two expert practitioners	The Company
Private room with chairs and a table	The Company

Specific instructions for the participants

The participants were asked if they were comfortable with speaking in English, they indicated that they were. The participants were instructed to speak out loud any thoughts they had whilst looking at either the SRS or the instrument. Participants were asked to not to speak any specific c.q. identifiable details regarding the sample SRS out loud.

5.2.4 Participants

The interview sessions were conducted in December 2014 and focused on practitioners with vast experience with different types of software projects as well as the drafting and validation of Software Requirements Specifications. Furthermore, the participants were to be proficient in the English language. Two senior practitioners from The Company were interviewed. They described that they had broad experience with both more structured, traditional software projects as well as agile approaches. They described having been involved with drafting and validating Software Requirements Specifications for numerous projects. The participants consented to being recorded and for recordings to be used for analysis purposes within the scope of this research.

5.2.5 Procedure

The participants were arranged for by the The Company's contact person and were selected based on their expertise, availability and interest. The interview sessions

were held during office hours¹, meaning the participants received indirect financial compensation from The Company. The participants were vocally informed of the details and goals of the interviews in advance and consented to the interviews being recorded and used for analysis. They verbally indicated having no problem with holding the interviews in English. Each interview session lasted 60-65 minutes and were held by the Researcher. The sessions were recorded using a laptop and microphone. The participants were friendly, talkative and had no problems in answering the interview questions. The participants spoke clearly, concise and their English was more than understandable. Although the interviews were kept according to the structure, both interviews took their own form and provided for interesting discussions.

5.2.6 Analysis

The interview recordings were transcribed and coded by the Researcher using Atlas TI² and resulted in two transcripts of roughly 8.000 words each. In an iterative manner:

1. the transcriptions were read
2. the code skeleton was created
3. natural meaning units were identified
4. codes were created based on the idiographic themes of the meaning units
5. the units were matched^(*:1) to one of codes
6. the codes were categorised by connecting^(*:1) each code to a corresponding node of the code skeleton
7. the code skeleton was re-structured and extended to support the induced codes

This process was gone through multiple times until: 1) every part of the transcriptions was covered by a quotation, 2) every quotation was found to be of the correct unit size, 3) every quotation was attributed exactly 1 code, 4) every code was connected to a fitting code category in the skeleton and 5) The code skeleton was structured logically in a tree form in line with the Research Goals. The finalised Code Tree can be found in Appendix B.3

Development of the code skeleton

Based on the Research Goals and the transcriptions having been read multiple times, a high-level Code Scheme (or: -skeleton) was created in a semi-inductive manner to be used as a starting point for the coding process. The (finalised) high-level code skeleton, which is very similar to the starting code scheme can be found in Appendix B.2.1. During the coding process the skeleton was re-structured and extended and the resulting low-level code skeleton can be found in Appendix B.2.2. Nodes belonging to the Code Skeleton are connected to one another via 'is part of'-relations.

¹9.00 to 17.30 in the Netherlands

²<https://atlasti.com/>

Identification and coding of the natural meaning units

In an iterative, deductive manner natural meaning units were identified, coded and categorised. The natural meaning units were identified and scoped based on the point the participant was trying to make; either in answering the question or whilst assessing the instrument. Based on the meaning units, idiographic themes were delineated and formed the basis for the codes. Short descriptions (or sentences) of the theme were used as the code's descriptions. Similar codes were combined and the units were categorised by connecting^(*,1) each to a node in the code skeleton. Where needed, the initial code skeleton was altered and extended to include newly identified categories. The codes that are grounded by one or more units are connected to nodes of the code skeleton via an 'is-a'-relation (illustrated as 'isa' in the diagrams).

To clarify this process, whilst quoting and coding parts of the transcriptions during the validation stage, each time a participant stated out a positive remark, a code was created for this piece of content. For example, the following excerpt of Participant 2, whilst reading the section in the checklist listing checks for individual Requirements:

Ah if you trace back, then it should be, yeah okay and this is internal if it doesn't conflict with the other requirements in the same list. Understandable. And ambiguous, that's really important, because otherwise you get discussions, you don't want that. Consistency, that's okay. Verifiable, it should be verifiable, yeah. (Participant 2)

, each part was quoted, and the following codes were created and added to these quotes: 1) internal/external Correctness, 2) Ambiguity, 3) Consistency and 4) Verifiability. These codes were then connected via an 'is-a' relation to the following node: *Validation -> Positive Aspects -> Contents (+)*:

Additional codes

Units that were deemed of insignificance (like the introduction for the participants) were coded as 'Irrelevant'. Units that described the participants' backgrounds were coded as 'Participants background information' and units that could not be connected back to the Research Goals were coded as 'Miscellaneous'.

The resulting Code Scheme can be found in Appendix B.3 and the Code Table can be found in Appendix B.4

Inter-subjectivity

The Researcher was the sole person to hold the sessions and to transcribe the interview recordings. Due to the interviews being more of an exploratory nature the coding process was done solely by the Researcher which removes any concerns regarding inter-subjective verifiability. The coding scheme as well as the code attributions were not judged by a third party.

Application of the coding

The Unit of analysis varied depending on how well the statements can be isolated and its worth in relation to achieving the Research Goals. In some instances, the unit consisted of the entire answer to a certain question, e.g. a question regarding the problem context. This means that this unit could consist of an entire paragraph. The

reasoning behind this was that this entire collection of inter-related sentences contributed to answering the interview question, and thus understanding the problem context. For others codes, such as those attributed to content of the instrument that was assessed positively, a small phrase like "Table of Contents, good.." was deemed an isolated statement and was warranted its own unit.

Choice of excerpts

Quotations were identified and scoped based on the point the stakeholders tried to make and their relation to the identified categories. The size of the quotations varied from words to complete answers. Excerpts used in the analysis were selected based on their significance to the research goals and their ability to clarify and ground conclusions being made. Furthermore, excerpts were selected that were contradictory to either their own or the other participant's statement.

5.3 Results

The two 60-minutes long interview sessions resulted in two transcriptions of roughly 8000 words each. The two participants were asked 40 and 43 questions respectively. In total 184 codes were created, of which 33 made up the code skeleton and 151 codes were created based on -and linked to- 145 units for Participant 1 and 149 units for Participant 2. The chapters of this section are structured to accommodate the code tree as depicted in Appendix B.3

5.3.1 Participants' background

Participant 1

Participant 1 (male) indicated having worked for The Company for 2.5 years. The participant described starting as a Business Analyst for a shipping container company. He described having a broad background in sales, programming, SAP implementations and IDE³ invoicing before switching to The Company. Here, he moved up from medior, to senior, to principle consultant. In these years at The Company he worked at around 5-6 projects where the specific scope of the project was related to SRS's where his responsibilities lied in both drafting and validating requirements specifications. The participant indicated being involved with both agile as well as more traditional ways of software engineering. The participant summarised his experience as consisting of one year managerial work, three years of in-dept IT work and roughly four years as more 'superfluous IT engineering'.

Participant 2

Participant 2 (female) indicated having worked for The company in the past for around 4 years before leaving to work in the space domain for three years, before coming back to The Company. In the space domain, she described having worked as a product assurance engineer where she gained extensive experience working with (including reviewing) standards, requirement specifications and regulations. Apart from the space industry, she indicated having worked for the Dutch Government where she worked in different groups, ranging from developing for legacy systems as well as developing new systems in an agile way. The participant indicated having

³Electronic Data Interchange

25 years of experience working in the software industry and software development. She had experiences as a team lead but not in a managerial position.

5.3.2 RG 1: To better understand the Problem Context

The way the SRS is used

The SRS is used to describe the specifications of the desired software system. In the case of outsourcing, the SRS is often used as the basis for contract forming with a software vendor. At the start, to discuss the costs, but also at the end to verify whether the resulting system delivers as promised. *'what you see often is that it is the basis for contract forming, and when you form a contract with your vendor, it is very important in many stages, and not only at the end, to check against: "what did you do and did you live up to the expectations?"'* (Participant 1)

'(if) your requirements specification facilitates the product, then you're going to have a high chance of having the the right product but that doesn't guarantee it.' (Participant 1)

'I firmly believe that if you are not one of these exceptional cases, it pays off to have a very structured methodical approach to having your requirements specification set up.' (Participant 1)

we have seen with some customers where we propose that we have a set a requirements list where we say: okay first make these requirements in to a skeleton of the system and then there's additional requirements on top of the first set that you need to implement. If you do that, you have a baseline at a certain time (Participant 1)

Furthermore, it can be a guide in determining whether the project is on track by comparing the implemented functionality to the project's time-line. The SRS is often drafted by the organisation itself, sometimes with help from external experts, and commonly the SRS is verified by stakeholders across the organisation before being published. The Company provides services to organisations and municipalities to help with drafting and evaluating SRS's. In cases where IT projects go wrong, their services are requested to act as intermediaries or to offer their expertise in court cases.

The fit between the SRS and variances in the Context

The participants indicated that the SRS should 'fit' its surrounding context. They indicated factors such as the software development approach being used, the system's Software Domain or variances in the project itself that would impact the form of the SRS, e.g. its scope and level of detail.

Participant 1 indicated that he would expect a different level of detail in the SRS for projects that are developed Agile as opposed to projects following the traditional Waterfall approach. The size of the project, as well as the level of "maturity" of the organisation that set ups the SRS also have a positive impact on the SRS's size and level of detail.

In organisations that are less well known to these sorts of things, you see that they just start with a list of the functional requirements, they'll forget most of the non-functional requirements and they don't even come up with a description of architecture or these kinds of things. Or. that will not be in there. So it's more of a maturity level, I think, in terms of what should be all in there, or what is there, or what we see in there. (Participant 1)

Furthermore, the contracting method and the current state of the procurement process should be taken into consideration when drafting or evaluating the SRS.

I want to note that not.... having a fully detailed thing, like this⁴, is not necessarily the best way to do it. It also depends on your contracting method. Because if you use, for example, best value procurement, then this is something that will be developed during the assignment with the vendor and the client themselves because you have a very much shorter list to base your decision on. And to base a first, let's say frame contract on. (Participant 1)

Participant 2 indicated having worked in the Space Domain where almost everything is specified in detail up front. Because everything is of importance, prioritisation between requirements doesn't necessarily take place and there's a bigger focus on testing as compared to 'standard' Software Engineering. Specifications are usually also broken down into multiple documents, and elements that are often described in the SRS are referenced to instead. For bigger projects you see tools being used for managing the requirements.

I mean you can imagine that if you work in an agile way, that you, I mean you don't completely finish it. You need something to start with, (..) at the end you don't have a full description of all your requirements, you never did. (..) Eh, on the other hand, if you work like in the space domain, with the V-model, there you really need it because in the end you're going to start here, (..) at a certain point you will validate the product you have against this document. (Participant 2)

This is by analysis because it's hard to test everything, which is by the way, is normal. You do not always test things. It depends actually, eh, because it's costly. And it depends on the importance. Sometimes, if it's really important, even if it's hard you're going to test it anyway because you don't want it to fail, especially for space missions because then you cannot go up. But if you think: well there are other ways to solve this if it fails, then you can also decide to do so by analysis, which is generally cheaper. But then, yeah (Participant 2)

Commonly seen pitfalls

The participants described that as organisations start to outsource their IT to focus more on their core businesses, they lack the IT capacity and knowledge to not only draft SRS's but also to evaluate the work being done by the vendor. This is especially problematic when the SRS forms the basis for the contract and the organisation is unable to steer the development-process in-between. This means that all the hardship due a mismatch between the developed system and the *actual* desired functionality gets left to the end.

But people outsourcing the work often don't have the expertise anymore to assess the work that's being done. And this makes the whole process of making correct requirement specifications a whole lot more difficult. Because, let's for instance take a company that used to make its software self, and then get rids of a lot of the company because it wants to focus on its core business, let's say a bank,

⁴The sample SRS

whatever. And they outsource everything, then what you often see is that there is not enough IT capacity internally anymore that knows how that process should work so that they can evaluate, not maybe steer it, and do it, and stuff like that, but at least evaluate how well a vendor is doing its work (Participant 1)

What we often see with customers is that they also don't do anything directly in IT. But they need IT as their support of their organisation. (...) But, because they are not IT professionals, they don't know how to do it, so the amount of work and effort and information that goes in is a lot less then what you're showing me here⁵. (Participant 1)

And this⁶ is often something that within vendor, customer relations, or the client-vendor relations that people are not keen on doing this and are doing it too little. So the hardships gets left to the end, which makes that based on that, we have a big discrepancy on what the requirements were at the beginning and what they are in the end. (Participant 1)

All this becomes even more problematic when the initial SRS is disregarded once actual implementation begins.

I think that one of the things that goes wrong often is that the requirements specification is drafted at the start, then put in a drawer and not looked at anymore. Because then a team comes in, they drop the people and decide what actually, really needs to be built, right? That's what's happening. (Participant 1)

Difficulties with drafting the SRS

Generally speaking, the people involved with the work have the domain knowledge but have no experience with writing SRS's whilst (external) IT experts often know how to draft requirements specifications but are not familiar with the work being done. This creates a mutual dependency on drafting the SRS.

'So what we often see is that our clients are able to come up with the functional specifications, of what should the system do and are then not able to correctly phrase this on the level that is needed.' (Participant 1)

If you go to the second case, most of the times when there's an expert involved that more, overlooks the process and is able to do this from his point of view outside of the organisation inwards, you see that a lot of the functional items get left out. Because they are not aware of the full complexity of the process. (Participant 1)

This separation of knowledge is apparently notably when it comes to the Functional- and Non-Function Requirements. For people that are not experienced with writing software specifications the NFR's are usually very difficult to specify correctly. For experts not involved with the work, it tends to be the FR's that are difficult to come up with.

⁵The sample SRS

⁶Checking whether the product lives up to the expectations

'And then the non-functionals are something that (the clients) do not know. Either not at all or incomplete' (Participant 1)

With Functional Requirements, it was described that often you see that people not only have difficulties coming up with them, but also in specifying them in such a way that they are actually measurable.

I'd say people aren't consistent on their abstraction level. For certain parts, what you often see is that people directly from the organisation are detailing what functional requirements there are and how they should implement it. Or there are a couple of experts from the organisation that are doing this for the rest of the organisation. In the first case, it is often the case that these people know a lot about a certain part, and about another part they don't know enough. So they tend to be very detailed about several parts they are involved in, which is their daily work, and then the other parts they really don't know so that tends to be not as detailed as it should be. So there's a discrepancy there. (Participant 1)

For NFRs, it was indicated multiple times that often people have trouble with specifying the right abstraction level, and being consistent in this level of abstraction. This is due to unfamiliarity of how to generate such a document, a lack of a structured approach but also because people will be more familiar with certain parts of the work than other parts. *'what we often see, is that it's difficult for a lot of people to specify their non-functionals in such a way that they are actually measurable.'* (Participant 1)

The participant went on to describe that political power comes into play when drafting the requirements specification. This needs to be addressed to make sure the requirements are fair.

The thing that you will see is that political power comes into it. To decide who is having the biggest say in putting things on the requirements list. So this something that from a drafting perspective, you need to it to be taken into account when you're responsive for doing this. That you don't have one session where everybody is at the table, because then the loud mouths get the most inputs. You need to make sure that the structure..that also that process is structured to make sure you put that in place correctly. (Participant 1)

Techniques to acquire Domain Knowledge

In order to acquire the necessary Domain Knowledge as external experts, the participants indicated that they used a variety of interviewing techniques and that they made use of their experience paired with their common sense. But they indicated that this is not a fool-proof way, if you really wish to gain a complete picture of the functionality involved you have to study the transaction trail.

'If you go a system, there's always a transaction trail. You just follow the transaction from start to end and you should have a full model of what they're doing. But this is why the functional part is more difficult than the non-functional part.' (Participant 1)

The SRS validation process

When tasked with stating out loud how they would validate the sample SRS, they indicated skimming the contents first. The table of contents, the scope, system description, references, use cases, presence of Functional- and Non-Functional Requirements, etc before looking more detailed into specific contents. Whilst doing so, they indicated things that caught their attentions, things that stood out to them.

I always do, first is just look at the contents. To see what kind of information is there. And then, okay, so they have scope, that's good, they need that. They have reference documents, that's also normal. That's funny, they have a system description here. That's sort of almost suggest that they don't have a higher level document, but I can only see if I look into the details. (Participant 2)

After gaining a global picture, they indicated looking for the overview of the requirements and diving into the requirements themselves. 'Right so you would first look at the total list of requirements (Participant 2: yeah) and then maybe also look at traceability (Participant 2: yeah). Right.' (Interviewer)

Both participants indicated that they would need at least 8 hours to go through the entire SRS. And perhaps another 8 hours from a commercial point of view to get back to the clients and write a report. But this can vary wildly based on the size and quality of the SRS. 'if they have a sort of a reference matrix between this higher level and this, then you could do it much quicker. Sometimes not even more than 2 hours.' (Participant 2)

The participants indicated that, as outside experts, generally speaking they managed to see check if the SRS is structured well, to indicate any missing information, see whether the requirements are complete, whether things are defined at the right level and whether these things are reasonable. 'I think we, as an outside party, we are fairly good at assessing whether these things are reasonable, whether they are defined on a correct level, and whether they see information as missing.' (Participant 1)

The participants indicated multiple aspects that are difficult when validating the SRS. Amongst other things, they listed: 1) checking whether the requirements are complete, 2) understanding the work processes, 3) identifying missing Use Cases, 4) possible ambiguity in terminology and descriptions, 5) tracing the requirements to the system description and seeing if your requirements are a coherent set (especially when you're unfamiliar with the system) and 6) evaluating whether you created the right requirements.

Completeness, this is the difficult part (Interviewer: right yes). This is really, that's what I was referring to, so okay, they don't have tracing from the system overview to the requirements themselves, so I'll have to do it myself and then it's very very difficult to see if it's complete, while it's important. Of course. This is very difficult. It would be nice to see how far you get to really help people in a checklist to check for completeness but this is difficult. (Participant 2)

Viability of standardising tasks involving SRS's

Both participants agreed that some aspects of working with SRS's could be standardised but felt that other aspects were difficult or impossible.

The process of evaluating? (..) You can standardise it. (Interviewer: Okay) And the fun thing is that that is not really done, because I had to find out for myself in space domain how I need to review this document and that's a bit crazy. (..) there's not really a checklist for evaluating. (..) this is the kind of knowledge that it's in the heads of people, which is kind of stupid and dangerous because in the space domain a lot of people that worked there started in the 70s and are about to retire (..) So if we don't do anything there, then a lot of knowledge will be lost in the coming ten years. (Participant 2)

The participants indicated that the use of a checklist could help with 1) drafting the SRS, 2) Validating the SRS, 3) checking processes revolving the document and specifically could help with specifying NFRs. Moreover, Participant 2 hinted that the checklist could help with sharing and retaining knowledge. This could translate to a more scientific approach to evaluations with more universal outcomes (e.g. multiple evaluators would end up with similar outcomes of the SRS evaluation).

Interviewer: *So do you think that the process of validation of a checklist could be standardised?*

Participant 1: *I think so, most of the things, yeah. A lot of the things, definitively if you look at the non-functionals, there's a lot of standard things in there. I think also the deviations that you often see with specific requirements for specific systems in terms of: it's a special system because it needs some sort of performance. Even that you can hint and guide while using a certain checklist I think.*

The participants indicated various aspects that they felt could not be standardised, e.g. that you would always need the the experience of an experienced person for a well-founded decision. Aspects such as assessing completeness, determining the right scope and required level of detail of the SRS and assessing both internal- and external correctness were listed as things that varied too much and that were too context-dependent to standardise. Making the decision itself was indicated as an expert approach, but the fact that the decision needs to be made might still be valuable to an evaluator.

I think that's if you would do that⁷ for the first time, I don't know that you would be aware that you should, eh, dive into that, check for that. Because, yeah I don't even know how I learned that, just by asking. And just looking at how my colleagues did that. But that's important (Participant 2)

5.3.3 RG 2: To assess and improve the quality of the checklist

Positively perceived Aspects

Regarding the form of the instrument, the participants agreed that a checklist would be a good tool to have whilst evaluating (or possibly drafting) the SRS. The participants approved of the division of the chapters, and indicated that the flow of the checklist mirrored the flow of their own evaluation approach.

'I think these are actually the things that you do. I think this is a reasonable break-up of things that you should look at.' (Participant 1)

Interviewer: *And do you also think this would follow, like, if you were to evaluate the requirements, would this process, these steps, would they follow also your approach?*

Participant 2: *I think that's pretty much what I did, right? I just looked at the content table, first. To see what was in there. Ehm. And then I immediately moved to the list. To see how they actually ordered things, and what attributes they had. And then you start looking into the list itself. This is a logical to do things*

⁷Checking whether you agree with the way requirements will be verified

The Participants agreed that some checks should be optional but that it will not be possible to standardise which checks should be optional or not. This is seen as context-dependent and requires expertise and experience to determine correctly.

Interviewer: So for one, for agile you might not want all kinds of these... (P: hmm hmm *agrees) Would you, specify that? So these checks apply to agile, and these not, or would you..

Participant 1: Nah it's project specific I think (I: right). I think for, I think you need to decide for a certain project whether these things need to be in there. It's good to have the list, and then to be able to say: 'okay this, yes yes, this no no'

Interviewer: Right and then for each check that does not, that it's a no, then you can say oh is it necessary or not, okay then.. (P: yep). Okay.

Figure 5.1 displays the results from the contents identified in a positive sense as described in section 5.3. Note that this figure lists all content vocally indicated as positively perceived by either of the Participant whilst skimming through the checklist. Although this cannot be seen as a strong method of validation, this does indicate that much of the content in the checklist was recognisable to the participants.

5.3.4 Answering the Research Questions

RQ 1 To what extent will a checklist help in evaluating SRS's?

RQ 1.1 Is a checklist a suitable form for an instrument?

RQ 1.2 What elements of the evaluation process can and can not be standardised?

A checklist was seen as a valuable tool to speed up the evaluation process, prevent the practitioner from missing important checks and could help guide inexperienced practitioners. Furthermore, a checklist could help drafting the Software Requirements Specification but also to share and retain knowledge.

It was agreed that some, if not most aspects of working with SRS's could be standardised but other aspects were seen as very difficult or impossible. Generally speaking, the checks and steps involved in the evaluation process could be standardised but deciding whether the check is required, how detailed the SRS should be and assessing whether the SRS satisfies to the described check given the specific context of the Software Project is seen to rely on the practitioner's expertise.

RQ 2 What is the quality of the instrument as-is?

RQ 2.1 What are its strengths?

RQ 2.2 What are its weaknesses?

The instrument, its structure and most of its content was perceived positively. The structure was seen to match the flow of the practitioners' evaluation process and the instrument was seen to 'make sense' (Participant 1). The biggest flaws indicated were its usability and practicality. The instrument was seen as too long, too detailed and too specific in some cases.

RQ 3 How can the instrument be improved?

Several suggestions were made on how to address the practicality problem. The most notable suggestion was to split up the instrument into a checklist sheet and a handbook that provides the user with more detailed information if he/she so desires. It was suggested to create the checklist sheet in matrix form to allow the practitioner to mark the individual checks. It was suggested to generalise individual checks more and multiple suggestions were made to improve specific checks.

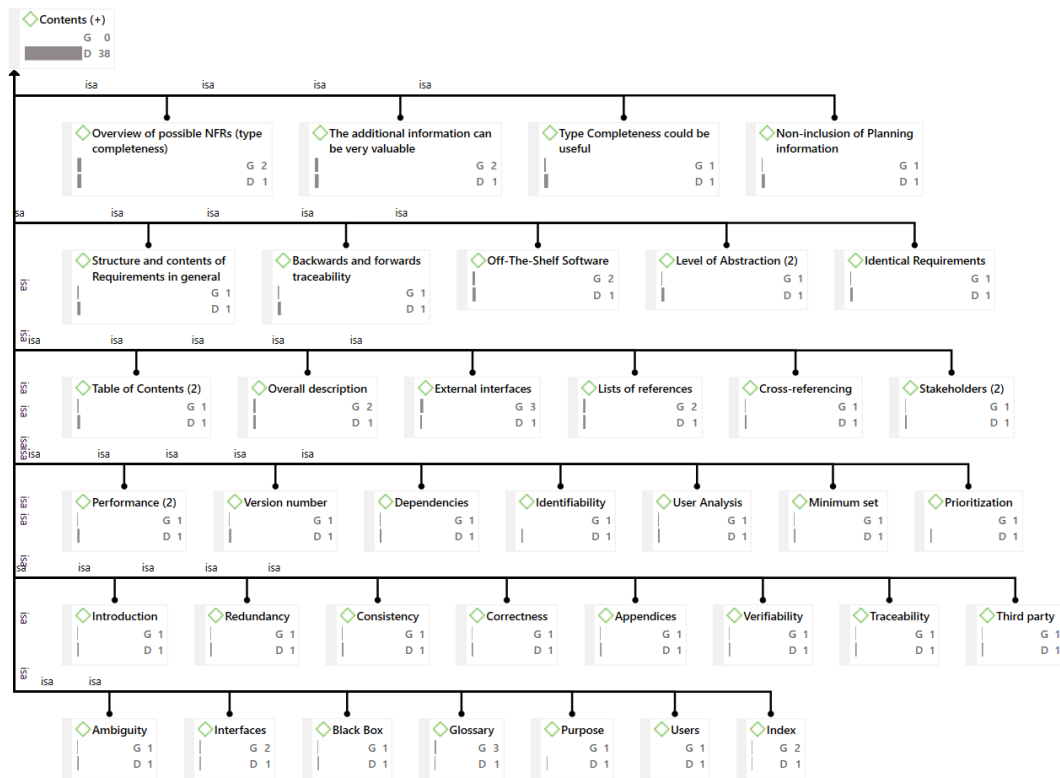


FIGURE 5.1: Positively perceived content during the interview sessions

Negatively perceived Aspects

The biggest critique for the form of the instrument was that in its current state that it was perceived as being too long, detailed and unpractical to use in their line of work. These critiques were phrased as as proposed improvements, which is why they are listed in subsection 5.3.4.

Content that was perceived as unclear or unwanted is listed in Figure 5.2. Most of the critiques revolved around parts of the checklist being overly extensive or some checks that were seen as optional and not critical.

Proposed Improvements

Figure 5.3 illustrates the total list of improvements suggested by the practitioners.

The majority of the suggested improvements for the checklist could be categorised as making the checklist more practical for the user. Both participants independently agreed that ideally the checklist should be condensed to roughly 3-4 pages. Various ideas were suggested to accomplish this. One idea was to split up the checklist into an essential part that would entail all the 'core' c.q. 'standard' checks, and

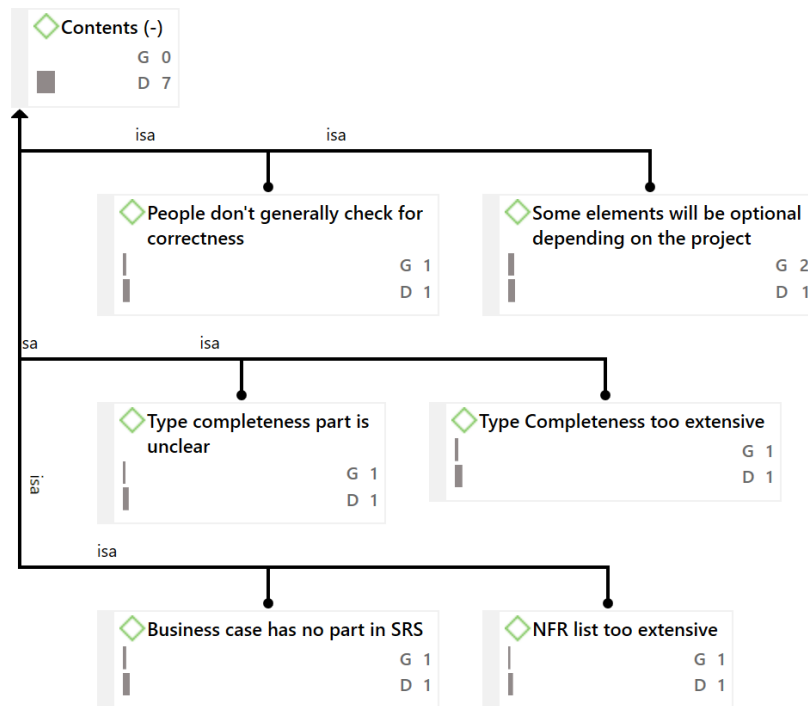


FIGURE 5.2: Negatively perceived content during the interview sessions

an optional part that would contain the 'optional' c.q. 'context-dependent' checks. However, it was also suggested that the evaluator should decide which check was deemed optional and which wasn't.

An other suggestion was to create 2 documents, the first a checklist denoting just the names of the checks and the second being a 'handbook' that would provide more information to the user. The headings, especially after being used to the checklist, would be enough information for the experienced practitioner to see what is expected there. Any references to sources or further documentation was suggested to be removed unless it proved of added (practical) benefit.

The (small) list of suggestions regarding specific contents were adapted into the checklist.

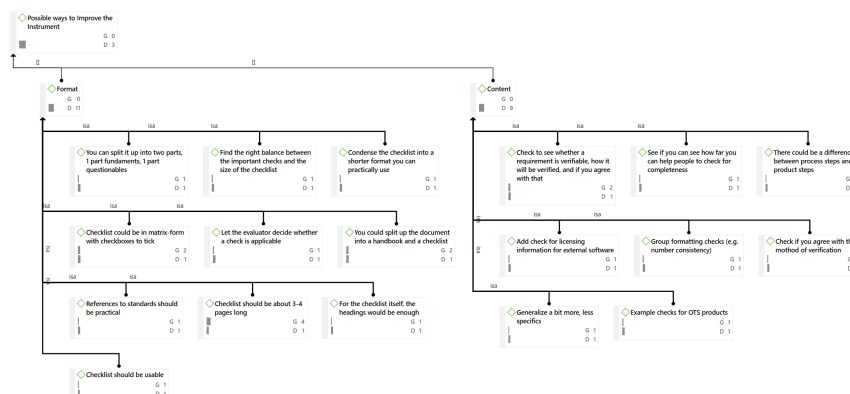


FIGURE 5.3: Suggested improvements during the interview sessions

Anticipated Effect on the Problem Context

Participant 2 indicated the checklist would speed up the SRS evaluation process and would prevent missing items.

Interviewer: Okay! So do you think this would be of help? For in your process?

Participant 2: Oh yes I always like it when I have a (..) and I don't have to, all the things that I don't have to think about, will save me time. And I'm always pressed for time, I'm always very very busy because, you know, like I said, then I can say: I can do this in a day, which is nice, of course there will be other stuff. I cannot, you know, have this one day only doing this. (right). So in the end, I'll have to do it quicker than that. And everything that really helps me to do it quicker is of aid. Of course.

Interviewer: Okay so it would help in speeding up the process, would it also have an affect on the quality of your validation?

Participant 2: Sure because I think you risk less that you miss something. Especially when you are pressed for time

Due to time constraints, participant 1 was not asked whether he thinks the checklist would help him. Concluding the session, however, the participant had the following to say: *'It makes sense. The document.'* (Participant 1)

5.4 Discussion & Conclusion

Although the time for the interviews were limited and the interviews were conducted with a sample size of 2, the practitioners possessed extensive experience with working with Software Requirements Specifications for both traditionally- as agile developed Software Projects. The interview sessions were deemed very valuable and despite the limited time and small sample size the Research Goals were reached. The format, concept and content of the instrument were perceived positively in general and the participants reckoned the instrument would be of value. Several improvements in the usability-aspect of the checklist (most notably the balance in size, scope and level of detail of the checks) were requested.

5.4.1 Trade-off between comprehensiveness and usability

The ideal instrument

The participants were in agreement that the checklist should standardise that which is standardizable, but leave certain aspects up to the expertise of the practitioner. The checklist should not be longer than 3-4 pages, but a significant amount of both the checks as well as the added explanatory texts were deemed valuable. A checklist with too many items or with too much text was deemed unusable.

So I think most of these things, it makes sense, my question would be whether a lot of people would work with an extensive checklist like that. (right). Because this is a little bit daunting. I think. To a lot of people, because if you have to come up with all of these kinds of these things.. (Participant 1)

Furthermore, some checks were indicated by the participants as optional, or non-essential, but they still felt the decision should be left to the practitioner to decide whether this check is essential or optional. *'I think you need to decide for a certain*

project whether these things need to be in there. It's good to have the list, and then to be able to say: 'okay this, yes yes, this no no' (Participant 1)

Conflict and addressing the problem

These criteria are to some extent conflicting and this creates a challenge. The more extensive the checklist becomes, the less usable. But the decision which checks can be seen as essential and which not is content-specific and should be left to the expert. A way to condense the checklist and make it more usable without inadvertently removing checks would be to condense the texts and, where possible, show diagrams instead of lists.

Interviewer: Would you prefer, something like a diagram right here, or, eh, more text?

*Participant 2: Oh no, please make it a diagram. (*laughing*) Oh no, I won't read it.*

Ideally speaking the instrument should be usable by practitioners with a wide variety of experience. What one finds useful another may find muddling. To quote participant 2:

this⁸ I wouldn't read. For this, I think for me, would be enough information then I actually can come up with these kinds of things myself, but that, of course, you know, I work in IT for 25 years. So I can imagine that is not the case for everybody. This could be useful but I would look at the picture and not at the list. (okay). (Participant 2)

Another way to condense the instrument without over-simplifying it is to attempt to generalise the checks more and make them less context-specific, as Participant 1 suggested:

I think you could even maybe generalise that a little bit more because else you're going to get specifics which you might not want in a checklist like that, but you could say that for example for performance, whatever system you have, there's always a front-end and that front-end needs to have a certain performance. On the other hand, some systems are transaction-based and they have a need for a certain architectural set-up for their transactions to be able to be done in a certain time. Batch processing, or whatever. That's a specific thing that not every -ehm-system needs but it's a decision you need to make for every system. (Participant 1)

Redefining a check

If one were to study the last sentence, an other insight becomes clear. It is not that every SRS needs a requirement to state how responsive the interface should be, but in every SRS evaluation the decision needs to be made whether what is described in the SRS regarding responsiveness is satisfactory. In other words: the fact that the decision has to be made is still important, even though the check itself might be rated as 'Not Applicable'. A solution here is to not see the checks as 'this is what the SRS should adhere to', but to see the checks as: 'given this check and the context of the Software Project, is what you're seeing in the SRS to your satisfaction'? In other words: all judgement is left to the practitioner, the instrument is just a tool to support their process and evaluation.

⁸The Type Completeness check

Splitting up the instrument

Instead of further condensing the instrument and removing low-detailed information in favour of high-level information, possible solution would be to make the instrument dynamic. Dynamic in the sense that the focus lies on the high-level aspects but that, when desired, additional information or checks could be found. To keep the checklist printable, the advice from Participant 2 was followed:

I think like I said the checklist itself should be 3 pages. Ehm and preferably in some way that you can actually use it. And I would put all the other information, which is really valuable, especially for somebody who has to do this for the first time, and also for a company like us, where you would like to sort of standardise the way in which people do it, agree on the things you are going to check. Ehm. Then it's very usable but I would kind of separate it. (..) So maybe you should sort of split it up. Ehm, in kind of a background, or handbook like things. How do you do it? And then actually, having the list, eh, yeah separately in a table. (Participant 2)

5.5 Changes incorporated to the instrument resulting the interview sessions

- The instrument was split up into a checklist and a handbook
- Checks were no longer seen as 'this should be done' but rather: 'see if this element is satisfactory based on your judgement'
- A checklist-sheet was created with the ratings 'Satisfactory', 'Not satisfactory' and: 'Not Applicable'
- Suggestions for improving the contents were adapted into the checklist

Chapter 6

Live Study Validation at REFSQ'18

6.1 Background

For the final step in the validation of the instrument, a live study was designed specifically for REFSQ 2018 to have the instrument evaluated by a group of experts. The proposal [21] was accepted and the conference provided for a unique opportunity to have the instrument assessed by a group of highly experienced researchers and practitioners.

6.1.1 About REFSQ

The Requirements Engineering: Foundation for Software Quality (REFSQ) is a working conference held each year at a different location. In 2018 the conference was held at Utrecht, the Netherlands, which formed the perfect opportunity to validate our instrument and introduce the research to the scientific community. Additional information taken from the REFSQ 2018 home page:

The REFSQ working conference is the leading European conference series on requirements engineering. It is the goal of REFSQ to foster the creation of a strong European RE community across industry and academia. Since 1994, Requirements Engineering continued to be a dominant factor influencing the quality of software, systems and services. REFSQ seeks contributions which report on novel ideas and techniques that enhance the quality of RE's products and processes, as well as reflections on current research. REFSQ has a long tradition of being a highly structured and interactive forum. The special format of each session includes the three roles presenter, discussant, and session chair, yielding into highly interactive sessions, filled with discussions among the excellent researchers and industry experts. In addition to the high-quality scientific program, REFSQ also features a highly interactive Industry Track, Workshops and a Research Methodology Track. [60, About REFSQ]

6.2 Study design

6.2.1 Research objectives and research questions

The main goal of the live study was to get a hands-on impression from experts in the field and assess the quality of the instrument. A side benefit of the live study was the announcement of the research to the scientific community and to gain traction among it. This live study served as the final validation step of the artefact and helps us anticipate the effect of its use in the described problem context.

To this end, we asked the following research questions:

RQ1 What is the quality of the artefact?

RQ2 What is the anticipated effect of applying the artefact in practice? 14

6.2.2 Methodology

For the live study the empirical evaluation methodology described by [75] was followed. This methodology is evaluation of any software engineering artefact or technology in context. In our evaluation study, participants will be provided with the materials mentioned in ref and asked to validate a sample SRS with the help of our proposed instrument. Participants were invited to share their perceptions and experiences by completing a questionnaire. The quality of the instrument will be assessed based on criteria mentioned in section refquestionnaire, rated by the participants in a questionnaire. This data is analysed with an extended version of the mixed method approach by [54]. The checklist consists of 112 checks. To cover all checks within the time-span of the session they are split into a 'core'- and four 'non-core' segments. The core segment consists of 30 checks and the remaining checks are evenly distributed into four groups. Each participant will cover the (core) segment and one of the (non-core)-segments. During the live session it was made sure that the four variations of materials were distributed evenly. In order to be confident in the outcome of the live study it was determined to aim for a minimum of 20 participants in order to have at least 5 participants working on each non-core segment of the checklist.

6.2.3 Logistics

Due to the size of the conference, a maximum number of 50 participants was anticipated. The study participants were to be drawn from the pool of attendees at REFSQ 2018 and the organisation would help promote the live study. No particular participant profile was required and all participants were to jointly attend the live study. Participants were to be provided with seating and ample space to write and read.

Timetable

- 00 - 05 min** Introductory presentation
- 05 - 10 min** Distribution of materials, seating
- 10 - 70 min** Test sessions applying the checklist
- 70 - 80 min** Filling in of questionnaire
- 80 - 90 min** Ending, gathering of materials, thanking

Equipment and materials involved in the study

- A presentation room large enough to seat 50-60 people provided for by the organisation
- A beamer and audio equipment provided for by the organisation
- A Microsoft PowerPoint presentation used during the introduction of the live study
- A sign up sheet where participants could leave their contact details if they desired to be kept informed about the research.

- 60 sets of materials to be handed out to each participant, consisting of:
 - The checklist handbook consisting of every check + description in full (Appendix A)
 - One of the four types of single-page checklist sheets (Appendix C.1.1)
 - A sample Software Requirements Specification
 - A post-use questionnaire

Participant instructions

The participants were asked to perform the following tasks:

- Get familiar with the given set of materials
- For each of the non-greyed-out check on their checklist sheet:
 1. If desired, look up extra information in the handbook
 2. If desired, look at the sample SRS to help envisioning a real-life scenario
 3. Envision how you would, and if you could, perform this check
 4. Write down issues you might have for this specific check on the checklist sheet
- Assess the instrument by filling in the post-use questionnaire. Write any recommendations for improving the instrument here.

6.2.4 Course of the live study

During the conference, a time slot of 90 minutes and a room equipped with presentation equipment (beamer, microphone) and ample chairs for our participants were provided for. The room lay-out did not allow for the participants to use a table, but using the materials on their laps still allowed the participants to use the given materials adequately. The session consisted of:

15 min A presentation describing the research so far, the logistics and goals of the session followed a short Q&A clarifying any remaining questions. The different sets of materials were distributed evenly among the participants.

60 min The session itself, where participants were asked to quickly browse through the materials and envision having to apply this checklist in a real-life scenario. The requested way to work was to:

Participants that finished were given the post-use survey

10 min Filling in of the post-use survey

10 min Q&A, the sharing of opinions and experiences, future plans, and thanking of the participants.

Including one participant joining the session halfway in, every participant managed to finish the checklist within the allocated time span. Roughly 20 participants were present during the live study, the large majority being senior researches in the RE field.

The study participants were drawn from the pool of attendees at REFSQ 2018 and no particular participant profile was required. The live study was promoted by the

organisation in the materials and schedule distributed to each conference attendant. Unfortunately less than participants than anticipated attended the live study. The materials as described in 6.2.3) were prepared for before the start of the live study and the four type of sets were stacked alternating (to assure an even distribution of the different type of materials). After the introductory presentation a small Q&A took place to resolve anything left unclear. The participants were handed out one of the four sets of materials to assure that the different type of sets were distributed evenly. The participants all finished within the available time and no specific concerns were raised. One participant joined the session 20 minutes late but managed to finish in time. During the Q&A interest was shown by the participants and the general response was positive. No notes or recordings were taken of this part of the session. The participants were thanked and asked to hand in their materials and, if they desired, leave their contact details at the sign up sheet on their way out. Any materials left on the seats, as well as the sign up sheet were collected after the participants had left the room.

6.3 Results and analysis

6.3.1 Gathering of the results and handling of the data

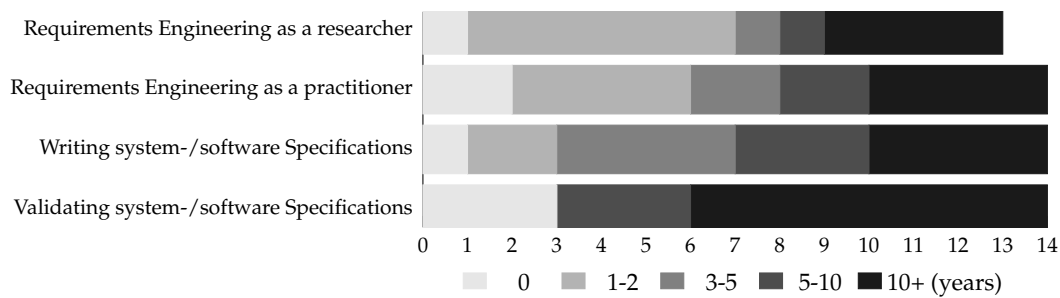
The live study resulted in 14 sets of filled in post-use questionnaires and 18 sets of the single-page checklist sheets containing varying degrees of written feedback. The collected sign up sheet naturally contained personal information but this was processed separate from the documents used for analysis. Apart from the person's hand writing, none of the gathered documents that were used for the analysis contained any personal information and were, as such, not traced back to the individual. All data and materials collected was in paper form. Data resulting from the live study was copied into, and analysed using Microsoft Excel. Any data that was copied over was checked at a separate instance to verify correctness. Tables and graphics that were generated were either included in this document as an image directly or copied over in L^AT_EX format. Any data generated was stored on a secure PC and shared to Google Drive for data backup purposes.

6.3.2 Categorisation of collected data and their purpose

The data resulting from the checklist sheets can be classified as follows:

1. Data resulting from the checklist sheets...
 - Quantitative data...
 - (a) indicating issues with specific checks
 - Qualitative data...
 - (b) written statements
2. Data resulting from the Post-Use Questionnaire
 - Quantitative data...
 - (a) indication of participants' experience
 - (b) quality assessment of the instrument
 - (c) assessment of expected effectiveness of applying the instrument in a problem context

FIGURE 6.1: Distribution of participants' indicated experience measured in years



- Qualitative data...
 - (d) addressing strengths, weaknesses
 - (e) stating missing items and recommendations for improving the checklist

6.3.3 Usage and purpose of the data collected

The collected data [1a](#), [1b](#) and [2e](#) were used to improve the instrument. The data from [2a](#) is used to assess the experience of the participants. The data from [2b](#), [2d](#) and [2e](#) is used to answer RQ1 (see [6.2.1](#)). The data from [2c](#) is used in answering RQ2.

6.3.4 Participants' experience within the domain

Figure [6.1](#) illustrates the distribution of the participants' indicated experience with the stated concept. To the question asking the participants to rate their experience with Requirements Engineering as a researcher, one participant did not provide an answer, which is why for this question only 13 responses are displayed.

6.3.5 Quality of the Instrument

Quantitative data

The quantitative results consist of a set of ratings regarding to the 8 criteria by Stufflebeam [[67](#)]. Table [6.1](#) displays the results from both quantitative data sets. Presumably due to ambiguity of what the 8 criteria exactly entailed, some participants opted to not rate the checklist on all 8 criteria, resulting in some criteria having between 10 and 14 actual number of responses.

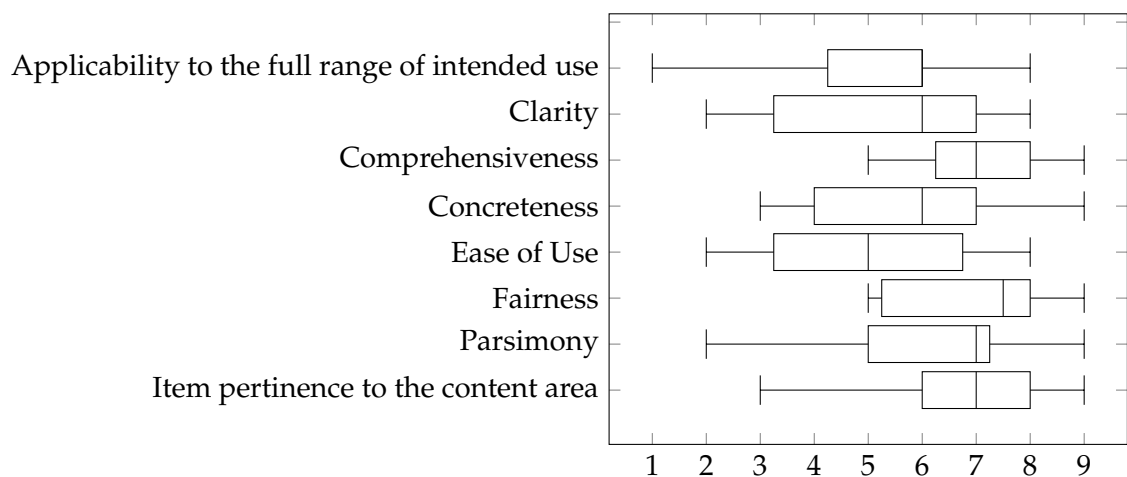
Qualitative data

The qualitative data consisted of written statements describing the Strengths, Weaknesses, Missing Items and Suggestions for Improvements for/of the checklist. The entire written response to one of these four questions made up the context unit. Out of these context units, single words or phrases describing a specific idea were divided into recording units. Of these 60 units, 19 were listed as key strengths of the checklist, 21 as key weaknesses, 4 as items missing, and 16 as ways to improve the checklist. Out of these context units, the written statement of one weakness was deemed incomprehensible and ignored during the analysis. In an effort to compare

TABLE 6.1: Results of Critical Feedback Survey questions set 1-a

	M	SD	n.o. responses
Applicability	5.43	1.74	14
Clarity	5.43	2.10	14
Comprehensiveness	7.21	1.19	14
Concreteness	5.85	1.77	13
Ease of use	5.07	1.94	14
Fairness	6.90	1.52	10
Parsimony	6.25	1.96	12
Pertinence	6.85	1.63	13

FIGURE 6.2: Box plot of the results displayed in Table 6.1



the quantitative results with the qualitative results, each of the remaining recording units was classified as one of the eight criteria mentioned above and a net strength score was determined. The Net Strength is defined as the difference between the criteria's normalised frequencies for its weaknesses minus its strengths. The normalised frequency is a proportional value determined by dividing the criterion's frequency of strengths (weaknesses) by the total number of strengths (weaknesses). The net strength provided a ranking of the perceived strengths of the checklist based on the qualitative results. Table 6.2 displays the (normalised) frequencies and net strength of each criteria's strengths and weaknesses.

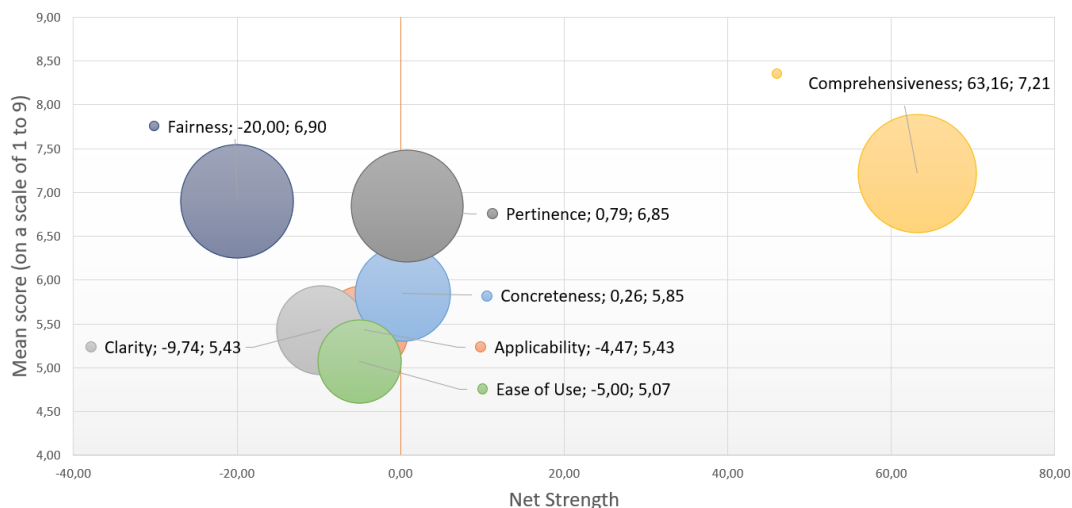
TABLE 6.2: Net strength score

	Strengths		Weaknesses		Net Strength Score (S'-W')
	Frequency (S)	Normalized (S')	Frequency (W)	Normalized (W')	
Applicability	2	10.53	3	3	-4.47
Clarity	1	5.26	3	3	-9.74
Comprehensiveness	12	63.16	0	0	63.16
Concreteness	1	5.26	1	1	0.26
Ease of Use	0	0	1	1	-5.00
Fairness	0	0	4	4	-20.00
Parsimony	0	0	5	5	-25.00
Pertinence	3	15.79	3	3	0.79

Combining the data

As proposed by Martz [54], in an attempt to compare the qualitative results to the quantitative results, the 'mean scores' were plotted against the 'net strength'. The resulting graph is displayed in Figure 6.3

FIGURE 6.3: Bubble plot of the net strength and mean score



Here, *Comprehensiveness* is shown to be a clear outlier for the Mean score as well as the Net strength. Most bubbles display a relatively equal relation between both types of scores with the exception of the *Fairness* criteria. *Fairness* received a relatively high Mean Score compared to its net strength and the other quality criteria meaning it received a comparatively high score for the quantitative ratings as opposed to the qualitative response.

6.3.6 Anticipated effect of instrument application in the problem context

Whilst the first set of questions addressed the quality of the checklist itself, the second set addressed the anticipated impact of using the checklist -as is- in the validation process. Participants were asked to rate the checklist+handbook's anticipated effectiveness based on a set of criteria based on a Likert scale from 1 ("Strongly Disagree") to 9 ("Strongly Agree"). The question was formalised as: "For a real life scenario please indicate whether you expect the checklist + handbook to.."

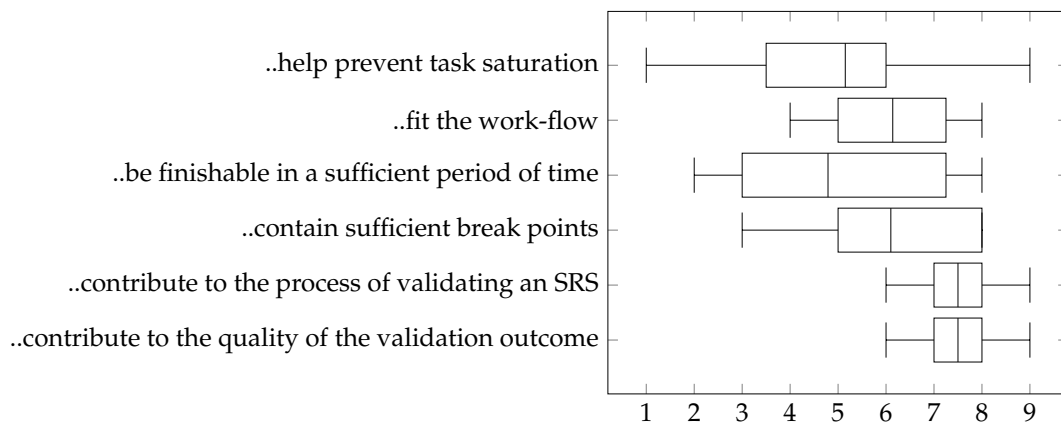
6.3 displays the results of said questions.

TABLE 6.3: Results of Critical Feedback Survey questions set 2

	M	SD	n.o. respondents
Help prevent task saturation	5.15	2.12	13
Fit the work flow	6.14	1.41	14
Be finish-able in a sufficient period of time	4.79	2.22	14
Contain sufficient break points	6.10	1.66	10
Contribute to the process of validating an SRS	7.50	0.94	14
Contribute to the quality of the validation outcome	7.50	0.85	14

Figure 6.4 visualises these results in a box plot where the whiskers denote the min and max scores and the vertical lines of the boxes denote the first-, second- and third quartile.

FIGURE 6.4: Box plot of the results displayed in Table 6.3



6.3.7 Improving the instrument

The collected data from 2e are shown in table 6.5. The data resulting from 1a is attached in Appendix C.1. The list with the list of the top 10 issues with the most indicated faults is displayed in table 6.4. The data resulting from 1b is not analysed, but was directly used to update the instrument (section 6.6).

In table 6.4 we can see that the checks with the most identified faults are due to ambiguity and their redundancy.

The items rated most Inapplicable are 5.c - Stakeholder agreement (6), 4.j - Completeness (3), 1.4.a - Implementation environment (3) and 2.i - Allocation of Requirements. The most ambiguous items are considered 2.b - Structuring (7), 4.1.e - Reflectiveness/trueness (7), 3.1.a - Semantical correctness (5) and 1.5.a - Stakeholder coverage (5).

TABLE 6.4: Top 10 checks with most indicated faults

#	Name	Inapplicable	Ambiguous	Redundant	Unimportant	Contains incorrect information	Total
3.1.e	Reflectiveness / trueness		7	7	2	1	17
4.i	SRS correctness		3	5		3	11
1.1.h	Overview		4	4	2		10
1.1.i	Document structure	1	3	3	3		10
2.b	Structuring	1	7	2			10
1.2.e	Scope of the work	1	4	1	2	1	9
2.e	Traceability		3		3		9
3.1.a	Semantical correctness	2	5	1		1	9
4.j	Completeness	3	1	2		3	9
1.3.a	Product perspective	1	3	4			8

The top items considered Unimportant are 3.3.a - *Syntactical correctness* (4), *Document structure* (3) and *Traceability* (3). Most notably, 7 out of the top 10 items classified as Unimportant were from Chapter 1 - Document structure and contents.

The top items considered containing incorrect information are 2.e *Traceability* (3), 4.i - *SRS correctness* (3), 4.j - *Completeness* (3) and 4.g - *Requirement analysis* (2).

Table 6.5 contains all missing items and overall recommendations written down by the participants on the questionnaire. Several of the proposed missing items would be specific to certain contexts (agile, info systems with bigdata/iot/etc.) but would be worthwhile to consider to add for tailored checklists. One element that is clearly missing in the instrument is an envisioned way to write down defects and should be added to the instrument. There are several mentions of ambiguities which should be addressed as well.

Several of the recommendations address the format of the instrument and the balance of making the instrument suitable to use without becoming too specific or generalised. Some suggestions suggested adding more details (i.e. 'It could be combined with document templates (or enrichen with examples of 'wrong' and 'good' for each document section)', balance the density abstraction levels of the data ('Maybe have some mini description of items in the list vs separate document' or to trim the instrument ('Make it shorter, only really important stuff'). The recommendations are, to a point, contradictory in nature. The 'perfect' balance might therefore not exist and applying the instrument in practice would hopefully provide valuable information in what can be considered 'good enough'.

6.4 Discussion

In advance of the conference, a higher attendance than the actual number of participants (18) was anticipated. Despite this, based on the indicated experience of the participants (Figure 6.1) and the prestige of Europe's leading RE conference the participants are regarded as experts in the field of RE and confidence is placed in the quality and validity of the participants' given responses.

At the start of the live study session, several participants indicated to be confused by the goal of the session and the materials that were provided. These questions were addressed and answered before the sessions started and the participants indicated having no further questions. Based on written feedback statements on the checklist sheet however, it seemed that some participants were under the impression that the checklist sheet was meant to validate the sample SRS, as opposed to validating the instrument. This meant that data resulting from the checklist sheets proved to be less valuable than anticipated.

The allocated time-slots of the live-study session did not allow for each participant to analyse the entire instrument. To mitigate this, four different sets of the

instrument were created where each set contained the 'core' set of checks and a selection of the 'non-core' set of checks. This meant that no participant analysed the checklist in its entirety but rather assessed the quality of the checklist based on their selection of checks and a general impression of the instrument. In an attempt to mitigate this problem, the (non-core) checks that weren't included in a specific set, were greyed out on the checklist sheet but still readable. Because of this, each participant was considered to still be able to get a good general impression of the overall instrument. This did mean that certain checks were only closely inspected by a fourth of the total number of participants. The classification of 'core' and 'non-core' were based on intuition by the author, indicating a risk that certain checks that were inadvertently categorised as 'non-core' did not receive the desired exposure. At the end of the session, no participant indicated requiring additional time. 18 sets of (at least partly) filled in checklist sheet were collected whilst only 14 sets of questionnaires were filled in. Some participants were seen leaving the session early, but their reasons for doing so are unclear.

In analysing the quantitative results of the questionnaire, it was observed that in four instances not all of the 8 validation criteria were rated. In some cases a question mark was written instead. This indicates that some participants did not fully understand the definition of certain criteria, despite being provided with a short description in the form of a footnote. For the participants who did rate these criteria, their interpretation might differ from the researcher's interpretation.

In analysing the qualitative results, the vast majority (12 out of 19) of written Strengths could be attributed to '*Comprehensiveness*'. Due to an unevenly distributed Strengths attribution and the small sample size, written Weaknesses that were categorised into other quality criteria had a relatively large influence on the net strength score. As a result, most quality criteria received a negative Mean score whilst receiving a decent net strength score (>5 out of 9). Furthermore, if one were to rank the criteria based on the net strengths, one written statement could affect the ordering. Although the combined illustration provides a good general impression of the strengths and weaknesses of the instrument, the small sample size as well as the unequal distribution of the written Strengths and Weaknesses statements call for caution when interpreting the criteria's Net strength as well as the combined results.

Based on the quantitative results, the instrument is considered to have scored well (>6.5) for Comprehensiveness, Pertinence and Fairness, and decently (>5) on the criteria Concreteness, Applicability, Clarity and Ease of Use. Based on the qualitative results, the checklist scored high (>60) for Comprehensiveness, and adequate (>0) for Pertinence and Concreteness, slightly below 0 for Applicability and Ease of Use, and well below 0 (-20) for Fairness and Parsimony. Comprehensiveness was regarded highly by both the qualitative and the quantitative responses. Ease of Use and Clarity scored low on both type of responses, indicating possible improvements to be made on these aspects. Fairness is a clear outlier as it is in the top 3 for the quantitative responses, but absolute bottom for the qualitative responses.

Although the instrument's comprehensiveness is seen as a positive aspect, it might also negatively impact the usability of the checklist. Written weaknesses included: 'Depending on the project, this completeness can be a hazard: many subjects can be skipped', 'Some questions seem difficult to answer or to answer in sufficient time with enough confidence', 'I'm not at all sure that 'one size fits all' here - might work on some programs and not on others' and that the checklist is 'too generic', 'too ambitious', 'too complex' and 'too long'. Despite the overall positive results in the post-use questionnaire, these results indicate concerns for the usability of the

checklist and applicability to a variety of contexts. What might be overly complicated and too long in one context, might be too generic for an other. Further research investigating the usability and applicability of the instrument to different contexts are deemed important, as well as investigating ways to mitigate this, i.e. by adapting the instrument to specific contexts.

In predicting how the artefact will perform in practice (RQ2), one has to consider possible differences between the applicants (the Expert Group) and the target audience (practitioners in the field). Although most applicants indicated having significant experience in research as well as in practice, a certain bias might exist in the social group that a conference attracts. As discussed in section 3.2.6, there are indications of differences between what researchers and practitioners find important in validating SRSs but the difference in experience of the practitioner might also be a factor to consider. The Strong and Weak points that were expressed by the (highly experienced) participants might not correspond to the assessments of other, more novice, practitioners.

Recommendation 1 in 6.5 mentions that there are checks in this instrument that are not in the IEEE template and addresses that the instrument goes beyond. This is indeed the case as the IEEE templates mostly specifies what the SRS should contain. The author is of the opinion, that as someone *evaluating* a SRS you have a duty to assert not only that the content of the SRS is not in conflict with any of the industry standards, but also to (at least some degree) evaluate whether the document makes sense given the project setting and is an adequate reflection of the stakeholders involved. Ultimately, the choice is left up to the evaluator whether he or she finds the contents of the SRS satisfactory or not.

6.5 Conclusion

6.5.1 RQ1: What is the quality of the artefact? (KQ 8)

Based on Stufflebeam's 8 quality criteria for evaluation checklists and Martz's mixed method design, the artefact is considered to be of acceptable to good quality. Based on the results from the quantitative analysis, the checklist's strengths are its *Comprehensiveness* (M=7.21¹), *Fairness* (M=6.90) and *Pertinence* (M=6.85). The respective weaknesses are its *Ease of use*, the *Applicability* (M=5.43) and its *Clarity* (M=5.43). Based on the results from the qualitative data analysis, the checklist's Strengths are its *Comprehensiveness* (with a Frequency of 12 out of 19) followed by its *Pertinence* (3 out of 19) and *Applicability* (2 out of 19). Weaknesses are the *Parsimony* (with a Frequency of 5 out of 20) and *Fairness* (4 out of 19). Overall, the artefact is seen as a comprehensive tool and a good starting point to guide a practitioner in evaluating SRSs. Indicated challenges are its usability, applicability and clarity.

6.5.2 RQ2: What is the anticipated effect of applying the artefact in practice? (PG 1)

Based on the analysis described in section 6.3.6, the instrument is anticipated to have a positive impact on both the validation process (M=7.50², SD=0.94) as well as the validation outcome (M=7.50², SD=0.85). The checklist is deemed to have sufficient break points (M=6.10, SD=1.66) and to fit the work-flow (M=6.14, SD=1.41) although it is unclear if it will be finish-able in a sufficient period of time (M=4.79, SD2.22). The

¹based on a scale of 1 ('Strongly Disagree') to 9 ('Strongly agree')

last result indicates a possible threat to **Requirement 5**. Although the overall results regarding the quality of the artefact were positive, the results of applying the artefact in practice might vary based on the software context. Based on the written recording units, there is strong indication that the artefact might be suitable for some type of projects, but not for all.

6.6 Changes to the instrument

- Chapter titles and check altered in an attempt to make them clearer.
- Several ambiguous texts were corrected.
- Check 1.1.i - *Document organisation* renamed to *Document structuring*.
- Check 2.b - *Structuring* removed due to ambiguity and redundancy with checks *Modifyability* and *Rationale*.
- Checks 3.1.d - *Relevancy* 3.1.e - *Reflectiveness / trueness* and check 3.1.f - *Currency* combined into check *Relevancy / currency*.
- New check to section 3.1 added: *Necessity*.
- Check 3.3.f - *Specificness* renamed to 3.3.f - *Level of abstraction / detail*
- Checks 4.1.k - *SRS correctness* and 4.1.l - *Completeness* are combined into check 4.1.k - *Completeness & Correctness*
- Section 3.1 *Requirement correctness* renamed to *Requirement impact*.
- Section 3.2 *Requirement description* renamed to *Requirement detailing*.
- Section 3.3 *Requirement wording* renamed to *Requirement phrasing*.
- Option *N/A* added to the checklist sheet.
- Field to write down additional information added to the checklist sheet.

TABLE 6.5: Recommendations from the Questionnaire

Missing items	
a	Key value check, making it possible to work agile
b	System projects (hardware/software). This is limited to info systems with BigData, IOT, cloud, it may not work as well.
c	Describe at the beginning how to write down defects this is important for the checking process
d	What about attributes of the requirements themselves? (Not sure that all the categories are always needed)
Recommendations for improving the checklist	
a	Maybe have some mini description of items in the list vs separate document
b	Use different (more clear) name of titles
c	System projects (hardware/software). This is limited to info systems, with BigData, IoT, cloud, it may not work as well
d	Make it shorter, only really important stuff
e	Make sure you avoid ambiguities.
f	Examples?
g	It could be combined with document templates (or enrichen with examples of 'wrong' and 'good' for each document section
h	Mandatory v.s. non-mandatory (good to have) could be {incomprehensible}
i	Checklist might be customisable to project type (e.g. system eng. Software app., etc)
j	A checklist for a product SRS should be different from a bespoke SW SRS. Here the {incomprehensible} product/SW/{incomprehensible} are used confusingly.
k	Separate different checking process. A) Items which can be checked automatically. B) Items with semantic checks an individual sections. C) Items with semantic checks over the whole document
l	Several things not typically in SRS: rationale, types, stakeholder agreement. I don't think it will work well in combination with IEEE template SRS. I have to expand SRS structure + practice at the same time. You go beyond what is required in the templates
m	Provide "n/a" - not all items are applicable in all contexts
n	Checklist should be structured along the chapters of the SRS template
o	Define "Internal consistency"
p	Define "correctness"

Chapter 7

Recommendations, conclusion & future work

7.1 Reflection on the research process

The Design Science Methodology introduced by Wieringa [75] proved very valuable in shaping, scoping and compartmentalising the research direction as well as providing for a structure to follow in writing the thesis. Although initial effort was required to fully grasp the rationale, concepts and abstractions described by Wieringa, the experience of iterating through the design cycle, combined with the methodology, has provided the author with many new insights and abstract lines of thinking. For that, the author is thankful.

In retrospect, the author would like to have spent additional effort 1) conducting and solidifying the problem investigation and 2) identifying and comparing additional treatments (primarily in industry and published books) before the decision was made to design a new artefact. For this particular design effort, additional opportunities to observe and interview practitioners would have been of great value in developing a practical tool.

The methodology introduced by Martz [54] to validate evaluation checklists using a mixed method design which was applied during the live sessions showed to be a useful, low-cost method of identifying Strengths, Weaknesses and Missing Items as well as providing for quantitative data to validate the checklist based on a series of (8) quality criteria. In the mixed method, the written feedback statements were attributed to one of the 8 quality criteria and a 'Net Strength' was calculated. Because the vast majority (12 out of 19) of written Strengths could be attributed to '*Comprehensiveness*', the distinction between the other criteria is based on a very small data size. Based on written feedback on the questionnaires it seems that some participants had trouble understanding some of the criteria (i.e. '*parsimony*'), despite being provided with a small description on the questionnaire. Future researchers are recommended to 1) assure a large enough sample size and 2) provide detailed descriptions of the quality criteria.

7.2 Challenges faced during the research

The research ended up taking longer than the standard time frame due to multiplicity of factors including health issues experienced by the author, the company discontinuing their involvement due to the contact person leaving his position, and further research delays. In this time frame, Dr. Maya Daneva took over as official supervisor whilst original supervisor, Nazim Madhavji, remained an unofficial supporter via his position at the Western University. The discontinuation of the company's

involvement meant that the original plan, to validate the instrument in a realistic setting with expert-practitioners, was no longer an option. Under guidance from Dr. Maya Daneva a back-up plan was instigated to validate the instrument during a Live Study session at the REFSQ 2018 conference in Utrecht, the Netherlands.

7.3 Recommendations for practitioners

The overall effectiveness of the checklist is predicted to vary depending on 1) the expertise of the applicant, 2) the properties of the requirements specification and 3) the context of the software project. In the live study, most participants indicated having multiple years of experience with RE as both a researcher and a practitioner. The checklist plus accompanying handbook was designed with both experienced and inexperienced users in mind. Although no significant barriers in using this instrument with little experience are anticipated, further research is needed assess how the experience of the applier affects the quality of the specification's evaluation. Depending on the context, a subset of the checklist tailored to the specifics of the context is foreseen. In other cases, such as when dealing with safety-critical systems, additional and more specific checks could be required.

7.3.1 State of the checklist

The checklist has shown promising results (6) but has not been used in practice. This means that practical challenges are likely to come up and that users runs the risk of the instrument needing to be adjusted to match your company's c.q. project's specific requirements. Nevertheless, the checklist is expected to be a valuable tool and a good starting point to base your own checklist from, if so desired.

7.3.2 Adjusting the checklist to context

It is deemed important to adjust the evaluation process (how much effort is put into evaluation which aspects) and -outcome (how you value the SRS) on the context. Differences in context include the software development methodology (i.e. agile v.s. traditional approaches), the domain (i.e. Web Development v.s. Security systems), the stage of the development life-cycle (at the very start, or at a later stage) and for example whether tendering is involved. For tendering projects and situations where there is a large communication gap between the developers and the stakeholders (i.e. outsourcing) more dependency is placed upon the quality of the SRS. To illustrate, for a software project for the space domain that is developed in a more traditional approach one would expect a very complete SRS before the development process starts and in evaluating the SRS a significant effort on assuring completeness of the requirements specification is to be expect. Contrast this to an agile (c.q. hybrid) approach where only the major requirements are written down at the start and coding begins before all requirements are crystallised and uncertainties are addressed, and the differences in the evaluation effort as well as how to evaluate the different types of SRSs becomes clear. This evaluation checklist is primarily developed with Information Systems (e.g. Government Information Systems) and more traditional approaches (e.g. waterfall, or gated and milestone-driven development processes) in mind but can serve as a starting template. Modifications to the checklist to create a better fit with the context are greatly encouraged. Examples include

- to omit or add additional checks,

- to rate or prioritise checks based on their importance or allotted time and
- to provide additional guidelines for evaluators to base his decision on (i.e. a 'scoring system').

7.3.3 Required expertise to use the checklist

Evaluating Software Requirements Specifications is seen as a task that will partly rely on the expertise of the reviewer. The checklist is designed to be used by a variety of expertise levels. Experts will most likely only use the checklist sheet, or occasionally consult the handbook and for novice evaluators reading the handbook and further references c.q. reading materials might prove be of value. Despite the checklist making recommendations regarding what constitutes a quality SRS, the judgement and final verdict is left to the reviewer. This means that the evaluation -and the quality of the evaluation- is still dependent on the expertise of the reviewer. We expect the checklist to be usable by-, of be of use to all types of users, but expect the quality of the evaluation to still be reliant on the expertise of the reviewer.

7.4 Future research

Artefact evaluation

This work could and should be extended in multiple directions. Below, the authors describes three areas where more future research will be beneficial for the commonality of practising requirements engineers and for RE researchers. The main focus point for future research would be to evaluate the artefact, e.g. to observe and analyse the application of the artefact in practise and to determine the right balance between the extremes of idealisation and practice [76]. Despite the overall positive results in the post-use questionnaire, the checklist's level of comprehensiveness and the time required to perform each test poses a real challenge to the effectiveness of the checklist. For future research, an effective trade-off between the comprehensiveness, applicability and usability of the checklist needs to be investigated for different contexts. We foresee to address the mentioned weaknesses by taking subsets of the checklist and tailoring the checklist to the specific needs for the given software project.

Based on [75, Table 4.1, 5.1] the next steps would be to investigate:

- whether the stated requirements are met
- what effects using the artefact has on the SRS evaluation process and -outcome
- whether these effects contribute and/or detract from the stakeholder goals
- understand why these effects contribute and/or detract in that way
- what the trade-offs are compared to other artefacts
- the sensitivity for different contexts

Based on newly obtained understanding, the artefact can be altered, or 'forked'¹ into different versions tailored to specific contexts.

¹[https://en.wikipedia.org/wiki/Fork_\(software_development\)](https://en.wikipedia.org/wiki/Fork_(software_development))

Additional knowledge questions

Certain additional knowledge questions came up during the development of the checklist that fell outside the scope of the research. Incorporating the new understanding would benefit the quality of the artefact.

- How does SRS validation and -evaluation actually done in industry relate to academic literature and best practices?
- In what way do contextual variances affect what is considered a quality SRS?
- In what way do contextual variances affect the SRS evaluation process?

Updates

In November 2018, a new version of the ISO/IEC/IEEE 29148 standard [42] was released. Changes should be adopted to the checklist.

Appendix A

SRS Evaluation Checklist

A.1 Evaluation sheet

SRS evaluation sheet		v3.2	Satisfactory	Unsatisfactory	N/A	Comments
1 Content and outline of the report						
1.1	Document structure and references					
a	Document details					
b	Table of contents					
c	Index					
d	Glossary					
e	Appendices					
f	Numbering and referencing					
g	List of references					
h	Overview					
i	Document structuring					
1.2	Understanding of the business the product will operate in					
a	Product purpose					
b	SRS purpose					
c	Goals of the project					
d	User business					
e	Scope of the work					
f	Business Environment					
1.3	Understanding of the project aspects and the product's responsibilities					
a	Product perspective					
b	Product functions and -functionality					
c	Performance indication of functions					
d	Identification of Limitations and Constraints					
e	Assumptions and dependencies					
f	Risks					
1.4	Understanding of the product environment					
a	Implementation environment					
b	External interfaces					
c	Partner or collaborative applications					
1.5	Understanding of the stakeholders					
a	Stakeholder coverage					
b	Stakeholder information					
c	User characteristics					
2 Requirements' Structure and Content						
a	Organisation					
b	Structuring					
c	Numbering consistency					
d	Modifiability					
e	Traceability					
f	Cross-referencing					
g	Categorization and ranking					
h	Apportioning of the Requirements					
i	Allocation of Requirements					
j	Degree of implementability					

	Satisfactory	Unsatisfactory	N/A	Comments
3 Specification of Individual Requirements				
3.1	Requirement impact			
a	Semantical correctness			
b	Internal consistency			
c	External consistency			
d	Relevancy / currency			
e	Necessity			
f	Feasibility/achieveability			
3.2	Requirement detailing			
a	Rationale			
b	No design projection			
c	Verifiability			
d	Validatability			
e	Mentioning of affecting assumptions			
f	Assertions			
g	Isolation			
h	Cohesiveness			
3.3	Requirement phrasing			
a	Syntactical correctness			
b	Unambiguous terminology			
c	Unambiguous meaning			
d	Understandability			
e	Preciseness			
f	Level of abstraction / detail			
4 Validity of the document as a whole				
4.1	Wording of the contents			
a	Quality of writing			
b	Conciseness			
c	Coherence			
d	External document referencing			
4.1	Validity of collective requirements			
a	Requirement types coverage			
d	Non-Redundancy			
e	Consistency of descriptions			
f	Internal consistency			
g	External consistency			
h	Justification of decisions			
i	Requirements analysis			
j	System feasibility, validity & achieveability			
k	Correctness & Completeness			
5 Context surrounding the document under review				
a	Joint preparation			
b	Usage of standards			
c	Stakeholder agreement			
6 Absence of unwanted elements				
a	Project requirements			
b	Design or implementation details			
c	Ambiguities			
d	To be Determined'			

A.2 Handbook

Software Requirements Specification Checklist

Handbook

version 3.2

MARTIN DE LAAT
University of Twente
July 4, 2019

Contents

1	Document structure and contents	2
1.1	Document structure and references	2
1.2	Understanding the business the product will operate in	2
1.3	Understanding the project aspects and the product's responsibilities	3
1.4	Understanding the product environment	3
1.5	Understanding the stakeholders involved	4
2	Requirement structure and contents	5
3	Specification of Individual Requirements	6
3.1	Requirement impact	6
3.2	Requirement detailing	6
3.3	Requirement phrasing	7
4	Validity of the document as a whole	8
4.1	Wording of the contents	8
4.2	Validity of the collective requirements	8
5	Context surrounding the document under review	10
6	Absence of unwanted elements	10
	References	11

1 Document structure and contents

A good requirements specification is a structured document steadily introducing the reader to the environment in which the system-to-be-developed will be built

1.1 Document structure and references

- (a) **Document details** *Sources and further reading:* [8]: 9.2.1, [3]: 2.17

Are the title, optional sub titles and version of the document properly specified?

- (b) **Table of contents** *Sources and further reading:* [8]: 9.2.2

Is a table of contents (including a list of figures and tables) properly specified?

- (c) **Index**

Is an index specified?

- (d) **Glossary** *Sources and further reading:* [8]: 9.2.3 & 9.2.5, [7]: 5.1.3

Is a glossary properly specified? Ideally, each term that might be misinterpreted and every acronyms and abbreviations used should be described or referred to.

- (e) **Appendices**

In case Appendices are included, are they properly identified as such and does the SRS explicitly state whether or not the appendixes are to be considered part of the requirements?

- (f) **Numbering and referencing** *Sources and further reading:* [8]: 9.2

Are all relevant items numbered and/or referenced to in a proper way? For example..

- A version number of the SRS
- Page-, figure-, table- and appendixes numbering and naming
- Units of measures where required

- (g) **List of references** *Sources and further reading:* [8]: 9.2.4, [7]: 5.1.4

Is a list of references properly specified? For example, the list of references could consist of

- A complete list of all documents referenced elsewhere in the SRS
- The identification of each document by title, report number (if applicable), date, and publishing organisation
- A specification of the sources from which the references can be obtained

- (h) **Overview** *Sources and further reading:* [7]: 5.1.5

Is an overview /introduction to the document present? For example, it could describe:

- what the rest of the SRS contains
- how the SRS is organised

- (i) **Document structuring** *Sources and further reading:* [3]: 2.23

Is the document organised and structured in a suitable way? Can readers easily locate information and are logical relationships among adjacent sections apparent? Is the document in line with the company's prescribed specification template?

1.2 Understanding the business the product will operate in

- (a) **Product purpose** *Sources and further reading:* [8]: 9.5.1, [7]: 5.1.1, [10]: 1

Is the product's purpose properly described? It could contain:

- A delineation of the purpose of the SRS; the business problem/opportunity the product addresses
- An explanation of how the product is intended to solve the problem

- (b) **SRS purpose** *Sources and further reading:* [8]: 9.5.1, [7]: 5.1.1, [10]: 1

Is the intended use, -purpose and -audience for the written SRS properly described?

- (c) **Goals of the project** *Sources and further reading:* [8]: 9.3.6, [10]: 1b

Are the goals of the product properly described? For example, it could describe:

- The business results to be obtained through or by the proposed system
- A description of what (measurable) advantage the product will bring to the overall goals of the work and the added value to the business and/or to the customer

- (d) **User business** *Sources and further reading:* [8]: 9.3.3, [10]: 1a

Is the business c.q. background of the project properly explained? For example, it could contain:

- A description of the business being done
- A description of the business context
- Internal and external environmental factors that should be taken into consideration in understanding the business

- A description of the situation that triggered the development effort
- A description of the work the user intends to do with the delivered product
- The motivation for the project
- A description of the business problem / -opportunity
- Internal and external environmental factors that should be taken into consideration in understanding the business
- Relevant internal divisions and external entities of the business domain concerned and how they are interrelated

(e) **Scope of the work** *Sources and further reading:* [7]: 5.1.2, [10]: 6&8a, [8]: 9.3.2

Is the scope of the work properly identified? For example, it could contain:

- An identification of the software product(s) to be produced by name
- The current situation and context of the work
- An explanation of what the software product(s) will, and, if necessary, will not do
- The boundaries of the work that you need to investigate to be able to build the product
- A description of the application of the software being specified, including relevant benefits, objectives, and goals
- A clear definition of the boundaries of the work that you need to investigate to be able to build the product (via i.e. a context diagram)
- A description of how the business area fits into its environment

(f) **Business environment** *Sources and further reading:* [8]: 9.3.5

Are relevant internal- and external environmental factors properly specified?

- The entity by name
- The way they influence system
- Possible influences to the business and consequently the system. Examples are market trends, laws and regulations, social responsibilities, and technology base

1.3 Understanding the project aspects and the product's responsibilities

(a) **Product perspective** *Sources and further reading:* [8]: 9.5.3, [7]: 5.2.1

Is the system's product perspective and relationship to other related products properly specified? If the product is independent and totally self-contained, is this clear? If the product is part of a larger system, are relations between relevant requirements of the larger system and the functionality of the product properly identified? And the interfaces between the two? Furthermore, the product perspective could describe how the software operates within the following constraints: the System-, User-, Hardware-, Software- and Communications interfaces, Memory, Operations and the Site adaptation requirements.

(b) **Product functions and -functionality** *Sources and further reading:* [8]: 9.5.4, [7]: 4.1.a

Is the function of the product properly described? Examples are:

- The overall functionality of the product
- A summary of the major functions that the software will perform
- A list of functions organized in such a way that it is understandable by anyone reading the document for the first time

(c) **Performance indication of functions** *Sources and further reading:* [7]: 4.1.c

Are performance constraints of the various software functions described? What is the speed, availability, response time, recovery time of various software functions?

(d) **Identification of Limitations and Constrains** *Sources and further reading:* [8]: 9.5.6, [7]: 5.2.4, [10]: 3

Are potential limitations or design constraints imposed on an implementation properly identified? Examples include: Regulatory policies, Criticality of the application, Quality requirements, Safety and security considerations and Hardware limitations

(e) **Assumptions and dependencies** *Sources and further reading:* [8]: 9.5.7, [7]: 5.2.5, [10]: 5

Are the essential factors and facts that affect the requirements specified? Examples include: Assumptions made, Dependencies, Relevant facts and Relevant business rules

(f) **Risks** *Sources and further reading:* [10]: 23, [1]: 3.d

Are the risks to the project properly specified? Examples include Technical-, Cost-schedule and Environmental risks and Interaction effects.

1.4 Understanding the product environment

(a) **Implementation environment** *Sources and further reading:* [10]: 7

Is the implementation environment properly specified? Is the technological and physical environment in

which the product is to be installed described? This includes automated-, mechanical-, organisational and other devices, along with the non-human adjacent systems

- (b) **External interfaces** *Sources and further reading:* [8]: 9.5.10, [7]: 5.3.1, [10]: 6c

Are external interfaces properly specified? Is a sufficiently detailed description of all inputs into and outputs from the software system present? As deemed applicable:

- Are all the inputs to the system specified including their source, accuracy, range of values, and frequency?
- Are all the outputs from the system specified including their destination, accuracy, range of values, frequency, and format?

- (c) **Partner or collaborative applications** *Sources and further reading:* [10]: 3b,c,d

Are applications that are not part of the product but with which the product will collaborate described? These can be software, human or hardware and can be within or outside your organisation. This includes open source or Off-the-Shelf Software. Conflicts might occur between future requirement changes or changes to the workings of the external applications

1.5 Understanding the stakeholders involved

- (a) **Stakeholder coverage** *Sources and further reading:* [10]: 2, [8]: 5.2.5

Are all stakeholders deemed important to the product identified? Common examples are:

Client the investor of the product. In case of in-house development this is also the customer

Customer the person intended to buy the product

User the (potential) users of the product

Other Maintenance users, Subject Matter Experts, Subject Matter Experts, Users of a current system, Representatives of external associations, etc.

Furthermore: Are no stakeholders deemed irrelevant mentioned?

- (b) **Stakeholder description** *Sources and further reading:* [8]: 9.3.4, [10]: 2

For each stakeholder, is all desired information specified? Examples are:

- Stakeholder identification (name, job title, organisation)
- What the importance / priority is of this type of stakeholder
- How they will influence the organisation and business
- How they will be related to the development and operation of the system
- Knowledge that the project needs from that stakeholder
- The degree of involvement necessary for that stakeholder/knowledge combination
- Their motivation to be involved with the product
- Agreement on how to address conflicts between stakeholders who have an interest in the same knowledge

- (c) **User characteristics** *Sources and further reading:* [8]: 9.5.5, [7]: 5.2.3

Are relevant user characteristics that may influence usability properly specified? Examples are:

- Educational level
- Experience
- Disabilities
- Technical expertise

2 Requirement structure and contents

This chapter contains checks that, generally speaking, are either done correctly or incorrectly for all requirements and can be done without extensively reading the requirements themselves.

- (a) **Organisation** *Sources and further reading:* [3]: 2.23
Are the requirements organised in a suitable way? F.e. are they grouped in a meaningful way, such as by *class of user, object or feature*?
- (b) **Structuring**
Are the requirements structured in a suitable way?
Is a suitable structure for detailing information being used? Are there no important types of information, such as Rationale, missing?
- (c) **Numbering Consistency**
Are the requirements numbered consistently?
- (d) **Modifiability** *Sources and further reading:* [7]: 4.3.7, [3]: 2.12
Are the requirements structured in such a way to allow for easy modifications? Ideally, the requirements should be structured and styled in such a way so that any changes to the requirements can be made easily, completely and consistently while retaining the structure and style
- (e) **Traceability** *Sources and further reading:* [7]: 4.3.8, [1]: 2.c, [5]: , [3]: 2.11, [8]: 5.2.5
Are the requirements traceable? Types of traceability include:
 - Backward traceability** Each requirement is traceable to its source in earlier documents
 - Forward traceability** Each requirement has a unique name or reference number
 - Business Goal-Requirements traceability** Each requirement is traceable to its business goal
 - System Goal-Requirements traceability** Each requirement is traceable to its system goal
 - Down-tracing** Each requirement is traced down to data, hardware, personnel, and software
- (f) **Cross-referencing** *Sources and further reading:* [7]: 5.3, [8]: 9.5.9, [5]: , [3]: 2.24
Whenever applicable, are the requirements cross-referenced to related requirements? Although not ideal, requirements might be described at a variety of levels which not only introduces redundancy but also makes it difficult to see which requirements are related. In order to increase understandability, cross-referencing could be applied when sections containing requirements are related to other sections containing:
 - Identical (i.e., redundant) requirements
 - More abstract or more detailed descriptions of the same requirements
 - Requirements that depend on them or on which they depend
- (g) **Categorisation and ranking** *Sources and further reading:* [7]: 5.3.7, [7]: 4.3.5, [5]: , [3]: 2.15-16
Are the requirements categorized c.q. organised in a satisfactory way in order to maximise readability?
Example ways to categorise are:
 - Type of requirement
 - Degree of stability
 - Degree of necessity (Prioritisation) such as..
 - Criticality of the requirement to stakeholders
 - Scheduling from an architectural standpoint
 - Implementation precedence
 - The minimisation of project risk
- (h) **Apportioning of Requirements** *Sources and further reading:* [8]: 9.5.8, [5]:
If so desired, are requirements apportioned and cross-referenced to software (architectural) elements?
- (i) **Allocation of Requirements** *Sources and further reading:* [5]:
Are the requirements allocated to the team or individual who will implement it?
- (j) **Degree of implementability**
Where desired, are degrees of how difficult a requirement is to implement specified?

3 Specification of Individual Requirements

3.1 Requirement impact

- (a) **Semantic correctness** *Sources and further reading:* [5]:
Is the requirement semantically correct? It is so if it
- meets some real need of its relevant stakeholder(s)
 - is an accurate elaboration of a documented business objective or goal
 - all numbers associated with the requirement have correct values
- (b) **Internal consistency** *Sources and further reading:* [8]: 5.2.5
Is the requirement internally consistent? It is so if it does not conflict with any other internal requirements or descriptions
- (c) **External consistency** *Sources and further reading:* [5]:
Is the requirement externally consistent? It is so if the requirement is consistent with documented sources such as higher-level goals and requirements
- (d) **Relevancy / currency** *Sources and further reading:* [5]: , [8]: 5.2.5
Is the requirement (still) relevant?
- Is the requirement of a current or anticipated customer or user need?
 - Has the requirement not been made obsolete by the passage of time?
 - Is it essential to the success of the product?
 - Will it be implemented?
 - Is it truly required by some stakeholder?
 - Is it within the scope of the business, application, or component being specified?
- (e) **Necessity** *Sources and further reading:* [8]: 5.2.5
Is the requirement necessary? It does so if it “defines an essential capability, characteristic, constraint, and/or quality factor. If it is removed or deleted, a deficiency will exist, which cannot be fulfilled by other capabilities of the product or process”[8]
- (f) **Feasibility/achieveability** *Sources and further reading:* [8]: 5.2.5,[5]: , [3]: 2.8
Is the requirement feasible c.q. achievable? Can the requirement be implemented given the..
- existing hardware or software technology?
 - endeavour’s budget?
 - endeavour’s schedule?
 - endeavour’s constraints on staffing (e.g., staff size, expertise, and experience)?
 - limitations of physics, chemistry, etc?

3.2 Requirement detailing

- (a) **Rationale** *Sources and further reading:* [6]:
Does the requirement have a reasonable rationale associated with it that justifies it being specified as a requirement?
- (b) **No design projection** *Sources and further reading:* [7]: 4.7, [5]: , [8]: 5.2.5
Is the requirement formulated in such a way that no implementation is projected, or at the very least, does not restrict the implementation more than deemed necessary? The requirement should:
- only specify externally observable behaviour (as a black-box)
 - avoid specifying any internal architecture, design, implementation or testing decision
 - clearly identify any design projections as constraints rather than as a pure requirement
- In short, it should answer a ‘what’, not a ‘how’
- (c) **Verifiability** *Sources and further reading:* [7]: 4.3.6, [10]: 0, [1]: 4, [3]: 2.5
Is the statement verifiable? A requirement is verifiable if, and only if, there exists some finite cost-effective process with which a person or machine can check that the software product meets the requirement. This can also be called a fit criterion
- (d) **Validatability** *Sources and further reading:* [5]:
Is the requirement validatable? Is it possible to ensure that the requirement is actually what the different types of stakeholders want and need?
- (e) **Mentioning of affecting assumptions**
If there is a factor present that could affect the requirement, is it properly specified in the document? An example would be that the requirement assumes that a specific operating system will be available at the right time)

- (f) **Assertions** *Sources and further reading:* [5]:
Does the requirement include the necessary
- Invariants
 - Preconditions
 - Post-conditions
- (g) **Isolation** *Sources and further reading:* [7]: 4.3.7.c, [5]: Completeness, [8]
Is the requirement isolated?
- Is the requirement expressed separately, as opposed to intermixed with other requirements?
 - Is the requirement actually a single requirement and not actually multiple requirements?
 - “The requirement statement includes only one requirement with no use of conjunctions”[8]
- (h) **Cohesiveness** *Sources and further reading:* [5]:
Is the requirement cohesive? Although the type of cohesion varies with the different type of requirements being specified:
- Does each requirement specify only one thing?
 - Do all parts of the requirement belong together:
 - Do all parts of a data requirement involve the same data abstraction?
 - Do all parts of a functional requirement involve the same functional abstraction?
 - Do all parts of an interface requirement involve the same interface?
 - Do all parts of a quality requirement involve the same quality factor or sub-factor?

3.3 Requirement phrasing

- (a) **Syntactical correctness** *Sources and further reading:* [5]: , [9]
Is the item syntactical correct? It is so if:
- It uses the proper standard format (if any)
 - It properly uses the words ‘shall’ or ‘must’ rather than ‘will’ or ‘may’
 - Its words are spelled correctly
 - The sentences are grammatically correct
- (b) **Unambiguous terminology** *Sources and further reading:* [7]: 4.3.2, [1]: 4, [5]: , [4]
Does the requirement contain no potentially ambiguous words? Examples are:
- Terms that could have multiple meanings
 - Vague subjects that can refer to multiple things (e.g., ‘it’ or ‘they’)
 - Vague adjectives (e.g., ‘very’, ‘strong’, ‘acceptable’, ‘soon’)
 - Vague prepositions (e.g., ‘above’, ‘high’, ‘over’)
 - Vague verbs that are more qualitative than quantitative (e.g., ‘increase’, ‘minimise’)
 - Subjective phrases (e.g., ‘If possible’, ‘when appropriate’)
- In case of terms with multiple meanings, is the correct meaning included and made more specific in the glossary/dictionary?
- (c) **Unambiguous meaning** *Sources and further reading:* [7]: 4.3.2, [1]: 4, [5]: , [4]
Is the requirement unambiguous? Will the interpretation of each requirement be the same for all types of stakeholders and potential readers of this document?
- (d) **Understandability** *Sources and further reading:* [5]: , [3]: 2.4
Is the requirement written in a way that the meaning is comprehend-able to all likely readers of this requirement with a minimum of explanation? Does it avoid the technical jargon of the development organisation? Potential readers include customers, users, project managers (PM), software developers and testers
- (e) **Preciseness** *Sources and further reading:* [1]: 4, [3]: 2.20
Is the requirement clear, concrete and precise? It is so when measurable quantities are used whenever possible and the appropriate levels of precision are used for all numeric quantities
- (f) **Level of abstraction/detail** *Sources and further reading:* [8]: 9.5.9, [1]: 4, [3]: 2.19
Does the requirement have the desired level of detail and is it specific enough? As an example, it should enable:

Designers to design a software system to satisfy those requirements

Testers to test that software system satisfies those requirements

4 Validity of the document as a whole

The following checks affect the entire document and will most likely only be answerable once you have read, and understood, the entire specification.

4.1 Wording of the contents

- (a) **Quality of writing**
Is the document well written? Is it free of spelling errors? Is the writing grammatically correct? Is the document written in non-technical language?
- (b) **Conciseness** *Sources and further reading: [3]: 2.9*
Is the SRS as short as possible without adversely affecting any other quality of the SRS?
- (c) **Coherence**
Is the terminology being used consistent?
- (d) **External document referencing**
Are all relevant external documents mentioned. Examples are Project-, System- and Stakeholder Requirements Specifications

4.2 Validity of the collective requirements

- (a) **Requirement types coverage** *Sources and further reading: [10]: 0, 9, 10-17*
Are all desired types of requirements for this system properly described? Examples being:
 - Functional Requirements** Specifies 1) what functions are to be performed on 2) what data to produce 3) what results at 4) what location for 5) whom?
 - Non-functional Requirements** Properties of the functions such as Reliability-, Availability-, Scalability-, Capacity-, Interfacing-, Legal-, Safety-Critical-, Security-, Maintainability-, Portability-, Performance-, Usability-, Database Requirements and Standards compliance
 - Other** Project constraints, Design constraints, Project drivers, Project issues
- (b) **Non-Redundancy** *Sources and further reading: [7]: 4.3.7.b, [3]: 2.18*
Are there no redundant requirements?
- (c) **Consistency of descriptions**
Are the requirements consistent in format, level of abstraction, amount of details and used language?
- (d) **Internal consistency** *Sources and further reading: [7]: 4.3.4.1, [1]: 2.a, [3]: 2.6, [8]: 5.2.6*
Is the SRS internally consistent?
An SRS is internally consistent if, and only if, no subset of individual requirements described in it conflict. Examples of likely conflicts in an SRS are:
 - The specified characteristics of real-world objects may conflict. For example,
 - The format of an output report may be described in one requirement as tabular but in another as textual
 - One requirement may state that all lights shall be green while another may state that all lights shall be blue
 - There may be a logical or temporal conflict between two specified actions. For example,
 - One requirement may specify that the program will add two inputs and another may specify that the program will multiply them.
 - One requirement may state that "A" must always follow "B," while another may require that "A and B" occur simultaneously
 - Inconsistent use of definitions, descriptions and terminology. Two or more requirements may describe the same real-world object but use different terms for that object.
- (e) **External consistency** *Sources and further reading: [8]: 9.5.2D, [1]: 2.b, [3]: 2.7*
Is the SRS externally consistent?
It is so if the Requirement is not in conflict with any already base lined project documentation. These base lined documents include system-level requirements specifications (RS), statements of work (SOW), white papers, an earlier version of the SRS to which this new SRS must be upward compatible, and Requirements Specifications for other systems to which this system must interface.
- (f) **Justification of decisions**
Is there sufficient documentation for any significant decision? Were all alternatives considered and their advantages and disadvantages properly explained? Is the reasoning objective?

(g) **Requirements analysis** *Sources and further reading:* [6]:

Have all requirements been analysed appropriately?

- Have appropriate analysis techniques been used?
- Have the analysis techniques been used to the right level of detail?
- Has a sufficient number of techniques been used?

(h) **System feasibility, validity & achievability** *Sources and further reading:* [1]: 3, [9], [3]: 2.8, [2]

Is the specified system feasible?

A specification is feasible when:

- All individual requirements are feasible
- a solution to the set of requirements exists and is achievable
- the life-cycle benefits of the system specified will exceed the life-cycle costs.
- the specified system will be sufficiently maintainable.
- it implies the identification and resolution of any high-risk issues involved

(i) **Completeness & Correctness** *Sources and further reading:* [7]: 4.3.1, 4.3.3, [1]: 1, [3]: 2.3, [8]: 5.2.6 Is every requirement stated in the SRS one that the final product will (likely) meet?

Is the Software Requirements Specification (sufficiently) complete?

An SRS is complete if it includes:

- All significant requirements
- Definition of the responses of the software to all realisable classes of input data in all realisable classes of situations. Note that it is important to specify the responses to both valid and invalid input values.
- Full labels and references to all Figures, tables, and diagrams in the SRS and definition of all terms and units of measure.

Is each characteristic of the final product described?

Furthermore, there should be no:

- reference to nonexistent functions, inputs or outputs
- missing specification items
- missing functions
- missing products

5 Context surrounding the document under review

These checks affect aspects outside the scope of written documentation, but which are still regarded as important for the validation of the SRS.

- (a) **Joint preparation** *Sources and further reading:* [7]: 4.4, [6]: 3
Is the SRS jointly prepared between customer and supplier?
- Were the requirements elicited from all critically-important stakeholders?
 - Were all types of stakeholders used as sources of the requirements?
 - Was the sampling of stakeholders large enough to be statistically valid?
- (b) **Usage of standards** *Sources and further reading:* [6]: 3
Have relevant laws, regulations, and standards been used as sources of requirements?
- (c) **Stakeholder agreement** *Sources and further reading:* [8]: 5.2.1, 6.2.3.1-6.2.3.3, 8.2.2, [10]: 4a, 6b
Do relevant stakeholders (including supplier and customer) agree on the SRS, including requirements, context of work and definitions?

6 Absence of unwanted elements

- (a) **Project requirements** *Sources and further reading:* [7]: 4.8
Are there no unwanted descriptions regarding the process of producing the software product? Examples of common items that should be avoided are:
- Costs/budgets
 - Implied project management constraints
 - Delivery schedules
 - Reporting procedures
 - Software development methods
 - Quality assurance
 - Validation and verification criteria
 - Acceptance procedures
- Project requirements should normally be specified in other documents, typically in a software development plan, a software quality assurance plan, or a statement of work
- (b) **Design/implementation independent** *Sources and further reading:* [7]: 4.2.b, 4.7, [3]: 2.10
Is the SRS free of design c.q. implementation details or otherwise free of statements that unnecessarily limit the range of valid designs?
A properly written SRS limits the range of valid designs, but does not specify any particular design. The SRS writer(s) should clearly distinguish between identifying required design constraints and projecting a specific design.
Note that every requirement in the SRS limits design alternatives. This does not mean, though, that every requirement is design. The SRS should normally not specify design items such as the following:
- Details of the system design
 - Details of system architecture
 - Partitioning the software into modules
 - Allocating functions to the modules
 - Describing the flow of information or control between modules
 - Describing the flow of information or control between modules
 - Choosing data structures
- In special cases however, some requirements may severely restrict the design. For example, security or safety requirements may reflect directly into design.
- (c) **Ambiguities** *Sources and further reading:* [3]: 2.1, [4]
Are there no ambiguities in the document? Examples are:
- Unclear or multi-interpretable sentences
 - Software jargon
- (d) **'To Be Determined'** *Sources and further reading:* [7]: 4.3.3.1
Is there no unwanted mentioning of 'To Be Determined' in the document? Inclusion of TBD's should be avoided whenever possible. If deemed necessary however, it could be accompanied with:
- A description of the conditions causing the TBD

- A description of what must be done to eliminate the TBD
- Who is responsible for its elimination
- By when it must be eliminated.

References

- [1] B. W. Boehm, "Guidelines for Verifying and Validating Software Requirements and Design Specifications," *Euro IFIP* 79, no. 1, pp. 711–719, 1979.
- [2] CxOne, "CxOne Requirements Checklist," CxOne, Tech. Rep., 2003. [Online]. Available: <https://www.construx.com/resources/requirements-checklist/>
- [3] A. Davis, S. Overmyer, K. Jordan, J. Caruso, F. Dandashi, A. Dinh, G. Kincaid, G. Ledeboer, P. Reynolds, P. Sitaram, and others, "Identifying and measuring quality in a software requirements specification," in *Proceedings First International Software Metrics Symposium*. IEEE, 1993, pp. 141–152.
- [4] A. M. Davis and D. Zowghi, "Good requirements practices are neither necessary nor sufficient," *Requirements Engineering*, vol. 11, no. 1, pp. 1–3, 2006.
- [5] D. Firesmith, "Specifying good requirements," *Journal of Object Technology*, vol. 2, no. 4, pp. 77–87, 2003.
- [6] —, "Quality requirements checklist," *Journal of Object Technology*, vol. 4, no. 9, pp. 31–38, 2005.
- [7] IEEE, "IEEE 830: Recommended Practice for Software Requirements Specifications," IEEE, Tech. Rep., 1998.
- [8] ISO/IEC 2011 and IEEE 2011, "ISO/IEC/IEEE 29148: Systems and software engineering — Life cycle processes — Requirements engineering," ISO/IEC/IEEE, Tech. Rep., 2011.
- [9] O. I. Lindland, G. Sindre, and A. Sølvberg, "Understanding Quality in Conceptual Modeling," *IEEE Software*, 1994.
- [10] J. Robertson and S. Robertson, "Volere," *Requirements Specification Templates*, 2000.

Appendix B

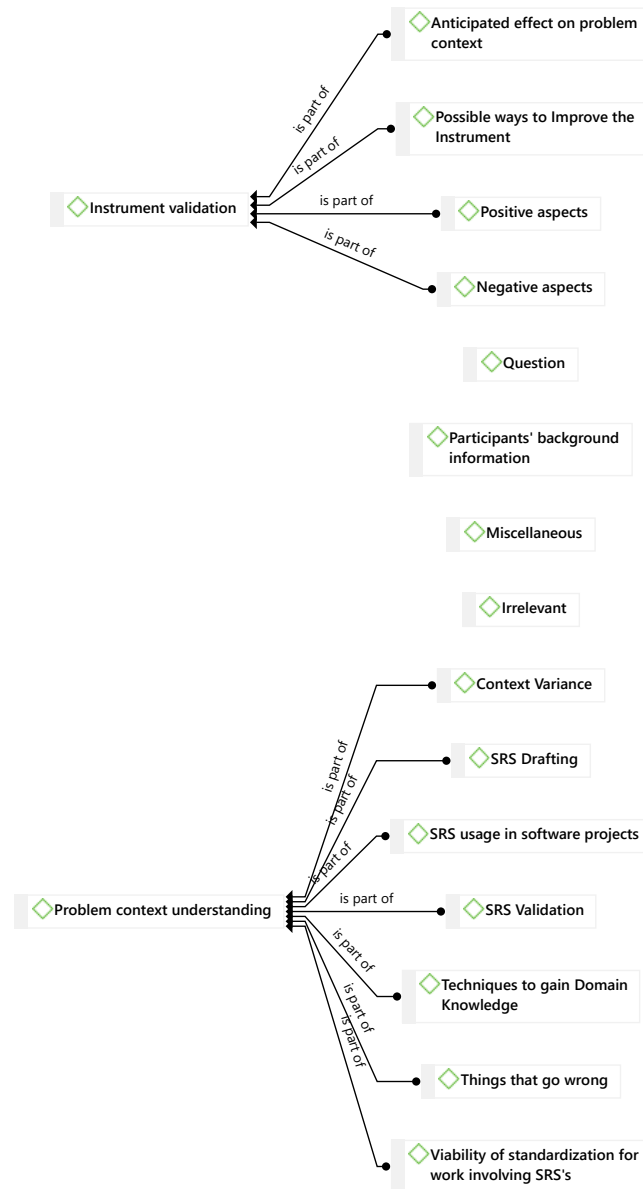
Interview sessions - analysis

B.1 Interview Scheme

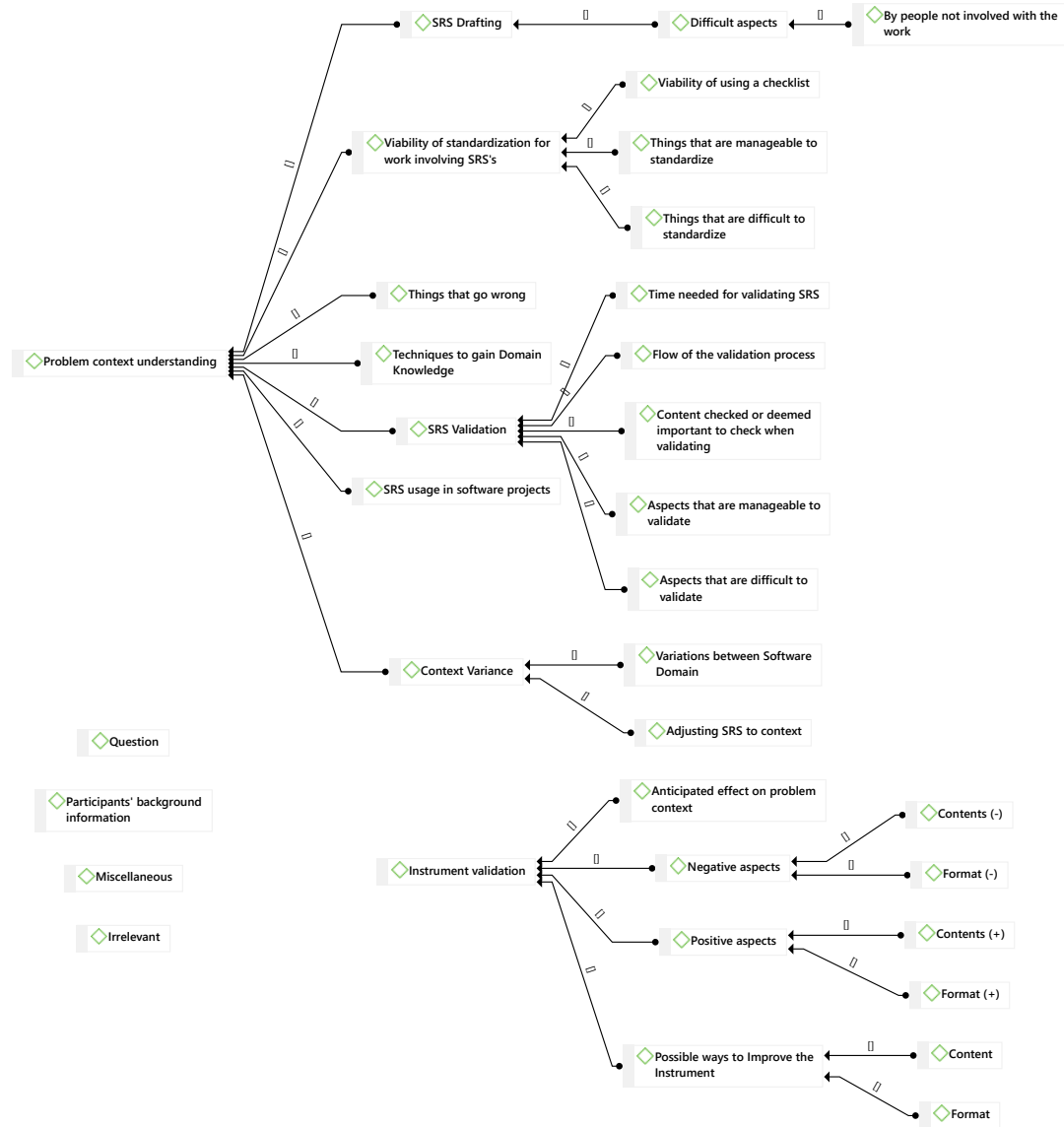
1. Ascertaining background information of the participant (5m)
 - General back ground
 - Participant's background with working in Software Projects
 - Participant's experience specific to SRS's
2. Ascertaining insights into the problem context (15m)
 - Participant's experiences within the problem contex
 - Participant's experiences with their work as outside experts c.q. consultants
 - Participant's experiences with how SRS's are usually created, used and evaluated of SRS's
 - Participant's personal experience with evaluating SRS's
3. Ascertaining insights into SRS evaluation (20m)
Showing the sample SRS and assessing:
 - how the participant would evaluate that SRS
 - the participant's impression of what could and could not be standardized in this process
4. Ascertaining participants' opinion on the designed instrument (20m)
Showing the designed instrument and assessing:
 - the Instrument's structure
 - the Instrument's contents
 - the Instrument's anticipated effect on problem context

B.2 Code Skeletons

B.2.1 High-level

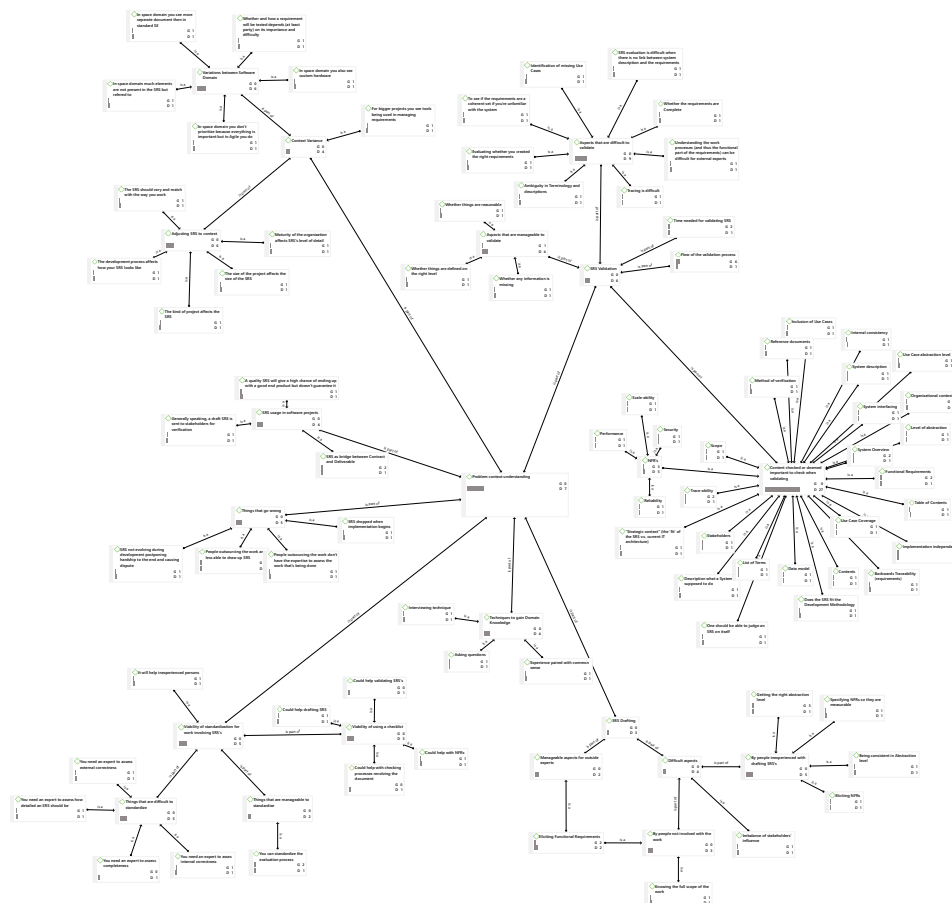


B.2.2 Complete



B.3 Code Tree

B.3.1 Problem Investigation



B.4 Code Table

Check Name	P1	P2	Total
"Strategic context" (the 'fit' of the SRS v.s. current IT architecture)	1	0	1
A checklist would help with checking aspects surrounding the SRS	1	0	1
A checklist would prevent you from missing things to check for	0	1	1
A checklist would speed up the SRS evaluation process	0	1	1
A quality SRS will give a high chance of ending up with a good end product but doesn't guarantee it	1	0	1
Add check for licensing information for external software	0	1	1
Ambiguity	0	1	1
Ambiguity in Terminology and descriptions	0	1	1
Appendices	1	0	1
Asking questions	1	0	1
Aspects that are manageable to validate	0	1	1
Backwards and forwards traceability	0	1	1
Backwards Traceability (requirements)	1	0	1
Being consistent in Abstraction level	1	0	1
Black Box	1	0	1
Business case has no part in SRS	1	0	1
Check if you agree with the method of verification	0	1	1
Check to see whether a requirement is verifiable, how it will be verified and if you agree with that	0	2	2
Checklist could be in matrix-form with checkboxes to tick	0	2	2
Checklist should be about 3-4 pages long	2	2	4
Checklist should be usable	0	1	1
Condense the checklist into a shorter format you can practically use	1	0	1
Consistency	0	1	1
Contents	0	1	1
Could help drafting SRS	1	0	1
Could help with NFRs	1	0	1
Cross-referencing	0	1	1
Data model	0	1	1
Dependencies	0	1	1
Description what a System is supposed to do	1	0	1
Does the SRS fit the Development Methodology	1	0	1
Eliciting Functional Requirements	2	0	2
Eliciting NFRs	1	0	1
Evaluating whether you created the right requirements	0	1	1
Example checks for OTS products	1	0	1
Experience paired with common sense	1	0	1
External interfaces	2	1	3
Find the right balance between the important checks and the size of the checklist	1	0	1
Flow of the validation process	2	4	6
For bigger projects you see tools being used in managing requirements	0	1	1
For the checklist itself, the headings would be enough	1	0	1
Functional Requirements	1	1	2
Generalize a bit more, less specifics	1	0	1
Generally speaking, a draft SRS is sent to stakeholders for verification	1	0	1

Table B.1 continued from previous page

Check Name	P1	P2	Totals
Getting the right abstraction level	3	0	3
Glossary	1	2	3
Group formatting checks (e.g. number consistency)	1	0	1
Identical Requirements	0	1	1
Identifiability	0	1	1
Identification of missing Use Cases	1	0	1
Imbalance of stakeholders' influence	1	0	1
Implementation independence	2	0	2
In space domain much elements are not present in the SRS but referred to	0	1	1
In space domain you also see custom hardware	0	1	1
In space domain you don't prioritize because everything is important but in Agile you do	0	1	1
In space domain you see more separate document then in standard SE	0	1	1
Inclusion of Use Cases	1	0	1
Index	1	1	2
Interfaces	1	1	2
Internal consistency	1	0	1
Internal/External Correctness	0	1	1
Interviewing technique	1	0	1
Introduction	0	1	1
Irrelevant	5	5	10
It is good to leave certain aspects up to the evaluator to decide whether things need to be in the SRS	1	0	1
It will help inexperienced persons	0	1	1
Knowing the full scope of the work	1	0	1
Let the evaluator decide whether a check is applicable	0	1	1
Level of abstraction	1	0	1
Level of Abstraction (2)	0	1	1
List of Terms	0	1	1
Lists of references	1	1	2
Maturity of the organization affects SRS's level of detail	1	0	1
Method of verification	0	1	1
Minimum set	0	1	1
Miscellaneous	4	1	5
NFR list too extensive	1	0	1
NFR's	2	1	3
Non-inclusion of Planning information	0	1	1
Off-The-Shelf Software	1	1	2
One should be able to judge an SRS on itself	0	1	1
Organizational context	1	0	1
Overall description	1	1	2
Overview of possible NFRs (type completeness)	1	1	2
Participants' background information	5	8	13
People don't generally check for correctness	1	0	1
People outsourcing the work are less able to draw up SRS	2	0	2
People outsourcing the work don't have the expertise to assess the work that's being done	1	0	1
Performance	1	0	1

Table B.1 continued from previous page

Check Name	P1	P2	Totals
Performance (2)	0	1	1
Prioritization	0	1	1
Purpose	0	1	1
Question	40	43	83
Redundancy	0	1	1
Reference documents	0	1	1
References to standards should be practical	0	1	1
Reliability	1	0	1
Scale-ability	1	0	1
Scope	0	1	1
Security	1	0	1
See if you can see how far you can help people to check for completeness	0	1	1
Some elements will be optional depending on the project	2	0	2
Specifying NFRs so they are measurable	1	0	1
SRS as bridge between Contract and Deliverable	2	0	2
SRS dropped when implementation begins	1	0	1
SRS evaluation is difficult when there is no link between system description and the requirements	0	1	1
SRS not evolving during development postponing hardship to the end and causing dispute	1	0	1
Stakeholders	1	0	1
Stakeholders (2)	0	1	1
Structure and contents of Requirements in general	0	1	1
System description	0	1	1
System interfacing	1	0	1
System Overview	1	1	2
Table of Contents	1	0	1
Table of Contents (2)	0	1	1
The additional information can be very valuable	0	2	2
The development process affects how your SRS looks like	1	0	1
The division of chapters is a good one	2	1	3
The document makes sense	1	0	1
The flow of the checklist item is a good one	0	1	1
The kind of project affects the SRS	0	1	1
The size of the project affects the size of the SRS	1	0	1
The SRS should vary and match with the way you work	0	1	1
There could be a difference between process steps and product steps	1	0	1
Third party	0	1	1
Time needed for validating SRS	1	1	2
To see if the requirements are a coherent set if you're unfamiliar with the system	0	1	1
Traceability	0	1	1
Trace-ability	0	2	2
Tracing is difficult	0	1	1
Type Completeness could be useful	0	1	1
Type completeness part is unclear	0	1	1
Type Completeness too extensive	0	1	1

Table B.1 continued from previous page

Check Name	P1	P2	Totals
Understanding the work processes (and thus the functional part of the requirements) can be difficult for external experts	1	0	1
Use Case abstraction level	1	0	1
Use Case Coverage	1	0	1
User Analysis	0	1	1
Users	0	1	1
Verifiability	0	1	1
Version number	0	1	1
Whether and how a requirement will be tested depends (at least partly) on its importance and difficulty	0	1	1
Whether any information is missing	1	0	1
Whether the requirements are Complete	0	1	1
Whether things are defined on the right level	1	0	1
Whether things are reasonable	1	0	1
You can split it up into two parts, 1 part fundamentals, 1 part questionables	1	0	1
You can standardize the evaluation process	0	2	2
You could split up the document into a handbook and a checklist	0	2	2
You need an expert to assess internal correctness	1	0	1
You need an expert to assess external correctness	1	0	1
You need an expert to assess how detailed an SRS should be	1	0	1
Total	146	150	296

Appendix C

Live Study Materials

C.1 Checklist sheet versions

C.1.1 Version 1

Version 1		For a real life scenario, please mark if you think this check is..					
		Inapplicable	Ambiguous	Redundant	Unimportant	Contains incorrect information	Other...
1 Content and outline of the report							
1.1 Report Structure and references							
a	Document details						
b	Table-of-contents						
c	Index						
d	Glossary						
e	Appendices						
f	Numbering-and-referencing						
g	List-of-references						
h	Overview						
i	Document structure						
1.2 Understanding the Product Environment							
a	Product purpose						
b	SRS-purpose						
c	Goals of the project						
d	User-business						
e	Scope of the work						
f	Business-environment						
1.3 Understanding the product to be built							
a	Product perspective						
b	Product functions and -functionality						
c	Performance-indication-of-functions						
d	Limitations/Constraints identification						
e	Assumptions-and-dependencies						
f	Risks						
1.4 Environment							
a	Implementation environment						
b	External-interfaces						
c	Partner or collaborative applications						
1.5 Understanding the stakeholders							
a	Stakeholder coverage						
b	Stakeholder-information						
c	User-characteristics						
2 Structure and content of the requirements in general							
a	Organization						
b	Structuring						
c	Numbering consistency						
d	Modifiability						
e	Traceability						
f	Cross-referencing						
g	Categorization-and-ranking						
h	Apportioning of the Requirements						
i	Allocation-of-Requirements						
j	Degree-of-implementability						

Version 1		For a real life scenario, please mark if you think this check is..					
		Inapplicable	Ambiguous	Redundant	Unimportant	Contains incorrect information	Other...
3 For each Requirement							
3.1 Correctness							
a	Semantical correctness						
b	Internal-consistency						
c	External consistency						
d	Relevancy						
e	Reflectiveness / trueness						
f	Currency						
g	Feasibility						
3.2 Specified information							
a	Rationale						
b	No design projection						
c	Verifiability						
d	Validatability						
e	Assertions						
f	Mentioning-of-affecting-assumptions						
g	Isolation						
h	Cohesiveness						
3.3 Wording							
a	Syntactical correctness						
b	Unambiguous-terminology						
c	Unambiguous-meaning						
d	Understandability						
e	Preciseness						
f	Specificness						
4 Affecting the span of all requirements							
a	Requirement types coverage						
b	Non-Redundancy						
c	Consistency of descriptions						
d	Internal-consistency						
e	External-consistency						
f	Justification-of-decisions						
g	Requirements analysis						
h	System-feasibility						
i	SRS correctness						
j	Completeness						
5 External to the document under review							
3	a Joint preparation						
4	b Usage of standards						
0	c Stakeholder agreement						
6 What should not be in there							
a	Project requirements						
b	Design or implementation details						
c	Ambiguities						
d	To-be-Determined						

C.1.2 Version 2

Version 2		For a real life scenario, please mark if you think this check is..					
		Inapplicable	Ambiguous	Redundant	Unimportant	Contains incorrect information	Other...
1 Content and outline of the report							
1.1 Report Structure and references							
a	Document details						
b	Table of contents						
c	Index						
d	Glossary						
e	Appendices						
f	Numbering and referencing						
g	List of references						
h	Overview						
i	Document structure						
1.2 Understanding the Product Environment							
a	Product purpose						
b	SRS purpose						
c	Goals of the project						
d	User business						
e	Scope of the work						
f	Business environment						
1.3 Understanding the product to be built							
a	Product perspective						
b	Product functions and -functionality						
c	Performance indication of functions						
d	Limitations/Constraints identification						
e	Assumptions and dependencies						
f	Risks						
1.4 Environment							
a	Implementation environment						
b	External interfaces						
c	Partner or collaborative applications						
1.5 Understanding the stakeholders							
a	Stakeholder coverage						
b	Stakeholder information						
c	User characteristics						
2 Structure and content of the requirements in general							
a	Organization						
b	Structuring						
c	Numbering consistency						
d	Modifiability						
e	Traceability						
f	Cross-referencing						
g	Categorization and ranking						
h	Apportioning of the Requirements						
i	Allocation of Requirements						
j	Degree of Implementability						

Version 2		For a real life scenario, please mark if you think this check is..					
		Inapplicable	Ambiguous	Redundant	Unimportant	Contains incorrect information	Other...
3 For each Requirement							
3.1 Correctness							
a	Semantical correctness						
b	Internal consistency						
c	External consistency						
d	Relevancy						
e	Reflectiveness / trueness						
f	Currency						
g	Feasibility						
3.2 Specified information							
a	Rationale						
b	No design projection						
c	Verifiability						
d	Validatability						
e	Assertions						
f	Mentioning of affecting assumptions						
g	Isolation						
h	Cohesiveness						
3.3 Wording							
a	Syntactical correctness						
b	Unambiguous terminology						
c	Unambiguous meaning						
d	Understandability						
e	Preciseness						
f	Specificness						
4 Affecting the span of all requirements							
a	Requirement types coverage						
b	Non-Redundancy						
c	Consistency of descriptions						
d	Internal consistency						
e	External consistency						
f	Justification of decisions						
g	Requirements analysis						
h	System feasibility						
i	SRS correctness						
j	Completeness						
5 External to the document under review							
3 a	Joint preparation						
4 b	Usage of standards						
0 c	Stakeholder agreement						
6 What should not be in there							
a	Project requirements						
b	Design or implementation details						
c	Ambiguities						
d	To-be-Determined						

C.1.3 Version 3

Version 3		For a real life scenario, please mark if you think this check is..					
		Inapplicable	Ambiguous	Redundant	Unimportant	Contains incorrect information	Other...
1 Content and outline of the report							
1.1 Report Structure and references							
a	Document details						
b	Table of contents						
c	Index						
d	Glossary						
e	Appendices						
f	Numbering and referencing						
g	List of references						
h	Overview						
i	Document structure						
1.2 Understanding the Product Environment							
a	Product purpose						
b	SRS purpose						
c	Goals of the project						
d	User business						
e	Scope of the work						
f	Business environment						
1.3 Understanding the product to be built							
a	Product perspective						
b	Product functions and -functionality						
c	Performance indication of functions						
d	Limitations/Constraints identification						
e	Assumptions and dependencies						
f	Risks						
1.4 Environment							
a	Implementation environment						
b	External interfaces						
c	Partner or collaborative applications						
1.5 Understanding the stakeholders							
a	Stakeholder coverage						
b	Stakeholder information						
c	User characteristics						
2 Structure and content of the requirements in general							
a	Organization						
b	Structuring						
c	Numbering consistency						
d	Modifiability						
e	Traceability						
f	Cross-referencing						
g	Categorization and ranking						
h	Apportioning of the Requirements						
i	Allocation of Requirements						
j	Degree of implementability						

Version 3		In a real life scenario, please mark if you think this check is..					
		Inapplicable	Ambiguous	Redundant	Unimportant	Contains incorrect information	Other...
3 For each Requirement							
3.1 Correctness							
a	Semantical correctness						
b	Internal consistency						
c	External consistency						
d	Relevancy						
e	Reflectiveness / trueness						
f	Currency						
g	Feasibility						
3.2 Specified information							
a	Rationale						
b	No design projection						
c	Verifiability						
d	Validatability						
e	Assertions						
f	Mentioning of affecting assumptions						
g	Isolation						
h	Cohesiveness						
3.3 Wording							
a	Syntactical correctness						
b	Unambiguous terminology						
c	Unambiguous meaning						
d	Understandability						
e	Preciseness						
f	Specificness						
4 Affecting the span of all requirements							
a	Requirement types coverage						
b	Non-Redundancy						
c	Consistency of descriptions						
d	Internal consistency						
e	External consistency						
f	Justification of decisions						
g	Requirements analysis						
h	System feasibility						
i	SRS correctness						
j	Completeness						
5 External to the document under review							
3 a	Joint preparation						
4 b	Usage of standards						
0 c	Stakeholder agreement						
6 What should not be in there							
a	Project requirements						
b	Design or implementation details						
c	Ambiguities						
d	To be Determined'						

C.1.4 Version 4

Version 4		For a real life scenario, please mark if you think this check is..					
		Inapplicable	Ambiguous	Redundant	Unimportant	Contains incorrect information	Other...
1 Content and outline of the report							
1.1 Report Structure and references							
a	Document details						
b	Table of contents						
c	Index						
d	Glossary						
e	Appendices						
f	Numbering and referencing						
g	List of references						
h	Overview						
i	Document structure						
1.2 Understanding the Product Environment							
a	Product purpose						
b	SRS purpose						
e	Goals of the project						
d	User business						
e	Scope of the work						
f	Business environment						
1.3 Understanding the product to be built							
a	Product perspective						
b	Product functions and -functionality						
c	Performance indication of functions						
d	Limitations/Constraints identification						
e	Assumptions and dependencies						
f	Risks						
1.4 Environment							
a	Implementation environment						
b	External interfaces						
c	Partner or collaborative applications						
1.5 Understanding the stakeholders							
a	Stakeholder coverage						
b	Stakeholder information						
e	User characteristics						
2 Structure and content of the requirements in general							
a	Organization						
b	Structuring						
c	Numbering consistency						
d	Modifiability						
e	Traceability						
f	Cross-referencing						
g	Categorization and ranking						
h	Apportioning of the Requirements						
i	Allocation of Requirements						
j	Degree of implementability						

Version 4		For a real life scenario, please mark if you think this check is..					
		Inapplicable	Ambiguous	Redundant	Unimportant	Contains incorrect information	Other...
3 For each Requirement							
3.1 Correctness							
a	Semantical correctness						
b	Internal consistency						
c	External consistency						
d	Relevancy						
e	Reflectiveness / trueness						
f	Currency						
g	Feasibility						
3.2 Specified information							
a	Rationale						
b	No design projection						
c	Verifiability						
d	Validatability						
e	Assertions						
f	Mentioning of affecting assumptions						
g	Isolation						
h	Cohesiveness						
3.3 Wording							
a	Syntactical correctness						
b	Unambiguous terminology						
c	Unambiguous meaning						
d	Understandability						
e	Preciseness						
f	Specificness						
4 Affecting the span of all requirements							
a	Requirement types coverage						
b	Non-Redundancy						
c	Consistency of descriptions						
d	Internal consistency						
e	External consistency						
f	Justification of decisions						
g	Requirements analysis						
h	System feasibility						
i	SRS correctness						
j	Completeness						
5 External to the document under review							
3	a Joint preparation						
4	b Usage of standards						
0	c Stakeholder agreement						
6 What should not be in there							
a	Project requirements						
b	Design or implementation details						
c	Ambiguities						
d	To be Determined:						

C.2 Post-Use Questionnaire

]

Post-Use Survey of Software Requirements Specification Checklist + Handbook -REFSQ2018

In years, please indicate your experience with..

1. Requirements Engineering as a researcher.. ☐ 0 ☐ 1-2 ☐ 3-5 ☐ 5-10 ☐ 10+
2. Requirements Engineering as a practitioner ☐ 0 ☐ 1-2 ☐ 3-5 ☐ 5-10 ☐ 10+
3. Writing System/Software Specifications..... ☐ 0 ☐ 1-2 ☐ 3-5 ☐ 5-10 ☐ 10+
4. Validating System/Software Specifications. ☐ 0 ☐ 1-2 ☐ 3-5 ☐ 5-10 ☐ 10+

Please rate whether you think the following aspects apply to the checklist + handbook:

1. Applicability to the full range of intended use Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
2. Clarity Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
3. Comprehensiveness¹ Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
4. Concreteness Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
5. Ease of use Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
6. Fairness Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
7. Parsimony² Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
8. Item pertinence to the content area³ Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree

For a real life scenario, please indicate whether you expect the checklist + handbook to..

1. ..help prevent task saturation Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
2. ..fit the work flow Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
3. ..be finishable in a sufficient period of time Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
4. ..contains sufficient break points Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
5. ..contribute to the process of validating an SRS Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree
6. ..contribute to the quality of the validation outcome Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree

In the following text fields, please identify possible Strengths, Weaknesses, Missing items and provide recommendations for improving the checklist.

1. Strengths

2. Weaknesses

3. Missing items

4. Recommendations for improving the checklist

¹Comprehensive: covering completely or broadly

²Principle of parsimony: entities should not be multiplied needlessly; the simplest of two competing theories is to be preferred

³Pertinence: having a clear decisive relevance to the matter in hand

Appendix D

Live Study results

D.1 Checklist sheet fault detection

			Inapplicable	Ambiguous	Redundant	Unimportant	Contains incorrect information	Total
1.1	a	Document details	1	3				4
	b	Table of contents			1	1		2
	c	Index	2			1		3
	d	Glossary						0
	e	Appendices	2			1		3
	f	Numbering and referencing		2	1	2		5
	g	List of references	1			1		2
	h	Overview		4	4	2		10
	i	Document structure	1	3	3	3		10
1.2	a	Product purpose	1	2	2	2		7
	b	SRS purpose						0
	c	Goals of the project	2	4	1			7
	d	User business	1	1	2			4
	e	Scope of the work	1	4	1	2	1	9
	f	Business environment		3		1		4
1.3	a	Product perspective	1	3	4			8
	b	Product functions and -functionality	1	2	3		1	7
	c	Performance indication of functions						0
	d	Limitations/Constraints identification		1			1	2
	e	Assumptions and dependencies		1				1

Table D.1 continued from previous page

			Inapplicable	Ambiguous	Redundant	Unimportant	Contains incorrect information	Total
	f	Risks						0
1.4	a	Implementation environment	3	2	1	2		8
	b	External interfaces			1			1
	c	Partner or collaborative applications	1	2	1			4
1.5	a	Stakeholder coverage		5	1	2		8
	b	Stakeholder information			2			2
	c	User characteristics	2					2
2	a	Organization						0
	b	Structuring	1	7	2			10
	c	Numbering consistency	1	2		1	1	5
	d	Modifiability		1	1			2
	e	Traceability		3		3	3	9
	f	Cross-referencing	1		1		1	3
	g	Categorization and ranking						0
	h	Apportioning of the Requirements	2	3	1			6
	i	Allocation of Requirements	3		1			4
	j	Degree of implementability	1	2				3
3.1	a	Semantical correctness	2	5	1		1	9
	b	Internal consistency						0
	c	External consistency	1	1	1	1	1	5
	d	Relevancy		1	1			2
	e	Reflectiveness / trueness		7	7	2	1	17
	f	Currency	2	1				3
	g	Feasibility						0
3.2	a	Rationale		1	2	1		4
	b	No design projection		1		1	1	3

Table D.1 continued from previous page

			Inapplicable	Ambiguous	Redundant	Unimportant	Contains incorrect information	Total
	c	Verifiability	1	3		1	1	6
	d	Validatability			1			1
	e	Assertions	1	1		1		3
	f	Mentioning of affecting assumptions						0
	g	Isolation		1			1	2
	h	Cohesiveness		1				1
3.3	a	Syntactical correctness		1		4		5
	b	Unambiguous terminology		1		2		3
	c	Unambiguous meaning						0
	d	Understandability	1	1				2
	e	Preciseness		1	1		1	3
	f	Specificness		1	1			2
4	a	Requirement types coverage			2	1	1	4
	b	Non-Redundancy						0
	c	Consistency of descriptions			1			1
	d	Internal consistency	1					1
	e	External consistency	1	1				2
	f	Justification of decisions						0
	g	Requirements analysis		1		1	2	4
	h	System feasibility	1					1
	i	SRS correctness		3	5		3	11
	j	Completeness	3	1	2		3	9
5	a	Joint preparation	2	1		1		4
	b	Usage of standards						0
	c	Stakeholder agreement	6				1	7
6	a	Project requirements		1		1		2
	b	Design or implementation details		1	1	1	1	4
	c	Ambiguities			1			1
	d	To be Determined'	2		1			3
			53	97	62	42	26	280

Bibliography

- [1] Ulrike Abelein and Barbara Paech. 'State of practice of user-developer communication in large-scale IT projects results of an expert interview series'. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2014. ISBN: 9783319058429. DOI: [10.1007/978-3-319-05843-6_{_}8](https://doi.org/10.1007/978-3-319-05843-6_{_}8).
- [2] Mahtab Alam. 'Software Security Requirements Checklist'. In: *International Journal of Software Engineering* 3 (2010), pp. 53–62. URL: http://www.ijse.org.eg/Content/Vol3/No1/Vol3_No1_4.pdf?origin=publication_detail.
- [3] Özlem Albayrak and Jeffrey C. Carver. 'Investigation of individual factors impacting the effectiveness of requirements inspections: A replicated experiment'. In: *Empirical Software Engineering* (2014). ISSN: 13823256. DOI: [10.1007/s10664-012-9221-0](https://doi.org/10.1007/s10664-012-9221-0).
- [4] Amira A. Alshazly, Ahmed M. Elfatratry and Mohamed S. Abougabal. 'Detecting defects in software requirements specification'. In: *Alexandria Engineering Journal* (2014). ISSN: 11100168. DOI: [10.1016/j.aej.2014.06.001](https://doi.org/10.1016/j.aej.2014.06.001).
- [5] Bente Anda and Dag I. K. Sjøberg. 'Towards an inspection technique for use case models'. In: 2004. DOI: [10.1145/568760.568785](https://doi.org/10.1145/568760.568785).
- [6] Daniel M. Berry et al. 'A New Quality Model for Natural Language Requirements Specifications'. In: *12th Working Conference on Requirements Engineering: Foundation for Software Quality, REFSQ 2006* (2006).
- [7] B. Bichelmeyer. *Checklist for formatting checklists*. 2003. URL: <https://wmich.edu/sites/default/files/attachments/u350/2014/formattingchecklist.pdf>.
- [8] Elizabeth Bjarnason, Krzysztof Wnuk and Björn Regnell. 'Requirements are slipping through the gaps - A case study on causes & effects of communication gaps in large-scale software development'. In: *Proceedings of the 2011 IEEE 19th International Requirements Engineering Conference, RE 2011*. 2011. ISBN: 9781457709234. DOI: [10.1109/RE.2011.6051639](https://doi.org/10.1109/RE.2011.6051639).
- [9] Barry W. Boehm. *Guidelines for verifying and validating software requirements and design specifications*. Tech. rep. Rondo Beach, CA, USA: TRW, 1979. URL: <http://csse.usc.edu/TECHRPTS/1979/usccse79-501/usccse79-501.pdf>.
- [10] Barry W. Boehm. 'Verifying and Validating Software Requirements and Design Specifications'. In: *IEEE Software* (1984). ISSN: 07407459. DOI: [10.1109/MS.1984.233702](https://doi.org/10.1109/MS.1984.233702).
- [11] Annegret Borchard et al. 'A systematic review of the effectiveness, compliance, and critical factors for implementation of safety checklists in surgery'. In: *Annals of Surgery* (2012). ISSN: 00034932. DOI: [10.1097/SLA.0b013e3182682f27](https://doi.org/10.1097/SLA.0b013e3182682f27).
- [12] Bill Brykczynski. 'A survey of software inspection checklists'. In: *ACM SIGSOFT Software Engineering Notes* (1999), p. 82. ISSN: 01635948. DOI: [10.1145/308769.308798](https://doi.org/10.1145/308769.308798).

- [13] Ken Catchpole and Stephanie Russ. 'The problem with checklists'. In: *BMJ Quality & Safety* 24.9 (2015), pp. 545–549. ISSN: 2044-5415. DOI: [10.1136/bmjqs-2015-004431](https://doi.org/10.1136/bmjqs-2015-004431). URL: <https://qualitysafety.bmj.com/content/24/9/545>.
- [14] *Checklist: Qualities of Good Requirements*. URL: http://www.utm.mx/~caff/doc/OpenUPWeb/openup/guidances/checklists/good_requirements_594ACCB.html.
- [15] Nelly Condori-Fernandez et al. 'A Systematic Mapping Study on Empirical Evaluation of Software Requirements Specifications Techniques'. In: *ESEM: 2009 3RD INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT*. International Symposium on Empirical Software Engineering and Measurement. Univ SE Calif, Ctr Syst & Software Engrn; ABB; Microsoft Res; IEEE; ACM SIGSOFT; N Carolina State Univ Comp Sci. 345 E 47TH ST, NEW YORK, NY 10017 USA: IEEE, 2009, pp. 503+. ISBN: 978-1-4244-4842-5. DOI: [10.1109/ESEM.2009.5314232](https://doi.org/10.1109/ESEM.2009.5314232).
- [16] Jane Coughlan and Robert D. Macredie. 'Effective communication in requirements elicitation: A comparison of methodologies'. In: *Requirements Engineering* (2002). ISSN: 1432010X. DOI: [10.1007/s007660200004](https://doi.org/10.1007/s007660200004).
- [17] *CxOne Requirements Checklist*. Tech. rep. CxOne, 2003. URL: <https://www.construx.com/resources/requirements-checklist/>.
- [18] John Dalbey. *Software Requirements Specification Document Quality Assurance Criteria*. URL: https://users.csc.calpoly.edu/~jdalbey/308/Deliver/SRS_QA.html.
- [19] Alan Davis et al. 'Identifying and measuring quality in a software requirements specification'. In: *[1993] Proceedings First International Software Metrics Symposium*. IEEE. 1993, pp. 141–152.
- [20] Alan M. Davis and Didar Zowghi. 'Good requirements practices are neither necessary nor sufficient'. In: *Requirements Engineering* 11.1 (2006), pp. 1–3. ISSN: 09473602. DOI: [10.1007/s00766-004-0206-4](https://doi.org/10.1007/s00766-004-0206-4).
- [21] Martin De Laat and Maya Daneva. 'Empirical validation of a software requirements specification checklist'. In: *CEUR Workshop Proceedings*. 2018.
- [22] Martin De Laat and Maya Daneva. 'Empirical validation of a software requirements specification checklist'. In: *CEUR Workshop Proceedings*. 2018.
- [23] Osvaldo Einstein. *Checklist for Evaluation Recommendations*. 2019. URL: <https://wmich.edu/sites/default/files/attachments/u350/2019/ChecklistforEvaluationRecommendationsFieldTestVersion.pdf>.
- [24] Systems Engineering, Standards Committee and Ieee Computer. 'IEEE Standard for Software Reviews and Audits'. In: *IEEE STD 1028-2008* (2008). DOI: [10.1109/IEEESTD.2008.4601584](https://doi.org/10.1109/IEEESTD.2008.4601584).
- [25] Systems Engineering et al. 'IEEE Standard for System and Software Verification and Validation'. In: *IEEE Std 1012-2012 (Revision of IEEE Std 1012-2004)* (2012). DOI: [10.1109/IEEESTD.2012.6204026](https://doi.org/10.1109/IEEESTD.2012.6204026).
- [26] *Example Requirements Review Checklist*. URL: <http://www.uccs.edu/Documents/tboults/RequirementsReviewChecklist.doc>.

- [27] Michael Fagan. 'Design and Code Inspections to Reduce Errors in Program Development BT - Software Pioneers: Contributions to Software Engineering'. In: ed. by Manfred Broy and Ernst Denert. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 575–607. ISBN: 978-3-642-59412-0. DOI: [10.1007/978-3-642-59412-0_35](https://doi.org/10.1007/978-3-642-59412-0_35). URL: https://doi.org/10.1007/978-3-642-59412-0_35.
- [28] Henning Femmer et al. 'Rapid quality assurance with Requirements Smells'. In: *Journal of Systems and Software* (2017). ISSN: 01641212. DOI: [10.1016/j.jss.2016.02.047](https://doi.org/10.1016/j.jss.2016.02.047).
- [29] Henning Femmer et al. 'Rapid requirements checks with requirements smells: two case studies'. In: 2014. DOI: [10.1145/2593812.2593817](https://doi.org/10.1145/2593812.2593817).
- [30] Donald Firesmith. 'Quality requirements checklist'. In: *Journal of Object Technology* (2005). ISSN: 16601769. DOI: [10.5381/jot.2005.4.9.c4](https://doi.org/10.5381/jot.2005.4.9.c4).
- [31] Donald Firesmith. 'Specifying good requirements'. In: *Journal of Object Technology* 2.4 (2003), pp. 77–87. ISSN: 16601769. DOI: [10.5381/jot.2003.2.4.c7](https://doi.org/10.5381/jot.2003.2.4.c7).
- [32] Don C Gause. 'User DRIVEN design—the luxury that has become a necessity'. In: *A Workshop in Full Life-Cycle Requirements Management*. ICRE. 2000.
- [33] Atul Gawande. *Checklist Manifesto, The (HB)*. Penguin Books India, 2010. ISBN: 0670084409.
- [34] Tony Gorschek and Alan M. Davis. 'Requirements engineering: In search of the dependent variables'. In: *Information and Software Technology* (2008). ISSN: 09505849. DOI: [10.1016/j.infsof.2007.10.003](https://doi.org/10.1016/j.infsof.2007.10.003).
- [35] Brigitte M. Hales and Peter J. Pronovost. 'The checklist-a tool for error management and performance improvement'. In: *Journal of Critical Care* (2006). ISSN: 08839441. DOI: [10.1016/j.jcrc.2006.06.002](https://doi.org/10.1016/j.jcrc.2006.06.002).
- [36] W. Hayes. 'Research synthesis in software engineering: a case for meta-analysis'. In: *Proceedings Sixth International Software Metrics Symposium (Cat. No.PR00403)*. 1999, pp. 143–151. ISBN: VO -. DOI: [10.1109/METRIC.1999.809735](https://doi.org/10.1109/METRIC.1999.809735).
- [37] Khin Hayman Oo et al. 'An Analysis of Ambiguity Detection Techniques for Software Requirements Specification (SRS)'. In: *International Journal of Engineering & Technology* (2018). DOI: [10.14419/ijet.v7i2.29.13808](https://doi.org/10.14419/ijet.v7i2.29.13808).
- [38] Tetiana Hovorushchenko and Olga Pavlova. 'Evaluating the Software Requirements Specifications Using Ontology-Based Intelligent Agent'. In: *2018 IEEE 13TH INTERNATIONAL SCIENTIFIC AND TECHNICAL CONFERENCE ON COMPUTER SCIENCES AND INFORMATION TECHNOLOGIES (CSIT), VOL 1*. IEEE; IEEE Ukraine Sect; IEEE W Ukraine AP ED MTT CPMT SSC Soc Joint Chapter; Minist Educ & Sci Ukraine; Lviv Polytechn Natl Univ; Natl Acad Sci Ukraine; Inst Comp Sci & Informat Technologies; IEEE Ukraine Sect W MTT ED AP EP SSC Soc Joint Chapter; Tech U. 345 E 47TH ST, NEW YORK, NY 10017 USA: IEEE, 2018, pp. 215–218. ISBN: 978-1-5386-6463-6.
- [39] IEEE. *IEEE 830: Recommended Practice for Software Requirements Specifications*. Tech. rep. IEEE, 1998, pp. 1–40. DOI: [10.1109/IEEESTD.1998.88286](https://doi.org/10.1109/IEEESTD.1998.88286).
- [40] International Organization For Standardization. *ISO/IEC 25010:2011 - Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*. Tech. rep. ISO/IEC, 2011.

- [41] ISO/IEC 2011 and IEEE 2011. *ISO/IEC/IEEE 29148: Systems and software engineering — Life cycle processes — Requirements engineering*. Tech. rep. ISO/IEC/IEEE, 2011. DOI: [10.1109/IEEESTD.2011.6146379](https://doi.org/10.1109/IEEESTD.2011.6146379).
- [42] ISO/IEC/IEEE International Standard - *Systems and software engineering – Life cycle processes – Requirements engineering*. Tech. rep. ISO/IEC/IEEE, Nov. 2018, pp. 1–104. DOI: [10.1109/IEEESTD.2018.8559686](https://doi.org/10.1109/IEEESTD.2018.8559686).
- [43] Jan Jürjens. *Secure systems development with UML*. 2005. ISBN: 3540007016. DOI: [10.1007/b137706](https://doi.org/10.1007/b137706).
- [44] Erik Kamsties and Barbara Peach. ‘Taming ambiguity in natural language requirements’. In: *Proceedings of the International Conference on System and Software Engineering and their Applications*. 2000. ISBN: 3-8167-5903-3.
- [45] Erik Kamsties et al. ‘Detecting Ambiguities in Requirements Documents Using Inspections’. In: *Engineering* (2001).
- [46] Nigel King and Christine Horrocks. *Interviews in Qualitative Research*. SAGE Publications Ltd, 2012. ISBN: 9781412912570.
- [47] Robert E. Kraut and Lynn A. Streeter. ‘Coordination in software development’. In: *Communications of the ACM* (1995). ISSN: 00010782. DOI: [10.1145/203330.203345](https://doi.org/10.1145/203330.203345).
- [48] Oliver Laitenberger et al. ‘Experimental comparison of reading techniques for defect detection in UML design documents’. In: *Journal of Systems and Software* (2000). ISSN: 01641212. DOI: [10.1016/S0164-1212\(00\)00052-2](https://doi.org/10.1016/S0164-1212(00)00052-2).
- [49] Giuseppe Lami. *QuARS: A Tool for Analyzing Requirement*. Tech. rep. CMU/SEI-2005-TR-014. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005. URL: <http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7681>.
- [50] Soren Lauesen. *Software requirements: styles and techniques*. Pearson Education, 2002. ISBN: 0201745704.
- [51] Timothy C. Lethbridge, Janice Singer and Andrew Forward. *How Software Engineers Use Documentation: The State of the Practice*. 2003. DOI: [10.1109/MS.2003.1241364](https://doi.org/10.1109/MS.2003.1241364).
- [52] Odd Ivar Lindland, Guttorm Sindre and Arne Sølvberg. ‘Understanding Quality in Conceptual Modeling’. In: *IEEE Software* (1994). ISSN: 07407459. DOI: [10.1109/52.268955](https://doi.org/10.1109/52.268955).
- [53] Olga Liskin. ‘How artifacts support and impede requirements communication’. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2015. ISBN: 9783319161006. DOI: [10.1007/978-3-319-16101-3_{_}9](https://doi.org/10.1007/978-3-319-16101-3_{_}9).
- [54] Wes Martz. ‘Validating an evaluation checklist using a mixed method design’. In: *Evaluation and Program Planning* 33.3 (2010), pp. 215–222. ISSN: 01497189. DOI: [10.1016/j.evalprogplan.2009.10.005](https://doi.org/10.1016/j.evalprogplan.2009.10.005).
- [55] J. Miller, M. Wood and M. Roper. ‘Further Experiences with Scenarios and Checklists’. In: *Empirical Software Engineering* (1998). ISSN: 13823256. DOI: [10.1023/A:1009735805377](https://doi.org/10.1023/A:1009735805377).
- [56] Jakob Mund et al. ‘Does Quality of Requirements Specifications Matter? Combined Results of Two Empirical Studies’. In: *International Symposium on Empirical Software Engineering and Measurement*. 2015. ISBN: 9781467378994. DOI: [10.1109/ESEM.2015.7321195](https://doi.org/10.1109/ESEM.2015.7321195).

- [57] Viktor Pekar, Michael Felderer and Ruth Breu. 'Improvement Methods for Software Requirement Specifications: A Mapping Study'. In: *2014 9TH INTERNATIONAL CONFERENCE ON THE QUALITY OF INFORMATION AND COMMUNICATIONS TECHNOLOGY (QUATIC)*. Univ Porto, Fac Engn; Univ Minho;; Univ Nova Lisboa, Facu Ciencias Tecnologia; ISCTE IUL; Univ Lisboa, Inst Super Tecnico; INOVARIA; TICE pt; Centro Computacao Grafica; AIS SC. 345 E 47TH ST, NEW YORK, NY 10017 USA: IEEE, 2014, pp. 242–245. ISBN: 978-1-4799-6133-7. DOI: [10.1109/QUATIC.2014.40](https://doi.org/10.1109/QUATIC.2014.40).
- [58] A.A. Porter and L.G. Votta. 'An experiment to assess different defect detection methods for software requirements inspections'. In: 2002. DOI: [10.1109/icse.1994.296770](https://doi.org/10.1109/icse.1994.296770).
- [59] Adam Porter and Lawrence Votta. 'Comparing detection methods for software requirements inspections: A replication using professional subjects'. In: *Empirical Software Engineering* (1998). ISSN: 13823256. DOI: [10.1023/A:1009776104355](https://doi.org/10.1023/A:1009776104355).
- [60] REFSQ 2018. URL: <https://refsq.org/2018/welcome/>.
- [61] James Robertson and Suzanne Robertson. 'Volere'. In: *Requirements Specification Templates* (2000).
- [62] Michael Scriven. 'Key evaluation checklist'. In: *Methodology* (2007). ISSN: 2072-6643.
- [63] Michael Scriven. 'The logic and methodology of checklists'. In: *Methodology* (2007). DOI: [10.1080/10705500802365482](https://doi.org/10.1080/10705500802365482).
- [64] Sun Sup So et al. 'An empirical evaluation of six methods to detect faults in software'. In: *Software Testing Verification and Reliability* (2002). ISSN: 09600833. DOI: [10.1002/stvr.248](https://doi.org/10.1002/stvr.248).
- [65] Ian Sommerville and Pete Sawyer. *Requirements Engineering: A Good Practice Guide*. 1997. ISBN: 0471974447.
- [66] Daniel L. Stufflebeam. 'Evaluation checklists: Practical tools for guiding and judging evaluations'. In: *American Journal of Evaluation* (2001). ISSN: 10982140. DOI: [10.1016/S1098-2140\(01\)00119-9](https://doi.org/10.1016/S1098-2140(01)00119-9).
- [67] Daniel L Stufflebeam. 'Guidelines for developing evaluation checklists: the checklists development checklist (CDC)'. In: *Kalamazoo, MI: The Evaluation Center. Retrieved on January 16 (2000), p. 2008*.
- [68] The Evaluation Center (Western Michigan University). *Checklist Validation*. URL: <https://www.wmich.edu/evaluation/checklists/checklistsvalidation>.
- [69] The Evaluation Center (Western Michigan University). *Evaluation Checklists Project Charter*. URL: <https://wmich.edu/sites/default/files/attachments/u350/2019/EvalChecklistsProject-Charter.pdf>.
- [70] Thomas Thelin, Per Runeson and Claes Wohlin. 'An experimental comparison of usage-based and checklist-based reading'. In: *IEEE Transactions on Software Engineering*. 2003. DOI: [10.1109/TSE.2003.1223644](https://doi.org/10.1109/TSE.2003.1223644).
- [71] Jonathan R. Treadwell, Scott Lucas and Amy Y. Tsou. 'Surgical checklists: A systematic review of impacts and implementation'. In: *BMJ Quality and Safety* (2014). ISSN: 20445415. DOI: [10.1136/bmjqs-2012-001797](https://doi.org/10.1136/bmjqs-2012-001797).
- [72] Gursimran Singh Walia and Jeffrey C. Carver. *A systematic literature review to identify and classify software requirement errors*. 2009. DOI: [10.1016/j.infsof.2009.01.004](https://doi.org/10.1016/j.infsof.2009.01.004).

- [73] David A Wheeler, Bill Brykczynski and Reginald N Meeson Jr., eds. *Software Inspection: An Industry Best Practice for Defect Detection and Removal*. 1st. Los Alamitos, CA, USA: IEEE Computer Society Press, 1996. ISBN: 0818673400.
- [74] Karl E. Wiegers. *Inspection Checklist for Software Requirements Specifications (SRS)*. 1999. URL: <http://www.csis.pace.edu/~scharff/cs3892005/reqreview.doc>.
- [75] Roel Wieringa. *Design Science Methodology for Information Systems and Software Engineering*. 1st ed. Springer-Verlag Berlin Heidelberg, 2014. ISBN: 978-3-662-43838-1. DOI: [10.1007/978-3-662-43839-8](https://doi.org/10.1007/978-3-662-43839-8). URL: <https://www.springer.com/gp/book/9783662438381>.
- [76] Roel Wieringa and Maya Daneva. ‘Six strategies for generalizing software engineering theories’. In: *Science of Computer Programming*. 2015. DOI: [10.1016/j.scico.2014.11.013](https://doi.org/10.1016/j.scico.2014.11.013).
- [77] Dietmar Winkler, Bernhard Riedl and Stefan Biffl. ‘Improvement of design specifications with inspection and testing’. In: *Software Engineering and Advanced Applications, 2005. 31st EUROMICRO Conference*. 2005. ISBN: 0769524311. DOI: [10.1109/EUROMICRO.2005.30](https://doi.org/10.1109/EUROMICRO.2005.30).