CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Requirement Strategy in Large-Scale Agile Development

A Design Science Research

Master's Thesis in Computer Science and Engineering

NASSIBA EL HASKOURI

DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's Thesis 2021

Requirement Strategy in Large-Scale Agile Development

A Design Science Research

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology University of Gothenburg
Gothenburg, Sweden 2021

Requirement Strategy in Large-Scale Agile Development
A Design Science Research
NASSIBA EL HASKOURI

Supervisor: Eric Knauss, Department of Computer Science and Engineering
Examiner: Gregory Gay, Department of Computer Science and Engineering

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg Sweden
Telephone + 46 (0)31-772 1000

Gothenburg, Sweden 2021

Requirement Strategy in Large-Scale Agile Development

A Design Science Research
NASSIBA EL HASKOURI

Department of Computer Science and Engineering
Chalmers University of Technology
and University of Gothenburg

Abstract

[**Context and Motivation**] Agile methodologies are created to facilitate short time to market. Large-scale system development such as complex products and embedded systems are not fairly treated by agile, especially with respect to managing stakeholders and system requirements. Despite approaches to agile requirements engineering, organizations are still facing many challenges regarding system development.
Traditional requirement engineering approaches offer a detailed analysis including elicitation, documentation, evaluation, and management, yet it is difficult to integrate them with agile ways of working.
[**Method**] Based on the design science research, we highlight critical challenges collected from a workshop and eight interviews. Aiming to find applicable solutions such as artifacts, practices, methods, or tools, we also validate those solutions by conducting five evaluations sessions.
[**Result**] The end result is a new artifact that we call a requirement strategy and two other supplements: technical templates and traceability tools.
We conclude that using the requirements engineering approach allows us to solve several challenges. In our case we produce a powerful artifact aiming to help agile teams especially if they are working with large-scale agile system development, the solution might be generalized to companies that are developing large scale software systems. Also, it is an encouragement to the industry to use requirement engineering even if they are using agile methodologies.

Keywords: agile requirements engineering, requirements strategy, design science research

Acknowledgements

Firstly, I would like to thank my supervisor Eric Knauss for all his effort and engagement in giving guidance and invaluable feedback throughout the study.
Secondly, a sincere thank you to all employees from Verisure Innovation AB in Linköping and Malmö, especially my ex-colleagues for their support and engagement on participating in this study.
Thirdly, I would also like to especially thank my father who supported my studies since I was a kid. Also, to all my family especially my kids Yahya and Maryam.
Finally, I would thank Gregory Gay for agreeing to be the examiner of this study.

Nassiba El Haskouri, Gothenburg, June 2021

Table of Content

1 Introduction

Reducing time to market speed often requires changing requirements during the sprint according to the business needs, which is possible within agile development methodology. One of the pillars from Agile manifesto is: “Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.” [1] However, Agile teams are facing many challenges regarding the requirement because of agile approaches [2][3]. For instance, some of those challenges are Missing high-level requirements, misunderstanding the customer needs and the difficulty of defining the right roles to have enough communication [3][4].

Nowadays, companies are struggling to implement agile requirements engineering [4][5]. Agile RE documented by short simple user stories works fine to deliver iteratively small features to the customer. On the other hand, in case the company implements system requirements for complex systems that might be a mixture of software and hardware, using agile RE would fail facing many critical challenges [3]. It is known that this problem occurs basically from transitioning to agile system development [3]. Additionally, many researchers stated that agile practices support traditional requirements engineering [7][9][21]. However, companies still do not implement traditional requirement engineering in large-scale agile development. Hence, in this thesis we intended to create artifacts that would help companies to practice traditional requirement engineering principles in the scope of large-scale agile development.

The purpose of this thesis is to spot the critical challenges that influence the quality of the requirements during the development process aiming to study all factors that influence the requirement handling within the case company. Through applying the research methodology design science research, we started with accumulating major challenges by collecting the data from the conducted workshops and interviews. Then elaborating applicable systematic solutions that might be artifacts, practices, or tools. Finally, to ensure the quality of all solutions four evaluation sessions were conducted.

The implemented solutions might help to improve the way of handling requirements using agile methodologies. Also, the study will be significant for all practitioners and researchers that are interested to see solutions validated by a company. Additionally, the research can provide a data point to overcome the lack of empirical foundation for RE in Agile [8].

1.1 Statement of the Problem

Several techniques exist that can be used during the requirements analysis, however the market speed influences the way of processing with requirements, and it is challenging to invest enough time to get proper requirements. The traditional requirement analysis begins with applying the elicitation techniques to find and formulate requirements. Common practices include stakeholder analysis, task demonstration, and conducting interviews or focus groups [9]. Presenting the analysis that are the results of the elicitation, the interpretation and documentation techniques are needed to filter the candidate requirements and pick up the suitable format that fits the requirements typology. Requirements analysis also covers negotiation to define the business values for all tentative requirements [6] (see the Figure 1 below). Since the agile methodology is not a good match for traditional ways of handling requirements engineering, this step is skipped and replaced with writing small user stories often with a lack of requirements analysis.

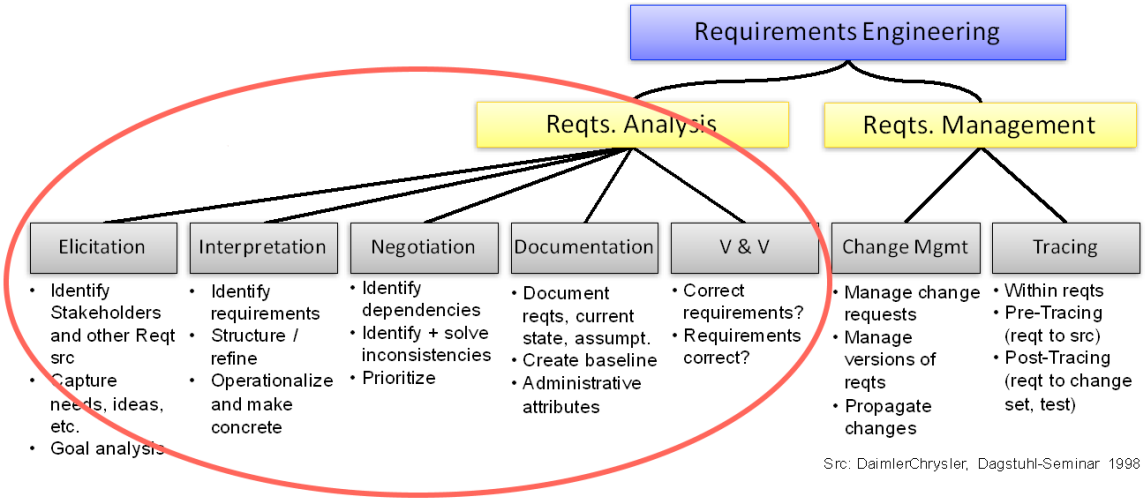


Figure 1: Overview of common requirements engineering activities

Research exposed different challenges about requirements within agile development [2][3][4]. Those challenges often relate to prioritizing speedy delivery over elaborating perfect requirements which might lead to the phenomena technical debt such as code debt (code smells), design debt, documentation debt, and people debt [51]. Technical debt is the accumulation of cruft in a software system that makes it hard to modify or add new features [52]. In addition to technical debt there is another phenomenon that seems accepted in companies which is spending more time reporting and fixing bugs instead of adding new features and increasing the quality of the software [9]. However, few proven solutions were provided [3][8]. In fact, there is a lack of overview about the relevant challenges that exist in practice and at the same time, a lack of proven solutions or even a solution strategy is missing [3][8]. Therefore, studying these challenges is crucial to define new solutions that fit nowadays market speed.

1.2 Research Question

To reach the purpose of this study that is elaborating the critical challenges and finding solutions to them; we conducted qualitative research to answer the following research questions:
RQ1. What are the critical challenges with requirements engineering in agile development?
RQ2. What practices, methods, or tools are potential solutions to critical challenges with requirements engineering in agile development?
RQ3. What critical challenges with requirements engineering in agile development can be addressed by specific practices, methods, or tools extending those practices, methods, or tools?

1.3 Purpose of the Study

The purpose of this study is to present the critical challenges particularly with the case company Verisure Innovation AB apropos how the requirements engineering among agile development are created and processed aiming to analyse and bring up practical solutions that may help and support agile teams with handling the requirements.
The contribution of this thesis is to highlight the critical challenges for the requirements handling specifically for Verisure Innovation AB that is involved with the time to market strategy. The study proposes a requirements strategy that aims to increase awareness and support with the provided solutions for all agile teams (Product Owner, Scrum master, developers, testers) within

the Verisure Innovation AB also for all researchers that are interested to see new solutions for those challenges.

2 Background and Related work

2.1 Requirements Engineering

"Requirements engineering is the process of eliciting stakeholder needs and desires and developing them into an agreed-upon set of detailed requirements that can serve as a basis for all subsequent development activities" [52]. The benefits of using requirements engineering as methodology is to transform an unknown problem to a clear and complete requirement that must be understandable by all stakeholders, the provided requirements are mainly potential solutions to the problem. Thus, elaborating requirements statements and producing the system specification is the purpose of requirements engineering [52]. However, elaborating solutions and completing requirement specifications before the development stage is not welcome in agile development. Even if using the requirements engineering techniques provide high quality requirements, still the Agile community considers involving requirements engineering in the development process is time consuming. On the other side, new researchers mentioned that there is a need to apply requirements engineering to elaborate system requirements [3].

2.2 Agile Requirement Engineering

Bjarnason et al. [14] studied the use of agile RE, investigating the use of agile RE in a case study with nine participants from a large-scale company that was recently transitioning to agile. The study highlights that agile methods face some classical RE challenges, for example the communication gaps between teams. In another study, that compares traditional RE and agile software development [50]. It is concluded that agile methods and RE are sharing similar goals especially the stakeholder involvement. From a systematic mapping of 28 articles, the author reported that there is no universal definition of agile RE. Further there are several problems in agile RE such as the lack of getting the customer involvement, the requirements prioritization and growing technical debt [8]. In a related work, a multiple-case study that was conducted on seven large-scale agile system development companies, authors concluded that there is a need for strong requirements engineering approaches, especially to document a system's behaviour or maintenance [3].

Inayat et al. define the term "Agile requirements engineering" as an agile way of planning, executing, and reasoning the requirements engineering activities [9]. Yet, the definition of the term "Agile RE" is still weak inside companies that are incubating agile development methodologies [11] and often, such companies rely only on user stories and product backlogs as requirements engineering [10].

From a systematic literature review, Inayat et al. grouped and presented the practical challenges of agile RE [9], some of those challenges are presented below:

1- Lack of documentation is a critical challenge that agile poses to the development teams [11]. However, Face to face communications and all kinds of verbal communications are insufficient especially with large projects [9].
2- Customer availability, especially when the customers are from different countries.
3- Budget and schedule estimation, it is hard to provide the concrete estimation for a large project which can influence the decision about the budget.
4- Finalised features in earlier stages of the project become inadequate in later stages with new requirements [7].
5- Neglecting non-functional requirements, and ignoring the quality …
6- Agile methods welcome change the requirements, but it can create trouble when evaluating the consequences of these changes.

Organizations are still searching for solutions for these challenges, and still, there is lack of information about how to address the challenges related to requirements engineering within collaboration-oriented agile methodologies [9]. It has been reported that traditional RE and agile methods can fit well together however there are still different opinions about the amount of documentation that is required [11]. Thus, there is still controversy about how Agile methods have many challenges regarding traditional requirements engineering, and this theme is the actual interest to software practitioners and researchers [9].

2.3 Requirement Engineering & Agile Methods

The mindset and values that requirements engineering is sharing with agile methods is the most important in both sides which is making the end user of the software happy. However, some values and mindset distinguish requirement engineering from agile methods [21]. The approach that combines both requirements engineering concepts and agile development processes can deliver a successful agile project with a good quality [21]. Furthermore, the requirements engineering practices are useful for some agile principles and techniques regardless of which the applied specific development method is [21]. In fact, both approaches have a common goal that is delivering good quality software [21]. The requirements engineering provides the proper methods and techniques that can transform the stakeholders' needs or problems to the right solution that can be a feature added to a software or the whole software itself. In other side agile methodologies aim to deliver software in time with an efficient way [21]
Software engineering teams are now widely practicing agile methodologies. Agile is a method for helping teams to develop and deliver solutions to the market of software applications, aiming to achieve high quality and increase the software business value [1], [16]. Here below the Table 1 shows what are the similarities in between RE and agile requirements [21]:

Requirement Engineering Principles [21]	Agile Principles [1]
"Relevant understanding of the users' needs to develop valuable software".	1st principle: "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software".
"Proper tools to recognize changes in the market for the stakeholders' competitive advantage".	2nd principle: "Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage".
"Proper tools and techniques to foster efficient collaboration between stakeholders and developers".	4th principle: "Business people and developers must work together daily throughout the project".
"Proper tools and techniques to support verbal communication".	6th principle: "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation".
"Relevant understanding of the stakeholders' desires and needs to minimize the development of unnecessary software".	10th principle: "Simplicity--the art of maximizing the amount of work not done--is essential".

Table 1: RE in relation to Agile Principles

3 Research Approach

3.1 Research Methodology

3.1.1 Design Science Research

The chosen research method for this research is Design Science Research [13]. According to Hevner et al., "The design-science paradigm seeks to extend the boundaries of human and organizational capabilities by creating new and innovative artefacts" [15]. Thus, design science is a good fit for the main goal of this thesis to improve the way of handling requirements within agile methodologies. This will also allow contributing to additional knowledge questions, including to identify the challenges and problems that agile teams are struggling with. The results will be an artifact that combines specific solutions to those challenges as well as knowledge about the extent to which they can solve relevant challenges. The artifact and answers towards the knowledge questions are developed, verified, and validated iteratively [13]. Below see the Table 2 that shows the three cycles in combination with what can be done for answering the research questions:

	RQ1 - Challenges	**RQ2 - Tentative Solutions**	**RQ3 - Evaluation**
Cycle 1	Workshop	Systematic work to derive requirements strategy from workshop data	Solutions An example of the requirement strategy
Cycle 2	Interviews	How to write technical issues as user story	Template for technical issues
Cycle 3	Interviews	The benefit of test strategy and how it is related to requirement strategy	Traceability between both strategies

Table 2: The three cycles and the research questions

3.1.2 Research Cycles

The artifacts, methods, practices, or tools would be constructed using design science research methodology by going through three cycles. The iterative approach provides continuous feedback to improve the developments and the progress for each solution. By following the design science research three principal patterns were identified: challenges, solutions, and evaluation.

- **Cycle I:** The first cycle focuses on spotting major challenges that influenced handling agile requirements among agile developments from the people that report requirements (Application specialists), a solution would be provided for the most critical challenges and

to validate the solution an evaluation session would be conducted to assure that our solution was valid for the case company.

- **Cycle II:** The second cycle focuses on getting more challenges from a complete agile team (Product owner, Scrum master, and developers) that might be not discovered in the previous cycle, a solution would be provided for crucial challenges as the participants requested and to validate the solution an evaluation session was conducted to assure that our solution was valid for the case company.
- **Cycle III:** The third cycle focuses on getting more challenges from testers to make sure that all staff that were interacted with the requirements somehow present their point of view, a solution would be provided for challenges that participants requested and to validate the solution an evaluation session was conducted to assure that our solution was valid for the case company.

3.2 Data Collection

The case company is Verisure Innovation AB, Verisure is the leading provider of security and protection to residential and small business customers across Europe and Latin America. Verisure produces professionally monitored smart alarms and service to 3.7 million customers in 16 countries across Europe and Latin America, with a team of more than 20,000 employees. Verisure innovation AB has two offices in Sweden that work on software development, one in Malmö and the other in Linköping.
Most of the data collected is from participants that are working in the Linköping office and only two participants are from Malmö.

This research is based on qualitative data. a workshop and a semi-interview were conducted with a total sample size of 15 participants:
- Study the collected data from the previous studies, to elaborate on the correct questions for getting data that are needed.
- Conduct a Workshop using Card sorting to spot and prioritize the challenges with application specialists who do the analysis and write the user stories in Jira.
- Semi-structured interviews would be conducted with four testers from different agile teams to highlight if the challenges are related to a specific team or that apply to all teams.
- Semi-structured interviews would be conducted with an agile team (Product Owner, Scrum master, 5 developers, and 3 testers), to analyse in close the flow of agile RE within an agile Team. See the Table 3 below that presents an overview about the participants, their role, and the sample size.

Participants	Role	Sample size
Application specialist	Software analyst, reports bugs, reports user stories to Dev team	4
Product Owner	Software analyst, reports bugs, report user stories to Dev team	1
Scrum master	background: Senior tester	1
Software architect	Senior developer, provide solutions as an architect	1

Developers	• Three senior developers • One junior developer	4
Testers	Senior testers	3
Total		14

Table 3: The participant's role

3.2.1 Workshop

In general, a workshop is a usually brief intensive educational program for a relatively small group of people that focuses especially on techniques and skills in a particular field [22]. In research, a workshop is a research approach that has the potential to summarise a fruitful discussion with an agreement between researchers and participants [23]. It can be used as a methodology to reveal what is the problem or for creating solutions.
In this study, grouping the participants in one room and letting them expose their problems was the aim. Before inviting participants to the workshop, the sample that was selected to participate was people who wrote user stories to agile teams (Product owners and application specialists). Those people were already aware of the objective of the thesis and were prepared before the workshop. An email was sent to the participants with written information about the thesis and which method will be used during the workshop.
The method that has been used is card sorting, card sorting is a methodology that "makes tacit information explicit" [32]. It is known that card sorting is used in end-user system design and user testing [32], however it can also be used as a tool for elicitation technique and conceptual modelling [33]. Each participant had a different colour of cards. They were given 15 minutes to write what are the challenges that they are facing when they are writing the user stories to the agile teams. Finally, 42 cards were accumulated from all participants. Since questions 1, 2 and 3 have gotten similar answers from the participants thus there was no need to use card sorting only question 4 that card sorting was used for.

The main questions that were treated during the workshop are:

1. How do you get the request that this issue should be in the backlog?
2. How is the format of the issue before you write it as a Jira ticket?
3. What would you do before writing an issue?
4. What are the challenges that you are facing when you write user stories or Jira issues in general?

3.2.2 Interview

To collect data for cycle two and cycle three following the design science research several interviews were conducted with the case company. After getting data from the workshop,
The semi-structured interview method was chosen to dig deeper on the challenges that were explored in the workshop. Basically, the semi-structured interview was chosen for planning the questions that need to be asked at the same time. It is not mandatory to follow the order of the questions list, instead the development of the conversation leads us to what the question is next [24]. Thus, the semi-structured interview helped us to explore the studied artifacts. The interview questions are divided into three categories: Interview questions for the product owner and application specialists (See Appendix section 9.2),

Interview question for developers and software architects (See Appendix section 9.3) and interview questions for testers (See Appendix section 9.4).

3.2.3 Evaluation

According to Hevner et al evaluating the artifact's "utility, quality, and efficacy" allows future work to be coordinated [15]. The evaluation of the artifact in this thesis was done within an iterative process as the design science research recommended. In Cycle I and II an evaluation session was conducted with some participants (Senior developer, Senior Tester & Scrum master, two application specialists). By using the concept of *Verification & Validation* From the requirement engineering approach [19] a technique was used called checklist forms. We included all the provided solutions in the checklist forms and exposed them to the participants (see appendix section 8.5 checklist forms). First, we explained the solutions then the participants gave us feedback and validated the solution. In Cycle III an evaluation session was conducted using PowerPoints for presenting the traceability tools with examples from requirements linked to their test cases. Two traceability tools were evaluated by one expert tester.

3.3 Data Analysis:

The data was analysed iteratively as recommended in design research.
In each cycle, the data will be analysed to give an answer to all research questions. Here below is the planning for those cycles:

1- Study all challenges from previous research and run the workshop with the application specialist from the company. Aiming to prioritize critical challenges that will be focused on within the next cycles. At the end of this month, all the findings should be documented.
2- Find a solution for the most pressing challenge, implement it, and evaluate it against the challenges. Document findings concerning the research questions and update the knowledge about challenges if needed.
3- Consider an additional cycle and go back to Step 2.
4- Document all results and finalise all the thesis contents.

3.3.1 Workshop Analysis

The data collected during the workshop is mapped and categorized by:
1- Eliminating the repeated topic and keep one card representing the topic
2- Giving a number for each card.
3- Group the cards that have the similar topics and the explanation from the participant.
4- Colour the cards with the same colour (no need to know who writes what...)
5- Give the name of the artifact for each group.
The extracted data from the card sorting in general was concentrated on three different themes: requirement strategy, Agile development, Architecture, and organisation (See the image in Appendix Section 9.1).

3.3.2 Interviews Analysis

Since the interview questions were inspired basically from the data that we have gotten from the workshop and the questions were formulated according to the codes that were collected in the workshop. The format of the data that was gotten from interviews was a bunch of phrases and descriptions. Thus, qualitative data analysis was needed to form an artifact.
In this phase, the approach of coding all data collected was conducted. Coding is the process of organizing the data by bracketing pieces of text or image segments and writing next to it a word that describes a meaning or a category in the margins [25]. Moreover, the process starts with

copying text data gathered during an interview, segmenting sentences, or paragraphs into categories, and putting a label for each category as a term. The term is basically picked up from the actual language of the participant called an in vivo term [26].

3.3.3 Coding

For coding the collected data, a combination of a priori and emergent coding was conducted and completed with a descriptive coding. A priori and emergent coding was applied for identifying new ideas based on descriptive codes for reasons to categorize the challenges in a meaningful way. A priori coding is a method of creating code based on a provisional developed "start list" of code prior. The list might be the results from literature review, conceptual framework, list of research questions, hypotheses, and problem areas [48]. In our case, running the workshop in the beginning of the research process provided us with a long start list that also helps to generate more a priori codes.

Emergent coding is for identifying the themes and their connection with each other aiming to explore more about the artifact in the case study. Predetermined coding may be based on a previous coding dictionary from another researcher or key concepts in a theoretical construct. They may derive from the interview guide or list of research questions.

When conducting emergent coding, the following steps were applied [29]:

1. Initial review:
    - Read through a subset of the data with a general question in mind. Starting with a small part of data.
    - When themes emerge from the data, highlight them in a particular colour, it can help to find examples when reporting findings.
2. Note themes:
    - After identifying a theme, document a description of the theme aiming to differentiate between themes and instances of previously identified themes.
    - Go back through earlier transcripts to find this theme.
    - review a subset of transcripts [2]

Descriptive coding is also called "topic coding" similar to hashtags [28]. This method is for first cycle coding that starts with reading through the data and coding text according to the meaning and the category. The format of Descriptive codes is often a noun, and summarizes the meaning of the data [27]. In this study, we used Excel to code the data collected and according to Saldana these are the following steps that were followed [28]:

- Read through the data and identify what is the subject that showed up in the data
- Create codes for each subject
- Code excerpts according to the subject
- Gather all the excerpts together that are related to each descriptive code

Finally, an index of codes was obtained, and based on codes a set of themes were accumulated.

4 Results

In this chapter, all the presented findings following systematically design science research, we started with identifying the challenges, finding solutions that might be artifacts, practices, or tools. Finally, to validate our study four evaluation sessions were conducted to ensure the significance of our solutions.

4.1 Requirement Challenges (RQ1)

4.1.1 Challenges Uncovered in Cycle I

Based on the data obtained from the workshop, there are several themes related to the beginning of the process of creating the requirements. Those themes are challenges for the participants that have been present in the workshop (See below the table 4).
Those challenges belong to various aspects for instance to Elicitation, Communication, Documentation, and Requirement management. From the Image that represents all themes (See the image in Appendix Section 8.1), The similar themes were categorized and coloured with the same colour see below figure 1. Each layer was given a significant name that might summarize the challenges.

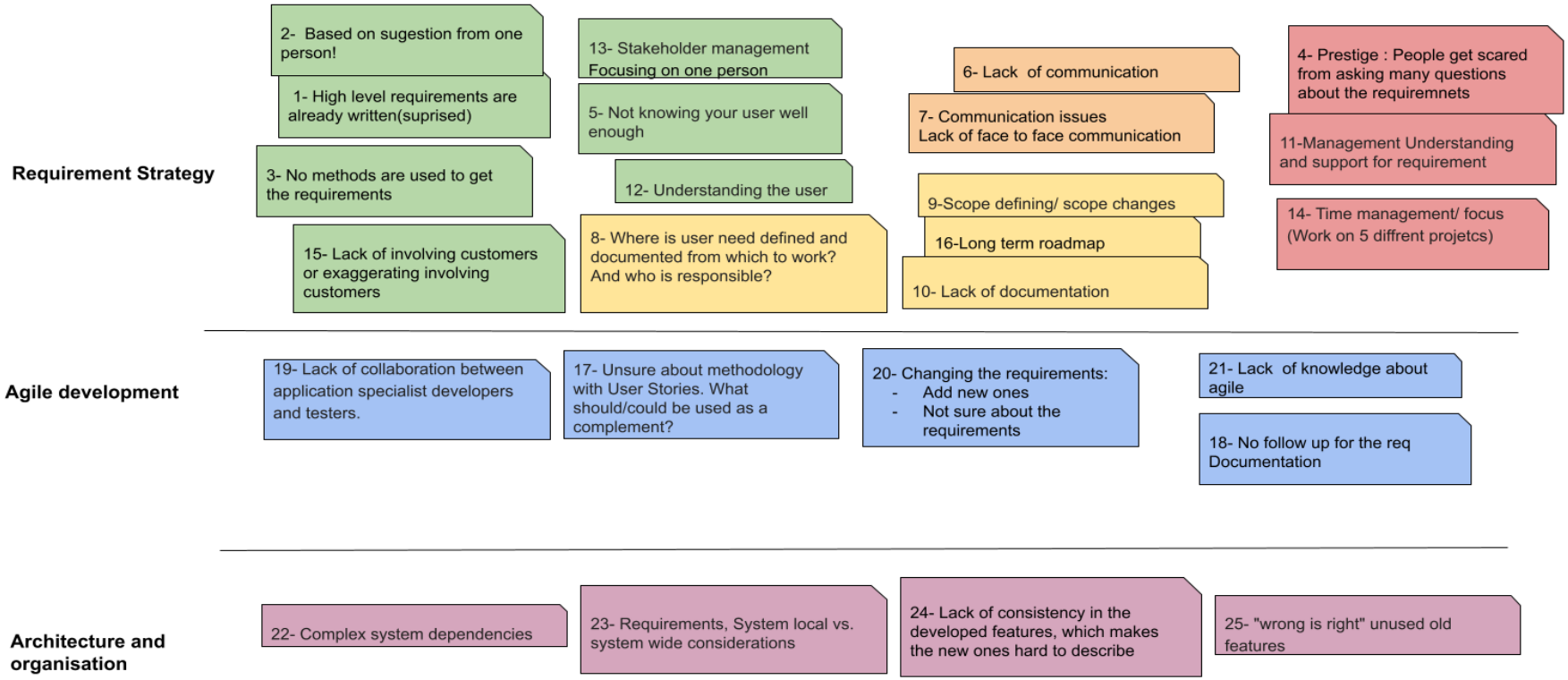


Figure 2: Themes emerging from the workshop

To continue processing with themes emerging from the workshop, similar challenges with the same colour were gathered and moved with their ID to a table and gaved tentative solutions (See below the Table). The challenges in the first row are mapped from green cards, they are related to the analysis before documenting the requirements, which we intended that the Elicitation & Validation is needed to help the application specialists that were confused about how handling the requirements is processed. For instance, some of those challenges are: Getting requirements and solutions already documented from stakeholders as high-level requirements and the lack of involving stakeholders limited to only one person. The second row from the reed cards is related to problems with communication. Participants expressed their concerns about the lack of communication within the team and cross-functional teams affecting the quality of their work. In the next row from orange cards, Participants stated that the documentation is very limited and there is no baseline for documentation among the process of handling the requirements. The challenges from the fourth row (ping cards) are related to the requirement management that is needed to settle the management that would support and help product owners and application specialists to do their analysis in practical conditions. The grey rows are challenges related to agile developments and architecture & organization, which we intended not to include in this research aiming to focus on the challenges related to requirements engineering.

Challenges	**Tentative solutions**
High-level requirements are already written (1) Requirements are based on suggestions from one person (2) No methods are used to get the requirements (3) Not knowing your user well enough (5) Understanding the user (12) Stakeholder management, Focusing on one person (13) Lack of involving customers or exaggerating involving customers (15)	Elicitation & Validation
Lack of communication (6) Communication issues, Lack of face-to-face communication(7)	Communication
Where are the user needs defined and documented from which to work? And who is responsible?(8) Scope defining/ scope changes (9) Lack of documentation(10) Long term roadmap(16)	Documentation

Prestige: People get scared from asking many questions about the requirements (4) Management Understanding and support for the requirement (11) Time management/ focus (Work on 5 different projects)(14)	Requirement managements
Unsure about methodology with user Stories. What should/could be used as a compliment?(17) No follow up for the requirement documentation(18) Lack of collaboration between application specialist developers and testers(19) Changing the requirements: Not sure about the requirements and Add new ones(20) Lack of knowledge about agile (21)	Agile RE & agile development
Complex system dependencies(22) Requirements, System local vs. system-wide considerations (23) Lack of consistency in the developed features, which makes the new ones hard to describe (24) "Wrong is right" unused old features (25)	Architecture & organization

Table 4: The themes from the worksh

Requirement information model

In addition to the data that we have obtained in the workshop, exactly from questions 1,2 and 3 that were asked during the workshop (see Section 4.3.1 Workshop), we intended to model how the process of the requirement goes through the organization, from getting the high-level requirements to the release.

To present and model the data Eclipse Modelling Framework (EMF) was used as a tool to model an Ecore meta-model using the core technology in Eclipse for model-driven engineering. EMF allows the definition of metamodels based on the metamodeling language called Ecore [30]. We managed to draw a metamodel that is an abstract syntax from the collected data. It composes many elements, attributes, relationships, and the rules for combining these concepts to build or complete meta-models [30]. Figure 3 presents the meta-model (Requirements information model) that presents a walkthrough of stories from High level Requirements to Release.

By providing a structure for requirements-related information and by representing the relationship and responsibilities of the product owner or the application specialist roles with respect to processing requirements, the requirements information model helps to address challenges with respect to communication, documentation, and management of requirements. From the analysis, it appears that the product owner is the engine that moves all types of requirements from the start of the process until the end of the release. The process started when the product owner got the high-level requirements from different countries whose format might be one phrase, two words or very detailed requirements that already proposed the solution to agile teams. Thus, the PO should conduct the analysis for all projects and convince all stakeholders with his solutions and refine the requirements before pushing them to the sprint. Unfortunately the product owner and application specialists are struggling with many challenges as described in the section above. In addition to that, stakeholders relied only on the PO and they have almost no direct connection with the scrum.

The requirements information model showed that the issue might be a feature or bug. The templates are limited to two templates that are feature templates and bug templates and both of them are integrated to Jira. The feature template is a guideline on how to write a user story and it contains:

- *Background*: a short description about the state of the system before the development.
- *User story:* expresses both user requirements and system requirements, and it might be a whole project written as a user story.
- *Acceptance criteria*: there might be some extra requirements needed to be implemented or some scenarios that need to be covered by the development.

Filling a feature template without categorizing which type of requirements that suit this kind of template, is one of the core challenges that the product owner and application specialist described. The bug template covers:

- *Bug description*: a short text defining what is the bug, the bug might be about the functionality developed, a quality requirement as performance.
- *Step by step:* how to repeat the bug.
- *Expected results*: what is the correct behaviour.
- *Rollback scripts*: provide a rollback to the previous releases.

All the issues are discussed in backlog refinement or backlog review. In addition to product owners and application specialists, developers might report technical issues, correct and change the requirements during the sprint because of the misunderstanding of the requirements in the backlog refinements.

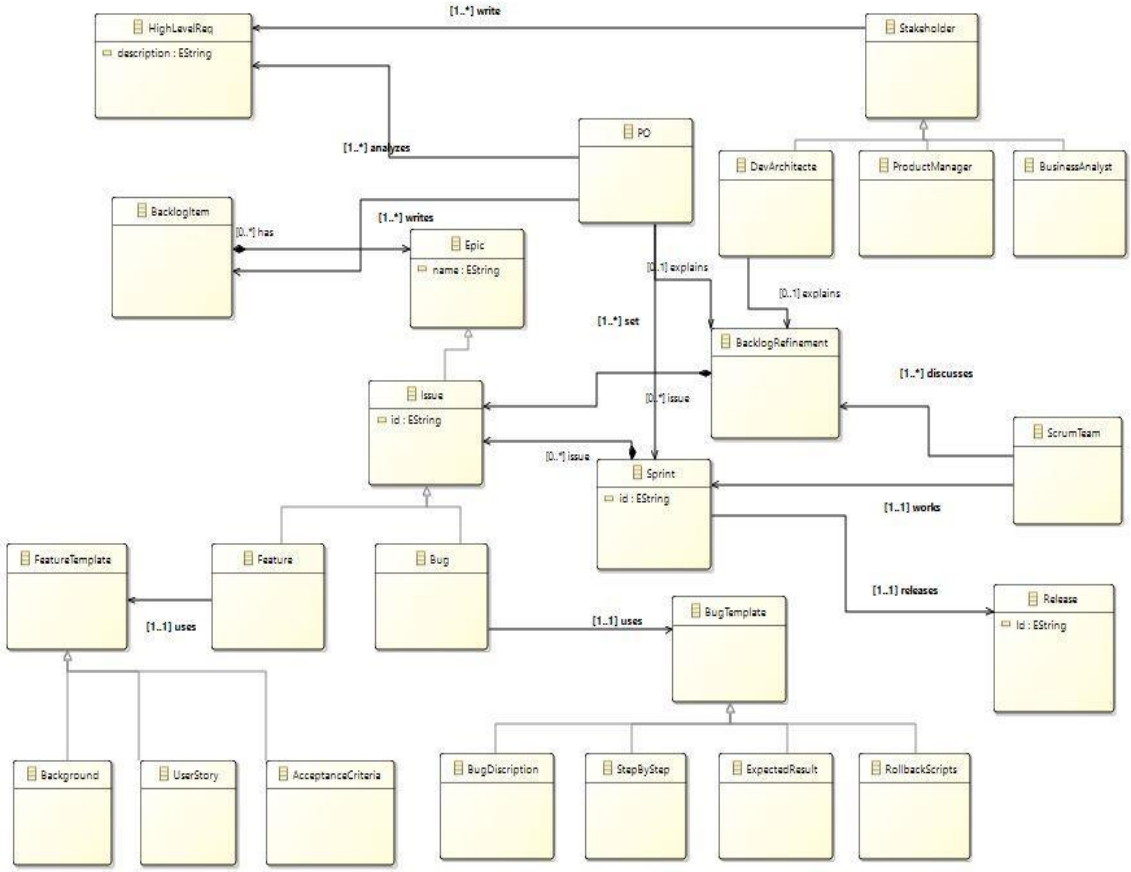


Figure 3: Requirement information model

HighlevelReq: is the bank where all the high-level requirements are stored. All stakeholders can write a ticket of what they need and push it to a PO or Application specialist.
BacklogItem: It composes all issues, even the ones that are ready for analysis.
Epic: each project is an epic specified with a name.
Issue: a created user story or feature that is reported by the PO or a developer. It can be a Bug issue.
Feature template: the reporter uses the template to fill into it: Background, UserStory, and Acceptance criteria ...
Bug template: the reporter uses the template to fill into it: BugDiscription, StepByStep, ExeptedResults and rollbackscript.
BacklogRefinement: PO, DevArchitect, and the scrum team meet every week to discuss the issues that will be in the incoming sprint.
Stakeholders: Internal stakeholders for the case company (business analyst, management, project manager...).
From the presented challenges from the workshop and the requirements information model, we concluded that the product owner and application specialist needed some practical solutions that could help them with the current circumstances. Solutions that might support the product owner to gather stakeholders and scrum teams as one unionen and build a transparente procedure of processing the requirements.

4.1.2 Challenges Uncovered in Cycle II

After analysing the data collected from the semi- structured interview (See the interview questions in appendix sections 8.2 & 8.3 Interview questions). The table below shows the collected codes and the themes:

Codes	Themes - Challenges
● Difficulty writing user stories ● irrelevant user stories for technical issues ● Several templates are needed for presenting technical issues ● Lack of Documentations	System requirements
● Features concentration ● Project related to one developer ● Stressed timeline	Solo developer
● Lack of discussing solutions before implementation ● No pre-backlog review ● Hard to keep remember issues from backlog review ● Waist time do re-analyses for issues during the development	Issues preparation

• Lack of involving stakeholders • Dislike scrum • Ineffective meetings • Time management • Scrumish (Not all scrum rules are applied) • Stakeholders unaware about how scrum is working • Difficult to make a long-term estimation	Scrum

Table 5: The themes from semi-structured interview (Cycle II)

Participants explained their concerns about the platform that they are using for writing user stories. There was only one template that was used by all reporters to write user stories and for all types of features, even projects. A participant expresses "I wish that there is a template that helps me to write technical issues. I am always struggling with technical requirements and how I should present them to the team". Another participant said, "It is not realistic to have one template for all kinds of stories, I think we should have several templates depending on the type of the feature". Moreover "Having the same template for mobile app, web page, SBN application and signal processing is not relevant…". On the other hand, a participant added that "there is always a lack of information in user stories" and "The user stories with backend features are always missing analysis, "I spend time doing my analysis so I can find a solution in agreement with the product owner".

From the presented challenges, we intended to focus on two challenges: System requirements and issues preparation, aiming to give a solution that would simplify the reporter task of writing requirements. Providing several templates that cover many patterns may help the requirements reporters choose the feasible template. These in particular would decrease the waste of time of confusion that the development team are facing trying to understand the requirement.

The templates would address various requirements, some were inspired by the requirement engineering approach and others from Agile requirements (user stories).

4.1.3 Challenges Uncovered in Cycle III

After conducting a semi- structured interview with four testers (see the interview questions in appendix sections 8.4 Interview questions), the table below shows the collected codes and the themes:

Codes	Themes - Challenges
• All Test information should be written in the issue • Present the risks • Several temples could help • Good Documentations is a must • Everything should be included in an issues • Test plan can be added	Template

• No follow up for the test strategy • The teams have the test strategy in their mind • Testers need to be reminded about the test strategy	Test strategy
• Lack of involving stakeholders • Apply scrum as you want • Stakeholders unaware about how scrum is working • Gaps between stakeholders needs and scrum teams	Management

Table 6: The themes from semi-structured interview (Cycle III)

Testers described their concerns about not following any specified test strategy and with the expected time frame to complete the test, it was usually forgotten or ignored. The test community management was aware about the importance of elaborating a test strategy or a test plan and started preparing for the test before the issue was in the test column. Unfortunately, "the complexity of the requirements presented in backlog review did not help the tester to understand what to test and how" mentioned by a tester. Another tester expressed that "testers used to do tests the same way, asking them to elaborate a test strategy for each project is a waste of time".
In this research, we included the test plan in all provided templates (see section 4.2.3 Technical templates) thus to visualize the agreed followed test strategy to the product owner and development team. In this cycle, we dug deeper on how we could trace the requirement with the test. The solution would be providing methods or tools that might elaborate a traceability relationship between requirements and tests.

4.2 Practices, Methods, and Tools (the Artifact, RQ2)

In this section, we presented the solutions for Cycle I, Cycle II, and Cycle III. by following the design science research, an artifact is created as a solution for challenges gotten in cycle I, we called it *Requirement Strategy*. Moreover, a supplement practice is elaborated in cycle II. It is about four technical templates that would help report complicated technical issues. Finally three traceability tools were recommended to keep tracking the implemented requirements during the development life cycle.

4.2.1 Requirement Strategy

All the teams at Verisure have their own test strategy as an outline for how the team should plan their test. The Test community in Verisure defines the test strategy as a guideline to be followed to achieve the quality assurance objectives. It contains for instance test objectives, test environment, test approach, automation tools and risk analysis. On the other hand, the requirements engineering community believes that the quality of the requirements determines if the product development process is successful or not regardless of which development methodology was applied [21]. Probably elaborating a requirement strategy would be linked to the test strategy somehow. In our case the requirement strategy will be guidelines for application specialists and Product owners of how to create agile requirements engineering step by step.

After analysing the challenges from the data, the artifact for the first cycle following the design science research is elaborating a requirement strategy.
The term “Requirement Strategy” is not familiar in organizations and research, and since it is inspired by the test strategy, we defined the term requirement strategy as an outline and guidelines for Product owners, application specialists, and all people that are involved with writing agile requirements to help them create requirements that are correct, complete, unambiguous, consistent, understandable, and traceable [19].
Since the requirements engineering approach requires a continuous process that must be performed systematically [18]. Thus, the requirements strategy approach presents a step-by-step plan of how to proceed with generalizing requirements. Additionally, this thesis proposes that the requirements strategy should describe methods that help to understand stakeholders' desires and needs and it points to which appropriate techniques can be used for elicitation in different steps of analyses. Also, it concludes each elicitation with verification and validation aiming to achieve better requirement quality.
The requirements strategy shall be documented, and it might vary from the type of the project or for a single feature. Additionally, each team would elaborate its requirement strategy. In the following examples of requirements strategy, the presented techniques are basically extracted from Soren Lauesen book, “Software requirements styles and techniques”[19]and from IREB (International Requirements Engineering Board)[21] in combination with the design thinking [18], we suggested a pipeline of requirements strategy that starts with preparation and ends with requirements management aiming to present a constructive way of handling requirements.
The following requirements strategy is an example for the test company that is open to any changes depending on the project and the estimated time frame. The main goal of this example is to present a basic structure with definitions of concepts and techniques and consider it as the baselines for all teams to learn and practice those techniques aiming to create quality requirements.
The requirements strategy covers five basics: Preparation, Elicitation, Documentation, Verification & Validation, Communication and requirements management.

4.2.2 Requirement Strategy Guidelines

4.2.2.1 Preparation

Intermediate work products are a preparation (a warm-up for the product owner or the application specialist) before starting the elicitation. For instance, the intermediate work products might contain:

- A description of the present work in the domain (how the current feature is working in the domain).
- A list of the present problems in the domain (What is wrong with the current feature).
- A list of the goals and critical issues about the domain
- Ideas for the large-scale structure of the future system
- Realistic possibilities
- Consequences and risks
- Commitment from stakeholders
- Conflict resolution between stakeholders

At the end of this step, the intermediate work products shall be documented.

4.2.2.2 Elicitation

Elicitation is the process of finding and elaborating requirements [19]. Since we wanted to present a constructive step by step, we found that combining the design thinking and elicitation technique could guide and explain where the elicitation technique would be used within which phase of the design thinking. Moreover, the main goal of design thinking is proceeding with the elicitation in

combination with the validation to provide practical and creative resolution of problems and to create solutions that can improve the future results [18]. The following table presents some elicitation techniques based on the design thinking approach.

A. Empathize

Product owners and application specialists develop empathy to understand the stakeholders behind the problem that must be solved [18]. Stakeholders are people that are needed to make sure that the problem is solved.

For instance, Stakeholder might be:

1- The sponsor who pays for the product.
2- Daily users from different departments have to live with the product.
3- Managers of the departments
4- The company's customers (clients of the system)
5- Business partners (suppliers, carriers, and banks)
6- Authorities (safety inspectors, auditors, local government...)
7- IT people...

In- dep stakeholders

Several methods could be used to get to know in close the stakeholders, here below we present some methods that can help to understand the stakeholders:

Method 1: Stakeholder analysis

During the stakeholder's analysis you try to answer the following questions:

- Who are the stakeholders? What are their attitudes and interests?
- What goals do they see for the system?
- Why would they like to contribute?
- What risks and costs do they see?
- What kind of solutions, resources and suppliers do they see?

To get the answers, conduct a meeting with all stakeholders together and discuss the above questions. If the meeting is not as effective as it should be. Call for other meetings for brainstorming or focus groups to discuss the possible solutions. In the case of grouping all stakeholders together again is impossible then try to meet stakeholders one by one.

Method 2: Personas

Create Personas in case the stakeholders are Users and customers to define their story and they need to solve their problem. Personas are fictional characters that have a name and picture; relevant characteristics such as a role, activities, behaviours, and attitudes; and a goal, which is the problem that must be addressed. Personas offer a great way to capture the users and the customers with their needs. Personas are fictional characters that have a name and picture; relevant characteristics such as a role, activities, behaviours, and attitudes; and a goal, which is the problem that must be addressed [36].

Method 3: Focus groups

A Focus group is a kind of brainstorming session where stakeholders meet for one hour to five hours aiming to gather ideas about the current problem or for creating a new product [19]. Groups might be developers, users, suppliers, and marketing...

Here are the steps of how running a focus group:

1- Invite participants: Make sure that all stakeholders are represented.
2- Open the meetings: Present the theme, spend some time letting people get to know each other.
3- Bad experiences: Conduct a discussion around the table about the product or the work domain. Record the issues on a whiteboard. Everyone should get the chance to tell his/her story and express ideas and wishes.

4- Imagine the future: Brainstorming the ideal solutions. Leave people to think as they like and record their ideas. At the end to concretize the ideas, ask: why do you want this? and when would you use it?
5- List the issues: A good focus group session creates around 40 issues that can be ideas, critical problems, requirements and why. The list of issues should be visible for everyone, similar issues should be combined, related issues should be grouped together.
6- Prioritize the issues: Stakeholders should pick up their top ten prioritized issues.
7- Review the lists: Concluding the session with an around-table discussion commenting about the prioritized issues from other stakeholders' groups.
It is recommended to repeat the focus group with different people until the issues seem to be repeated [19].

B. Define

The product owners or application specialists rephrase the problem to get a shared understanding of the details of the problem. Several methods could be used to define the problem with simplicity, here below we present some methods:

Method 1: Task demonstration

Task demonstration is conducting an interview in combination with observations. Basically, you ask a user to show you how he/she performs a specific task, it can also help to observe critical tasks.

Method 2: Scenarios

A scenario is "a case story illustrating one or more tasks or a specific case to be tested" [31].
Vivid scenarios: is a vivid description of experiences that trigger reflection about design research [31] in your text that describes a scenario make sure that the following elements exist: "Actors (who is involved), goals (why), Settings(where and when), objects(what's things are involved), actions(what happens to actors), event(what do actors do), and results(what is the learning achieved)" [31].

Method 3: Epics

From Personas, capture the functionality as epics as high-level stories. Write all the epics necessary to meet the personas goals but keep them sketchy at this stage [36].
Additionally, capture the user interaction and the sequences in which the epics are used and the visual design of your product. Use for instance, workflow diagrams or the presented methods above.

C. Ideate

Product owners and application specialists focus on idea generation aiming to develop as many ideas as possible. To finalize this phase the ideas were selected for prototyping.
The methods of the *Focus group* can be used again in this phase (See instructions under section a-Empathize) also *Brainstorming* and *Design workshop*.

Method 1: Brainstorming

A brainstorming session focuses on innovative ideas to get requirements. Call a group of people(users, customers, students from the university, suppliers, anybody that you feel that can add to your creative session) to meet in a stimulated atmosphere and let them come up with ideas without any barriers. Note down all ideas, even the ones that look ridiculous. It can turn out to be a brilliant idea in the future.

Method 2: Design workshop

A design workshop is a workshop where the users and developers meet and co-operate to design something, for instance, a user interface. (See more details in Soren. L 2002 [19])

D. Prototype

Product owners and application specialists in collaboration with developers create very simple prototypes from the final ideas. Here below some guiding principles for the use of prototyping [20]:

- Understand your stakeholders and purpose.
- Plan a little - prototype the rest
- Set expectations
- If you can't make it, fake it
- Prototype only what you need
- Reduce risk - early and often

In case it is not possible to create a prototype, you can pick up an appropriate tool to create **Mockups** or **wireframes** (Use the tool Balsamiq as expert designers recommended). Otherwise use simply sketching and transfer your design idea onto paper [37]. It is a very fast and efficient way to visualize your results.

1. E. Test

Product owners and application specialists test the prototype with real customers to get feedback. Task demonstration can be used (see Section d-Define) also observation and pilot experiments.

Method 1: Observation

Observation improves your knowledge about work and problems. You can run a usability test using the prototype and observe the user how he /she would perform the prototype without asking any questions. Note down the use case how the user performed the task.

Method 2: Pilot experiments

For instance, you pilot a prototype where a small part of the organization tries the new system with real production data. The users experiment with the changed tasks, and you observe the results and evaluate the costs and benefits of the new system. Pilot experiments help to identify the final requirements and their priorities [19].

4.2.2.3 Documentation

To translate the analysis from the elicitation to a format that architects, developers, and testers would understand, proper documentation is needed. To document the requirement, we are suggesting using the following table to specify first what is the typology of your requirements and then choose the convenient technique to present each requirement. The table presents a few techniques as an example but if you want to access more techniques see Soren Lausen book [19].

- If the analyses were based on the end-user or a stakeholder then you need to specify if the requirements are functional, non-functional or both.
- If the analyses were based on the system (back-end) requirement then you need to specify if the requirements are functional, non-functional or both.
- The analyses were based on both end-user and system requirements then you need to specify if the requirements are functional, non-functional or both.

	Functional requirements	Non-Functional Requirements
Stakeholders' requirements	User stories Feature description Task description Use cases diagram	PLangauge Quality Model Quality Grid
System requirements	Enable story	Maintenance requirements

	Use cases State diagram	Security requirements Usability requirements Performance requirements

Table 7: Requirements summary

Stakeholders & Functional requirements
Several formats can be used to present those requirements, we expose some of them here:
User stories: Break your epics into smaller stories until it looks clear, feasible, and testable and comfortably fit into a sprint. The development team should have a shared understanding of the story.
Instead of the user role, use personas name to connect the story with the relevant persona. As (persona), I want (what?) so that (Why?) [36].
Task description: is a structured text that describes user tasks [19]. A task is how the user and the system interact with each other. See an example from Soren L Book "software requirements styles and techniques "[19] in the appendix (Section 8.5 Task description).
Data expression: 'compact formulas that describe data sequences' [19]. Data expression is short and can clearly show the structure of the data. See Soren L book.

Stakeholders & Non-Functional requirements
Quality grid: Look at all quality factors on your list and assess how is important for the product. Finally, specify concerns for the critical factors (see an example from Soren L book in Appendix Section 8.6).
Quality model: According to the quality factors measuring the quality is possible using a quality model: you start by describing the quality characteristics and sub-characteristics that is relevant for your product and then specify the quality attributes and measure it by using metrics see the example [38] in the appendix (Section 8.7).
Planguage: is a language and a notation for discussing quality factors [39] it contains several concepts (Tag, Gist, Scale, Meter, Must, Plan, Wish, Past) see example from Soren Book in the appendix (section 8.9 Planguage).

System & Functional requirements
Feature requirements: 'a feature is one or more related product functions and data'[19]. The format is to specify the feature in plain text: The product shall record/ show/ compute… [19].
Virtual windows: Simplified screens with graphics and realistic data, but no buttons and menus see example in Appendix Section 8.11 Virtual Windows.
Prototype / Screens: In case a prototype or screens were the results gotten from the elicitation then write a requirement support them for instance:
'R1: The product shall use the screen pictures shown in Appendix. xx.
R2: The menu points and buttons shall work according to the process description in App. yy' [19].
From more functional details use *Complex & simple Functions, Tables & decision tables, Textual process descriptions, activity diagram, sequence diagram, class diagram, State diagram.* See Soren Lauesen: Software Requirements.
Use cases: is the product's part of a task and its intersection with the user (See an example in Section 4.2.3.4.use cases template).

System & Non-Functional requirements
Before documenting the quality requirements, make sure that the quality requirements did not counteract each others [38], for instance:

• Higher performance against lower maintainability
• Higher security against lower usability
Usability requirements: See example from Lauesen book [19] in Appendix (Section 8.9 Usability requirements).
Performance requirement: See an example from the book Software Requirements [19].
See more quality requirements in Soren Lauesen book [19].

4.2.2.4 Verification & Validation

To assure that the created requirement, an assessment need to be done according to the quality criteria of the requirement based on the following properties: Correct, Complete, Unambiguous, Consistent, Ranked for importance and stability, Modifiable, Verifiable, Traceable, and Understandable (See quality requirements criterias in Soren Lauesen book [19]).
- Verify that requirements meet the requirements quality criterias
- All created requirements shall be reviewed by stakeholders (project manager, Application specialist, developers...).
- Book a *review session* with at least one stakeholder.
- *Backlog review* and *Backlog refinement* meetings can be used to check and validate the quality of the requirements.
- Use the *Definition of Ready* as a checklist during the backlog refinements to make sure that the requirements fulfil the quality criteria and they are ready to be implemented. in case it is not ready it should not be included in the sprint.
- All materials from elicitation and final requirements should be documented and attached with the used material (Info, project, analyses…).

4.2.2.5 Communication

Effective and efficient communication is one of the crucial aspects of a successful agile development process. In fact, communication is the key to achieving the organization's objectives and the organization management should establish an adequate communication strategy.
- Provide a multitude of technical communication infrastructure when face-to-face communication is not always possible [21].
- 'Ensure numerous opportunities for face-to-face meetings between geographically distributed teams' [21].
- Co-locate all members of product development if possible.
- Improve communication within and among teams.
- Build a communication structure between all participating stakeholders, for large and complex projects.
- Build trustful collaboration and communication among all relevant stakeholders.
- Access the direct knowledge exchange within and among teams.
- Make sure that all teams use direct communication and fast feedback cycles as a baseline to success.

4.2.2.6 Requirement Management

Requirements management is the process of managing requirements and requirements-based artifacts [41]. It includes documenting, changing, and tracing requirements [42].

Prioritization	**Planning Poker:** Use planning Poker to evaluate and estimate with reference to costs, benefits, or other requirements criteria according to the scheduling of the requirements in releases [41].

Tracing requirements	-Traceability is used to document relationships between artifacts (e.g., requirement is tested by test case or textual requirement is formalized by requirements model). -Traceability is possible within the product backlog (dependencies) and from user stories to the source code [41].
Requirements Change management	-Be ready and prepared for changed and scheduled requirements. -Elaborate an effective change process. And keep in mind that 'The longer a project runs, the greater the probability of changes to your requirements' [40]. -New requirements or changes to requirements are written to the product backlog and considered in the next sprint planning. -Ends up the user story in the wastepaper bin in case it has not yet been realized and replaced with another one and no committee are necessary [41]
Management	The product owner knows the requirements and can elicit them [41]. The product owner can consult multiple additional people; those people must be constantly available and must work actively in the project [41]. "The development team organizes itself and takes responsibility for its own work"[41]. The management supports and helps the product owner and development team to get access to the stakeholders.

Table 8: Requirements management

Since the organization uses Confluence and Jira to document all information needed for the development process, we recommend that the requirement strategy should also be documented for each project and linked to Jira.

4.2.3 Technical Templates

The solution for the presented challenges in Cycle II which are *issue preparation* and *system requirements* was creating technical templates. Based on the knowledge from the requirements strategy, we intended to design four technical templates that covered stakeholders requirements and system requirements (see Section 4.2.2.3 Documentation). The aim is to support and guide the product owner or the application specialist on how to report technical issues and user stories also would have multiple alternatives that allowed them to choose the suitable templates for their needs. Those templates would be integrated into the Jira platform when the reporters wanted to report an issue thus, they would pick up the one that fit the elaborated requirements strategy.

4.2.3.1 Feature Description Template

Based on SAFe (The world's leading framework for business agility) [35], we suggest a *feature description template* to help manage the concept of 'feature'. It simply describes the stakeholders' needs using a short phrase with a name and context [35]. This template is dedicated to technical issues when there is no user, and a clear feature needs to be done. Additionally, the Hypotheses & Future plans are to expose what is the future plan for the feature developed to the development team. Acceptance criteria is conditions that the implemented feature must meet to be accepted by the customer or other systems[35].

The template also contains layers for Non-functional requirements (see Section 4.2.2.3 Documentation), and we thought that all templates shall present clearly what can be done for measuring the quality for any kind of requirements. Test plan and test responsibility is part of the template to make sure that the test strategy for each issue is addressed and it is the responsibility of all development team members. In the end, the Material links shall be provided: The Elicitation process, Stakeholders contact and all-important documentation.

<table>
<tr><td rowspan="2">Feature description:
Create a new microservices shell</td><td>Non- functional requirements</td></tr>
<tr><td>- Create Grafana board with all metrics
-Verify if the memory is enough for creating a new microservice</td></tr>
<tr><td>Hypotheses & Future plans:
The microservice will talk to other microservices
The microservice will talk to SBN</td><td></td></tr>
<tr><td>Acceptance criteria:
The MS returns valid data from the database</td><td></td></tr>
<tr><td>Test plan: Verify the MS is up and running
Test responsible: developer</td><td>Verify all metrics are presented in Grafana board</td></tr>
<tr><td colspan="2">Material links:
The Elicitation process link
Stakeholders contact who requested that requirement
Any documentation needed</td></tr>
</table>

Table 9: Feature description template

4.2.3.2 EARS Template

The basics of this template is inspired by the previous templates only we changed the feature description by using The Easy Approach to Requirements Syntax instead (EARS) [34]. EARS is an approach that provides structured guidance that enables the reporter to write high quality textual [34] requirements. Based on those guidelines we are suggesting a template that addresses different requirements syntax. Depending on the typology of the requirements, Ears presents a multitude of syntax:

- **Generic EARS syntax:** While <optional pre-condition>, when <optional trigger>, the <system name> shall <system response>This syntax produces requirements in a small number of patterns, depending on the clauses that are used.
- **Ubiquitous requirements:** The <system name> shall <system response> (always active)
- **State driven requirements:** While <precondition(s)>, the <system name> shall <system response> (always active unless the while is true).
- **Event driven requirements:** When <trigger>, the <system name> shall <system response> (When specifies how a system must respond)

- **Optional feature requirements:** Where <feature is included>, the <system name> shall <system response> (Where products or systems that include the specified feature).
- **Unwanted behaviour requirements:** If <trigger>, then the <system name> shall <system response> (If and Then specify the required system response to undesired situations).
- **Complex requirements:** While <precondition(s)>, When <trigger>, the <system name> shall <system response> (Requirements that include more than one EARS aim to specify requirements for richer system behaviour)

Those syntaxes might be used for presenting both stakeholders requirements and system requirements, EARS provided different ways of stating the preconditions and it would be a perfect match to present complicated requirements that have several constraints.

EARS: - While <precondition(s)>, When <trigger>, the <system name> shall <system response> - **While** No data is available, **When** the microservice is down, **the** old functionality **shall** replace the microservice functionality.	**Non- functional requirements** -Create Grafana board with all metrics -Verify if the memory is enough for creating a new microservice
Acceptance criteria:	
Test plan: Verify the MS is up and running **Test responsible:** developer	Verify all metrics are presented in Grafana board
Material links: The Elicitation process link Stakeholders contact who requested that requirement Any documentation needed	

Table 10: EARS template

4.2.3.3 User Story Template

Inspired by the previous templates and according to SAFe's [35] a template is created using the user story and enabler story. *The user story* is a format of expressing needed functionality. These kinds of agile requirements focus on the user as the subject of interest, and not the system. The user story format is: As a (user role), I want to (activity), so that (business value). *Enabler Stories* is a supplement to user stories. Some architecture or infrastructure needs to be implemented before implementing user stories.
There are many types of enabler stories, for instance: Refactoring and Spikes, Building or improving development/deployment infrastructure, running jobs that require human interaction

Creating the required product or component configurations for different purposes [35]. The acceptance criteria is very important to be addressed with the user story; it provides the information needed to ensure that the story is implemented correctly [35]. Use the following syntax for writing an acceptance criteria (Given <pre-condition> when <trigger> then <system response>)

User story: As an operator I want to see customer info, so that we can contact the customer in case of an ongoing alarm incident.	**Non- functional requirements** Refactoring requested
Enable story: (system) Create a table called customers in the database	
Acceptance criteria: Given <pre-condition> when <trigger> then <system response>	
Test plan: check the requested data is available **Test responsible:** tester or developer	

Table 11: User story template

4.2.3.4 Use Cases Template

A use case is a written description of how users will perform tasks on a software. It outlines, from a user's point of view, a system's behaviour as it responds to a request. Each use case is represented as a sequence of simple steps, beginning with a user's goal, and ending when that goal is fulfilled [19]. See the use case example below from Oracle home page describing technical requirements [43]. The benefit from using a use case is to provide a list that explains well large projects that need to be presented step by step aiming to give a clear description of what the system does. The use cases help and support establishing the cost and complexity of the system. This template has two different fields than the previous templates, the use cases and precondition. Preconditions are examples of constraints that shall be covered by the implemented use case.

Use cases: Authentication Portal User	**Non- functional requirements**

1. User enters the portal URL. 2. If the customization parameter [remember login] is set, then automatically login the user and provide a session ID. 3. If a first time user, prompt for LDAP user ID and password. 4. User enters the previously assigned user ID and password. 5. Information is passed to the access Manager for validation. 6. If authentication passes, assign session ID and continue. 7. If authentication fails, display error message, return user to login page; decrement remaining attempts; if priest attempts exceed limit, notify user and lock out the account.	Only authenticated users are allowed to gain access to the portal resources. This access restriction applies to all portal resources, including content and services. This portal relies on the user IDs maintained in the corporate LDAP directory.
Precondition: The portal user is an authorized user. Standard corporate LDAP user ID. Must be provided to each employee. Authorized LDAP entry. Every employee has access to the corporate intranet. No guest account.	
Test plan: UX, usability test, test environment, test materials **Test responsible:** Tester	Security

Table 12: Use case template

4.2.4 Traceability

4.2.4.1 Requirements Traceability

“Traceability is crucial for many activities in software and systems engineering including monitoring the development progress and proving compliance with standards"[44]. Software traceability permits to create and link between artifacts, from source artifact to a target artifact. For instance, connect the requirements to its origin and any other artifacts used in the next phases in the software lifecycle [44]. Artifacts can be for instance a requirement, an element in a model, a line of code, an issue in a bug tracking system, or the result of a test case.
Using the traceability approach to connect the test plan (test strategy) with the requirements may build a strong connection between requirements and test strategy during a long period of the software life cycle.
Traceability allows assuring the success of artifacts for example [41]:

- traceability using test coverage analysis: Identify the missing test coverage for requirements

- Traceability using reusability: Identify reusable artifacts
- Traceability using frequency of change: Identify the frequency and the background to changes to requirements
- Traceability using proof of implementation: Prove the implementation of requirements

4.2.4.2 Traceability Tools

The development teams in the case company are aware of the importance of traceability. Therefore, our goal is not explaining what traceability is, in contrast in this section, we are highlighting solutions to trace requirements to the produced artifacts and visualize it to the team by using the test plan in the provided templates. Those solutions are tools that allow tracing and managing the requirements, each tool has its own properties, and some of them can cover major artifacts.

A. Traceability Matrix and Link Graph

Traceability matrix can report traceability between different artifacts such as features, stories, tasks, tests, bugs, and other custom issue types [45]. Users can save different traceability matrices in JIRA and then compare baselines and report the changes between two baselines. Before creating baseline users have to create a traceability baseline filter[45]. See the table below:

		Bug	Improvement	New Feature	Story	Task	
		DEMO-5	DEMO-3	DEMO-2	DEMO-4	DEMO-1	DEMO-6
Bug	DEMO-5			✓*(2)		✓	
Improvement	DEMO-3						
New Feature	DEMO-2	✓*(2)				✓	
Story	DEMO-4						
Task	DEMO-1	✓		✓			
	DEMO-6						

Figure 4: Traceability matrix [45]

- Baselines can be Bug, Improvement, New features, story, and task.
- Each baseline might have several tickets with a unique ID
- Reed mark means that there is a multi-level relation between baseline it be specified with a star and the number of how many relations
- Green mark means one level relation.

B. Eclipse Capra

Capra allows the creation of trace links between arbitrary artefacts in case an adapter for these artefacts is available. A trace link can be created between EMF model elements, source code files supported by the Eclipse Platform (Java, C, Python) [46].
“External artefacts for which the Eclipse Platform does not offer built-in support can also be linked if a fitting adapter is provided. Through its EMF adapter, Capra currently supports elements from UML, SysML, AADL, EAST-ADL, or AUTOSAR models created in, e.g., Eclipse Papyrus, Eclipse EATOP, or ARTOP. The same adapter allows tracing from and to requirements modelled in ProR. Furthermore, adapters for test case executions managed by a continuous integration server

like Hudson or Jenkins can be traced to" [46]. See more details in Capra foundation documentation (https://projects.eclipse.org/proposals/capra).

C. TReqs

TReqs supports agile development of large systems with long lifetimes [47]. It basically allows managing requirements together with changes of code and test, by using a typical Git based infrastructure to peer-review pull or merge requests [47].
T-Reqs is an open source tool that aims to help formulate a dialogue on how to change requirements engineering according to the agile system. In the following rules we recommended TReqs as a practical solution for traceability challenges in large scale agile system development [5]:

- "T-Reqs allows us to manage requirements in exactly the same way as Git. A team pulling the latest changes from the main branch will see conflicts on either of these artefacts as merge conflicts in git.
- Development teams are used to git, thus T-Reqs provide them with a familiar interface to manipulate requirements. T-Reqs-specific conventions, templates and scripts allow generating specific views and reports for non-technical stakeholders.
- Git allows creating branches to experiment with requirements, but also with models and source code. Git merge and Gerrit help to merge branches, without blocking the main database. Merge conflicts will directly relate to requirements conflicts since requirements are stored line-wise.
- Git automatically links changes of code and requirements in commits. T-Reqs adds conventions, templates, and scripts for additional finer-grained tracing. Providing cross-functional teams with feedback based on tracing information can further motivate good trace link quality.
- T-Reqs suggests a suitable review process via Gerrit that has proven to spread information about critical changes effectively between key stakeholders. Ease of use makes it more likely that teams share requirements updates in a timely manner" [5].

4.3 Evaluation (RQ3)

Cycle I: According to the feedback and the evaluation that participants provided, many changes and additions have been done to the requirements strategy to fulfil the participants' needs. Those additions are for instance, the reason behind including many techniques in the requirement strategy was that they wanted several alternatives to choose from also to learn to use different techniques rather than stick to one and use it in different situations. In addition to that, some participants were encouraged that documenting everything in the requirements strategy is what they needed. Moreover, participants asked for specifying who are the people invited to the brainstorming sessions as well as explaining and categorizing what are the user or non-functional requirements. Participants assured that all documentation should be reviewed by another person. Some participants even advised that the organization should support the created requirement strategy and that, by having this requirements strategy, it would allow changing the mindset to focus more on requirements analysis even if there is a lack of stakeholder availability. We interpret this as an indication that the proposed artifact is a step in the right direction. Besides such positive feedback, the evaluation also offered insights in potential next steps. An application specialist for example

explained that he needed to know what to use from elicitation and which suitable format to use to document the requirements. For more details see the checklist forms in Appendix Section 8.11.

Cycle II: In the second round of evaluation, all participants were happy about the implemented technical template and looking forward to using it. According to their feedback, it will solve the frustration that they are encountering while reporting system requirements and complicated projects. Also, it will uncover many unseen executed tasks for instance having the non-functional requirements or an enabler story in the template will allow them to show what has been implemented to the team and to provide better estimations. The most relevant additions were connecting the template with the overall requirements strategy and putting links to the future project. For more details see the checklist forms in Appendix Section 8.11.

Cycle III: The participant agreed on the importance of having a traceability tool that can help them to keep track of the requirements' evolution. He liked the tool T-Reqs and added that both developers and testers shall be involved using the tool.

4.4 Results Summary

In this section, we summarize our results with respect to each research question.

RQ1. *What are the critical challenges with requirements engineering in agile development?*
Several challenges were accumulated based on the data analysis from the workshop and semi-structured interviews. According to the participants' major needs, the challenges were prioritized and the critical challenges with requirements engineering in agile development are presented in Table NNN.
RQ2. *What practices, methods, or tools are potential solutions to critical challenges with requirements engineering in agile development?*
Based on the design science research method, the first solution provided is an artifact in the first cycle for the case company challenges. In the second Cycle a technical template was created as a practice that reflects what has been done in the requirements strategy. In the third cycle, some traceability tools are presented to link the requirement with the code and test to ensure follow up to implemented requirements.
In fact, combining the artifact (requirement strategy), the practice (technical template) and a tool (Traceability tool) and applying them as a package would complete each other. For instance, When the product owner gets the high-level requirements from stakeholders to implement new functionality or new project, the product owner conducts the analysis by following the requirement strategy. The product owner might start with the first step which is *Preparation* with running an intermediate work product to study the present functionality in the domain, what are the critical problems and future goals and what would be the consequences and risks of adding any new functionalities. After the preparation, the product owner must elaborate the *Elicitation* and as we recommended by following the design thinking, The first phase would be *Empathie*, aiming to understand the stakeholders behind the problem by applying some methods, for instance, stakeholders analysis or elaborate personas. Then *Define* the new functionalities using the methods Scenarios and since the personas were already elaborated Epics would be preferable. The next step is *Ideate* where new and several ideas should be created to cover all possibilities, the focus group or brainstorming sessions would help to achieve good ideas. Then *Prototype* the best ideas or provide mockups or wireframes. To close the elicitation all prototypes or mockups should be tested with real users using methods observation or Pilot experiments. To document the results from the elicitation, the product owner should categorize the requirements (see Section 4.2.2.3 Documentation) if they are stakeholders requirements or system requirements and which format is

applicable on both according to the requirements strategy. The methods that have been used in elicitation might guide to choose the efficient format that represents the requirements category. For instance, if the personas and epics were done in elicitation then documenting the user story would be convenient. and if developing this user story requires enabling many other system requirements then the product owner shall use the user story template that covers both user stories and enabler stories. In fact, if the elicitation is done properly for those new functionalities, the documentation and filling the templates would be as a conclusion that summarized all the analysis conducted in elicitation. furthermore, when the template was filled and the requirements were ready for review thus the validation and verification shall be done whether review them in the backlog review using the definition of ready or another product owner verify the created requirements (see Section 4.2.2.3 Verification & Validation). After the review was done thus the development team shall prioritize the requirements using the Planning Poker and decide on which one should be in the sprint and ones that should wait until other teams or stakeholders were done with the requirements (see section 4.2.2.5 Requirement Management). In case the stakeholders want to change the requirement, thus the changes should be reported and considered in the next sprint. Moreover to create a follow up to the requirements changeability for the new functionality a trace link will be created between the user stories and the test cases. By using T-Reqs, the development team is able to track the requirements changeability and check if the code and the test cases are covering those changes. Hence, a proper established requirements strategy would be reflected on the template, also creating quality requirements using the requirements strategy would be helpful to trace the requirements with the test which can be stated in the test plan in the template.
Table NNN presents the summary of the thesis results, each cycle with its challenges, Practices/Methods/Tools(artifact), and evaluation:

Cycles	RQ1: Challenges	RQ2: Practices/Methods/ Tools (the Artifact)	RQ3: Evaluation
Cycle I	Elicitation & Validation Communication strategy Documentation Requirement managements	Requirements strategy	Verified
Cycle II	System requirements Issues preparation	Technical templates	Verified
Cycle III	Test strategy	Traceability	Verified

Table 13: Results summary

RQ3. *What critical challenges with requirements engineering in agile development can be addressed by specific practices, methods, or tools extending those practices, methods, or tools?*
The evaluation ensures the significance of the artifact, the practice and the tool. It also provided feedback that helped to improve all solutions. All participants in the evaluation showed their interest and looked forward to using the artifact in their context.

Some of the challenges that were highlighted in grey in Table 4, 5 and 6 might be already solved

by the provided artifact. Some challenges are related to agile development and to scrum. The way of working is triggering confusion between the development teams, Product owners and stakeholders. This problem can be another topic for a thesis. Probably a new ideology of bracing a hybrid way of working and how we can change the mindset of sticking with one way of working and apply it on all projects regardless of its type.

5 Discussion

5.1 Discussion

The artifact of this thesis packages a set of tools, practices and templates that help an organization to handle the requirement strategically from start to end. It places particular emphasis on continuous feedback and following up requirements-related issues. Within a case study, the artifact guides teams to define a requirements strategy for agile development and facilitates to involve all team members in its practices and tools. The quality of processing creating requirements using the requirement strategy will be reflected on the chosen templates. Finally, when a traceability tool is used and a permanent link created between requirements and the implemented code or with test cases, the continuation of changing or adding requirements will be consistent.

Since companies aim to achieve the market position, therefore implementing the time to market strategy for tracking the product development and working on different types of projects may affect the quality of the requirements. For instance, handling system requirements in the same way as the stakeholders requirement is not sufficient and it propagates confusion between developers when they want to report a complex functional or non-functional requirement.
System requirements are the major challenge in agile development [3], and the only proposed solution is to use traditional requirement engineering to present functional and non-functional requirements. For a complex long project, it is preferable to be analysed by requirement engineering techniques and presented with use cases, feature description using EARS (see section 4.5.2 EARS Template) or any suitable format from requirement engineering that would translate the complex project to smoothly simple described tasks. Without a mechanism such as the requirements strategy proposed in this thesis, it is unclear how these can be fitted into agile development.

The proposed requirements strategy covers both system requirements and stakeholders requirements and enables teams to keep them consistent. For example, in case the product owner wants to report a user story, the requirement strategy guides the creation of personas first, to facilitate understanding the users and all the involved stakeholders. Then, a focus groups or a brainstorming session could be conducted to analyse the user needs and generate epics (high level requirements). Finally, descriptive user stories will be the result of systematically refining these epics.

The requirements strategy and the provided templates are validated by the participants from the company, and they contributed to improve it. While specific to the concrete case context, the artifact can potentially provide value and inspiration to any company who is facing similar challenges. It was formulated with the need of adjusting it to concrete development domains in mind. Based on the results of this thesis, any agile team can establish their own requirement strategy, templates and traceability tools utilizing the provided baselines.

The contributions of this design science study are mechanisms to better integrate traditional requirement engineering practices into agile methodologies in order to solve real world challenges. Both approaches have similar principal goals for instance ‘satisfy the customer needs’ (see section

2.2 Requirement engineering & Agile Methods), additionally, the mixture between both approaches may solve other untreated challenges.
The mindset of the agile community that is against the requirement engineering probably needs to change and rethink on the general benefits of applying what is needed from the requirements engineering approach. In fact, the race toward agile transition in companies is beneficial especially when the company wants to provide market innovation. However, large companies should be aware that there is a significant correlation between the type of project and the suitable way of working on it. For instance, for large-scale system developments, it is hard to use agile methodologies; moreover, hardware and mechanics cannot be developed agile [49].

Agile is not a fashion that an organization should stick to. In contrast, waterfall and plan driven are perfect for large scale development [21]. incubating a hybrid process with elements from all methodologies depending on the type of project, that may solve many challenges and frustrations. Using elements from requirement engineering is proven in this research and this is a convenient solution for the organization otherwise tailoring various methodologies could play a crucial role regarding involving requirements engineering in the process of developments. This thesis is a good start-up for other future studies that may conduct an empirical assessment on companies that incubate hybrid processes using requirements engineering. It also aims to encourage companies to use what is useful for their projects and their development teams and to integrate it into their overall ways of working.

5.2 Validity threats

The presented validity threats are constructed as Runeson recommended [24], and it includes: internal validity, construct validity, external validity, and reliability.

Internal validity

The goal of internal validity is to reveal factors that affect the relationship between variables, factors investigated and results. In a qualitative study, this relates to the degree to which trustworthy cause-effect relationships can be established and the degree to which a study allows to eliminate alternative explanations. One potential threat is therefore misinterpretation of interview questions or answers. Nowadays companies are struggling with the new mode of working remotely because of “Covid-19”. Thus, it was not possible to interview employees face to face and all interviews were remotely conducted. Since the topic was discussed with the supervisor at the end of the year 2019. The workshop was conducted in February 2020, a few weeks before the company decided to work remotely
because of the pandemic. This delay might also have affected internal validity. To mitigate these threats and to make sure that the data obtained from the workshops is still up to date, workshop participants were asked in January 2021 (start of thesis) to review the data to see if it is still applicable during the remote way of working. Further, the thesis author is familiar with Verisure (the case company), where she had worked as a software test engineer from 12/2018 until 12/2020. This thesis's general theme and objectives were conceived during this work, which ensures that questions and artifacts are relevant to the case company.

Construct validity

Runeson defined the construct validity to what extent the measures that are studied represent what the researcher interprets and what is investigated according to research questions [49]. Since the participants in this research were colleagues to the author, the expression “you know already about

it” was said many times from different participants even though we stated in the beginning of all interviews that they should forget about the author's engagement with the company. This threat was validated by the way of treating data iteratively using design science research. Accordingly, working within cycles with getting feedback about the solution in evaluation sessions allows us to ensure the quality of the data collected. The more we move from one cycle to another, the more significant data was collected every cycle.

External validity

External validity relates to the extent to which the findings can be generalized [49]. The generated artifact is the result of a structured investigation process for revealing the challenges of RE in agile methods in depth also enabling both concepts through one set called Requirement strategy. The requirements strategy and rest of results are applicable in the case company, and it can likely be generalized to any similar company that works on large-scale software development, but further research may be needed to verify this.

Reliability

Reliability is the threat where the produced solution is the same as other researchers or repeating the same study again [49]. In a qualitative study, some subjectivity of the researcher cannot be avoided and therefore the recoverability becomes important, i.e., the potential that another researcher, who finds different results can find an explanation for these differences. We ensured the reliability partially before starting to work on the thesis. This research is a continuation of recent research that recommended in a few papers a study is needed to involve RE among agile development [3]. Well known challenges from these papers were also spotted when the author was working with the case company and that was reflected on the degree of awareness in interview questions presented in the appendix. Additionally, several discussions and a workshop were conducted with the supervisor that allows covering all related works and literature. In addition to basing the study well within the established research, execution and data extraction is discussed in detail to allow for replication and recoverability by other researchers.

Conclusion validity

Conclusion validity measures if the researcher establishes correct relationships from the findings. Several challenges from the workshop were presented but some of those challenges were excluded. The reason behind that was to focus on depth and provide concrete solutions that are valuable for the case company rather than spend time on all challenges and provide weak solutions. Due to the company's interest, repeated evaluation in the research cycles allowed us to verify that relationships indeed are valid in the case context. Therefore, the interviews and the supervision play an important role in guiding and prioritizing the challenges to get in the end relevant results.

6 Conclusion

Based on the usage of design science research, a new artifact is created for industry and research. In the first cycle an artifact was created, we called it a requirement strategy. This outline covers all strategies that allow product owners, and the development teams with support from the organization to set up a skeleton of techniques, tools and rules that help to enhance the analyses and elaborate high quality requirements.
The second cycle result is a supplement for the basic artifact (requirement strategy), it is called a technical template. Basically, four technical templates guide and help report all various degrees of requirements complexity. It covers presenting the system and stakeholders requirements related to what can be done for non-functional requirements, also a test plan referring to the test strategy and a link to the elicitation conducted with sharing the possible future project.
To ensure the follow up of the created and changed requirement in the third cycle another solution was provided as an extension to the basic artifact. The solution is using traceability tools to keep an eye on the changeability of the requirement or the code. We recommended a powerful open source tool (such as T-Reqs [5]) that links system requirements implemented with test cases and any changes that occurred to the code the test cases failed and it spotted any requirement that missed test cases. T-Reqs has many advantages regarding traceability for system requirements (see Section C-TReqs).

As it said before the solution might be generalized to companies that are developing large scale software systems. The thesis is encouraging the industry to use requirement engineering even if they are using agile methodologies, the combination of both aspects works fine and might resolve many challenges as the ones that were accumulated in this study. For the research community, this study is a motivation for diving more on agile RE and presents several solutions for the industry challenges. For instance, conducting a quantitative study with an organization that would apply the implemented artifact can be interesting to measure the efficiency of the solution provided.

7 References

[1] Agile manifesto link <https://agilemanifesto.org/principles.html>
[2] Fredrik H, Master's thesis in Software Engineering "Impediments Associated with Requirements in Agile Projects" ,Chalmer university 2018.
[3] Rashidah K, Grischa L, Eric K, Swathi G, Benjamin K, "Requirements Engineering Challenges in Large-Scale Agile System Development" ,2017 IEEE 25th International Requirements Engineering Conference (RE), DOI:10.1109/RE.2017.60
[4] Rebekka W, Patrizio P, Eric K, Mats L, "Boundary Objects and their Use in Agile Systems Engineering", journal of Software: Evolution and Process,2047-7481 (ISSN) Chalmers & University of Gothenburg 2017,DOI:10.1002/smr.2166.
[5] Rashidah K, Grischa L, Eric K, Swathi G, Benjamin K, "T-Reqs: Tool Support for Managing Requirements in Large-Scale Agile System Development", Chalmers & University of Gothenburg 2018,26th IEEE International Requirements Engineering Conference, RE 2018,Banff, Canada, 2018-08-20 - 2018-08-24,DOI 10.1109/RE.2018.00073.
[6] Eric K, course Requirement engineering 2019, Lecture 1. Chalmers & University of Gothenburg 2019.
[7] B. Ramesh, L. Cao, R. Baskerville, Agile requirements engineering practices and challenges: an empirical study, Information Systems Journal 20 (5) (2010) 449–480. doi:10.1111/j.1365-2575.2007.00259.x.
[8] Heikkila, V. T., Damian, D., Lassenius, C., and Paasivaara, M. A mapping study on requirements engineering in agile software development". In: 41st Euromicro Conference on Software Engineering and Advanced Applications, 2015, pp. 199–207. Doi: 10.1109/SEAA.2015.70.
[9] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, S. Shamshirband, A systematic literature review on agile requirements engineering practices and challenges, Computers in Human Behavior 51 (2015) 915 – 929, computing for Human Learning, Behaviour and Collaboration in the Social and Mobile Networks Era. Doi:https://doi.org/10.1016/j.chb.2014.10.046.
[10] Zhu, Y. . Requirements engineering in an agile environment. Master thesis (2009), Uppsala University.URN: urn:nbn:se:uu:diva-108027.
[11] Cao, L. C. L., & Ramesh, B. (2008). Agile requirements engineering practices: An empirical study. IEEE Software, 25(1), 60–67.
[13] Alan R. Hevner, Salvatore T. March, Jinsoo Park and Sudha Ram 'Design Science Research in Information Systems',2004 ,(pp.75-105). https://doi.org/10.2307/25148625.
[14] E. Bjarnason, K. Wnuk, B. Regnell, A case study on benefits and side-effects of agile practices in large-scale requirements engineering, in: Proceedings of the 1st Workshop on Agile Requirements Engineering, AREW '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 1–5. doi:10.1145/2068783.2068786.
[15] Hevner, Alan, and Samir Chatterjee. "Design science research in information systems." Design research in information systems. Springer, Boston, MA, 2010. 9-22.
[16] Marcelo M, Thiago A,Edson O,and Pedro L "Considerations about the efficiency and sufficiency of the utilization of the Scrum methodology: A survey for analyzing results for development teams" Science Direct, November 2020.
[17] M.G. Software, SCRUM Overview for Software Development, link <https://www.mountaingoatsoftware.com/agile/SCRUM/overview >
[18]IREB Certified Professional for Requirements Engineering , September 24, 2020 link<https://www.ireb.org/en/downloads/#cpre-foundation-level-syllabus-3-0 >
[19] Soren L, "Software requirements styles and techniques", Great Britain 2002.

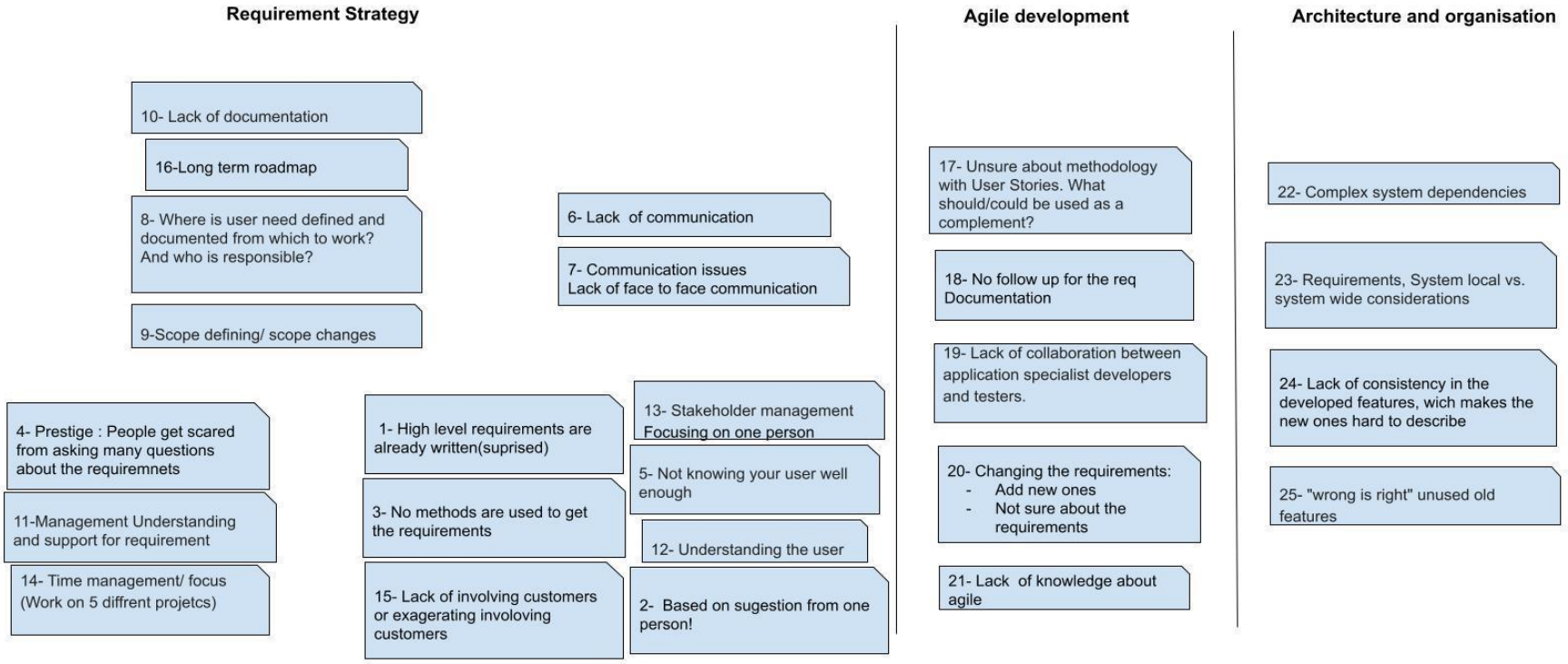
[20] T. Warfel: Prototyping - A Practitioner's Guide. Rosenfeld Media, 2009.
[21]IREB Certified Professional for Requirements Engineering RE@Agile Primer,September 24, 2020 link <https://www.ireb.org/en/downloads/#cpre-foundation-level-syllabus-3-0 >
[22] Merriam-Webster Dictionary and Thesaurus, 2016. Available at: <https://www.merriam-webster.com/dictionary/workshop> (Access 2021).
[23]Rikke O, Karin T, "Workshops as a research methodology" ,April 2017,Electronic Journal of e-Learning 15(1):70-81.
[24] P. Runeson, M. Höst, "Guidelines for conducting and reporting case study research in software engineering," Empirical Softw. Engg., vol. 14, pp. 131-164, 2009.
[25] Rossman, G., & Rallis, S. F. (2012). Learning in the field: An introduction to qualitative research (3rd ed.). Thousand Oaks, CA: Sage.
[26] John W.C , Research Design: Qualitative, Quantitative, and Mixed Methods Approaches, Edition 2014
[27] Saldaña, J. The coding manual for qualitative researchers.Sage Publications Ltd (2009)
[28]Saldaña, J. The coding manual for qualitative researchers. Sage Publications Ltd (2015)
[29] ESKOLTA. Qualitative Data Analysis, 2013
<http://www.eskolta.org/researchfundamentals/pdf/4A.QualitativeDataAnalysis.pdf>
[30] Iv´an .G, Rub´en .FJorge J. G. Guideline for the definition of EMF metamodels using an Entity-Relationship approach.Universidad Complutense, 2017,pp1217–1230
https://doi.org/10.1016/j.infsof.2009.02.003
[31]Carroll.J.M. Five reasons for scenarios-based design: 32nd hawaii international conference on system sciences (1999), DOI: 10.1109/HICSS.1999.772890.
[32] Salma Patel, Rebecca Cain, Kevin Neailey, and Lucy Hooberman. Exploring Patients' Views Toward Giving Web-Based Feedback and Ratings to General Practitioners in England: A Qualitative Descriptive Study. Journal of Medical Internet Research 18, (2016), doi:10.2196/jmir.5865
[33] Ann M. Hickey and Alan M. Davis. 2003. Elicitation technique selection: How do experts do it? In Proceedings of the 11th IEEE International Requirements Engineering Conference (IREC 2003), 2003.1232748
[34]Alistair M,The Easy Approach to Requirements Syntax (EARS),2019
[35]SAFE. Story. <https://www.scaledagileframework.com/story/> 2021
[36] Roman P link <https://www.romanpichler.com/blog/personas-epics-user-stories/>
Pichler Consulting
[37] Olga S ,"What is sketching and the 4 most popular types of it", link
<https://schoolofsketching.com/blog-in-english/sketchingwhatisit>, March 2021
[38] Eric K, course Requirement engineering 2019, Lecture 4 & lecture 5, Chalmers & University of Gothenburg 2020
[39]Glib,T, A Handbook for Systems Engineering Requirements Engineering, and Software Engineering Using Planguage, UK 2005.
[40] K. Wiegers and J. Beatty: Software Requirements. 3rd Edition. Microsoft Press, 2013

[41] IREB handbook for requirements management advanced level en v1.1.
[42] Martin G: A Glossary of Requirements Engineering Terminology. Version 1. May 2014.
[43] Oracle, the link<https://docs.oracle.com/cd/E19636-01/819-2582/auto30/index.html>
[44] Rebekka W,Eric K , Jan-P S , Salome M, Anthony A, Patrizio P , "Collaborative traceability management: a multiple case study from the perspectives of organization, process, and culture" ,November 2018.
[45] Traceability Matrix and Link Graph, link

<https://marketplace.atlassian.com/apps/1211580/traceability-matrix-and-link-graph?hosting=server&tab=overview>
[46] Capra, link<https://projects.eclipse.org/proposals/capra>
[47] Treqs repository, link < https://gitlab.com/treqs-on-git/treqs-ng/-/tree/master >
[48] Miles MB ,Huberman A M,Saldaña J , "Qualitative data analysis, a method source book", ch 4 , edition 4, 2020
[49] Alan M. D, Didar Z, "Good requirements practices are neither necessary nor sufficient". Requirements Engineering 11(1):1- .DOI: 10.1007/s00766-004-0206-4
March 2006.
[50] F. Paetsch, A. Eberlein, F. Maurer, Requirements engineering and agile software development., in: WET ICE 2003. Proceedings. 12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises,2003, pp. 308–313. doi:10.1109/ENABL.2003.1231428.
[51] Lim E., Taksande N., Seaman C., 2012. A balancing act: what software practitioners have to say about technical debt. IEEE Software, 22–28.
[52] M. Fowler, "TechnicalDebt" , May 2019, link:
<https://martinfowler.com/bliki/TechnicalDebt.html >
[53] Zhi J, "Environment Modeling-Based Requirements Engineering for Software Intensive Systems", 2018

8 Appendix

8.1 Card sorting (Workshop)



8.2 Interview question (Product owner & application specialist)

1- For how many years have you been working at Verisure? as a PO?
2- Do you have any guides or template that help you to write Jira issues?
PO and APP specialist do have sessions or a leader who can coach them?
3- how are you experiencing working with different dealers: it can be different culture, requirements
4- How do you get the request that this requirement must be in the backlog review?
5- How much does the project roadmap control the velocity of the sprint?
6- In case you need to understand the high-level requirement from the business or stakeholder, what kind of meetings or techniques do you use to get the answers or the information that you need?
7- Would a developer or a tester attend those meetings?
2- What techniques do you use to write the user stories?

Change the req
From agile Manifesto: "Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage."
8- Why do you change the requirements? Who requests that?
9- How many times do you change the requirement during the sprint?
10- How often does the dev team ask for changing the requirements?
11- Why would you **add** a new story to the sprint?
12- Would you change the requirements when the development and test are done?
why...
Template
13- What is your input about the Jira template for writing user stories? is it enough?
14- Would you like to have a better template that can help you to write user stories?
15- if testers and developers would like to have better templates or tools to write the user stories would you agree with that.
16- In the case of technical documentation, what are the challenges that you face writing technical requirements?
17- How do you define the acceptance criteria?
18- What do you think about the estimation from the dev team
19- How the PO use the estimation

8.3 Interview questions (software architect & developers)

1- How many years have you worked as a developer?
2- For how many years have you been working in Verisure?
3- How can you describe your role?
4- Do you have any specific tasks or features related to you in your team? Project
5- PO told me that she comes up with solutions and you validate them, how do you verify the solutions? Daniel
6- If you come up with solutions, who validates them?
7- Before you come up with solutions, what do you do? What architecture methodologies do you use?
8- How are you doing with scrum?

9- Would you like to use another methodology rather than Scrum
10- What is wrong with scrum?
11- What do you think about the backlog refinement: Do you prepare for the issues? Do 12- you get a clear picture of what to do or things change when you start working on it?
13- When you come up with a solution, do you communicate it with other developers?
14- What is your impression of code review? Does that help you to improve the quality of code?
15- You are the microservices team, how much does the user stories template serve your type of features?
16- What do you think is wrong with that user story template?
17- Would you like to have a template for technical issues?
18- What are the challenges that you face when you are reporting an issue?
19- Do you report non-functional requirements? Performance, Maintenance...?
20- Is it common that non -functional requirements are part of any technical issue?
21- Would you like to have a non-functional req story written separately using a specific template for that?
22- How can you describe Testing in your team?
23- what can be done better?
24- What do you think about test strategy, test policy...

8.4 Interview questions (Testers)

1- How many years have you worked as a Tester?
2- For how many years have you been working with Verisure?
3- How do you describe your role?
4- Do you have any specific tasks or features related to you in your team? project
8- How are you doing with scrum?
9- Would you like to use another methodology rather than Scrum
10- What ‘s wrong with scrum?
11- What do you think about the backlog refinement: Do you prepare for the issues?
12- Do you get a clear picture of what to test?
13- When you pick up an issue, how do you plan your testing? Do you communicate the requirements with PO and developers before starting the testing?
14- What ‘s methods do you use for testing? activities?
15- You test the microservices and backend issues, how much does the user stories template serve you to do testing?
16- What do you think is wrong with that user story template?
17- Would you like to have a template for technical issues?
18- What are the challenges that you are facing doing the test?
19- Do you report non-functional requirements? Performance, Maintenance...?
20- Is it common that non -functional requirements are part of any technical issue?
21- Would you like to have a non-functional req story written separately using a specific template for that?
22- How can you describe Testing in your team?
23- What can be done better in testing?
24- Is there any follow up for test strategy, test policy… in your team

8.5 Task description

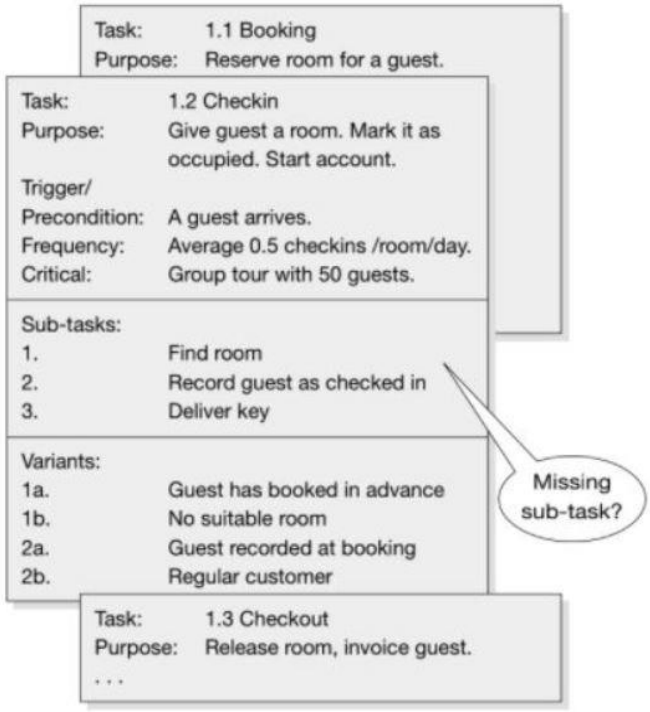


Figure 5: Task description example [19]

Purpose: The purpose in the example of check-in is to give the guest a room, mark it as occupied,and start the accounting for the stay.

Trigger/Precondition A trigger declares when the task starts, for instance the trigger that initiates the event in the example is the arrival of a guest. A precondition must be fulfilled before the user can carry out the task. In the example only the trigger was specified and not a precondition.

Frequency and Critical The fields frequency and critical are useful in practice. The presented requirement in the example supports frequency 0.5 check-ins per room per day, and support critical activities with 50 guests arriving. Thus, those lines can be considered as design constraints for these requirements.

Sub-tasks. The receptionist must find a suitable room for the guest, record guest data, . Finally deliver the room key to the guest. These sub-tasks are on the domain level. They specify what the user and the computer must do together.

Variants. For instance, Sub-task 1 (find room) has two variants: (1a) The guest may have booked in advance, so a room is already assigned to him. (1b) There is no suitable room (suggests some communication between receptionists and guests about what is available). **Variants** are very recommended for customers as well as for developers. There is no need to describe rules or specify logic for the cases; only list the variants [19].

8.6 quality grid

Quality factors for Hotel system	Critical	Impor-tant	As usual	Unim-portant	Ignore
Operation					
Integrity/security			X		
Correctness			X		
Reliability/availab.		1			
Usability		2			
Efficiency			X		
Revision					
Maintainability			X		
Testability			X		
Flexibility			X		
Transition					
Portability					X
Interoperability	3			4	
Reusability					X
Installability		5			

Concerns:

1. Hard to run the hotel if system is down. Checking in guests is impossible since room status is not visible.
2. We aim at small hotels too. They have less qualified staff.
3. Customers have many kinds of account systems. They prioritize smooth integration with what they have.
4. Integration with spreadsheet etc. unimportant. Built-in statistics suffice.
5. Must be much easier than present system. Staff in small hotels should ideally do it themselves.

Figure 6: Quality grid

8.7 Quality model

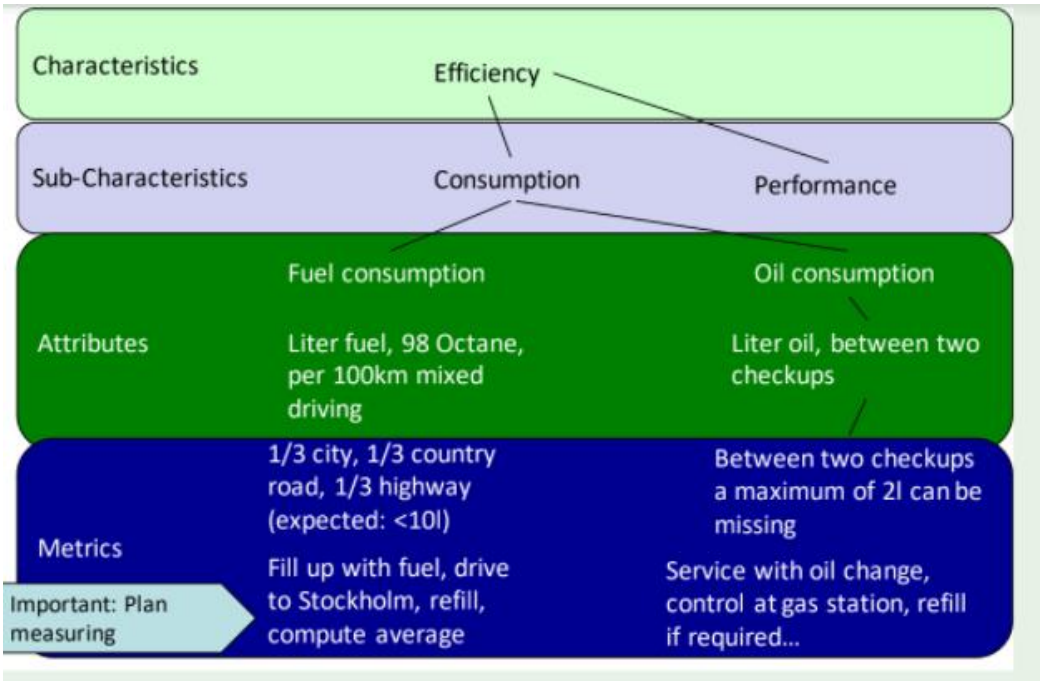


Figure 7: Quality model

8.8 Planguage

Forecast speed [Tag]: How quickly the system completes a forecast report [Gist]

Scale: average number of seconds from pushing button, to report appearing.

Meter: Measured 10 times by a stopwatch during busy hours in hotel reception.

Must: 8 minutes, because the competitive system does it this fast.

Plan: ____ (supplier, please specify).

Wish: 2 minutes.

Past: Done as batch job taking about an hour.

Figure 8: Planguage

2.

8.9 Usability requirements
3.

Usability requirements?

R1: System shall be easy to use??

R2: 4 out of 5 new users can book a guest in 5 minutes, check in in 10 minutes, . . . *New user* means . . . Training . . .

Achieving usability

- Prototypes (mockups) before programming.
- Usability test the prototype.
- Redesign or revise the prototype.

Easier programming. High customer satisfaction.

Defect types

Program error: Not as intended by the programmer.

Missing functionality: Unsupported task or variant.

Usability problem: User cannot figure out . . .

Usability problem:

Examples of usability problems

P1: User takes long time to start search. Doesn't notice "Use F10". Tries many other ways first.

P2: Believes task completed and result saved. Should have used *Update* before closing.

P3: Cannot figure out which discount code to give customer. Knows which field to use.

P4: Crazy to go through 6 screens to fill 10 fields.

Problem classification

Task failure: Task not completed - or believes it is completed.

Critical problem: Task failure or complaints that it is cumbersome.

Medium problem: Finds out solution after lengthy attempts.

Minor problem: Finds out solution after short attempts

Figure 9: Usability requirements

8.10 Virtual windows

R1: The product shall store data corresponding to the following virtual windows:

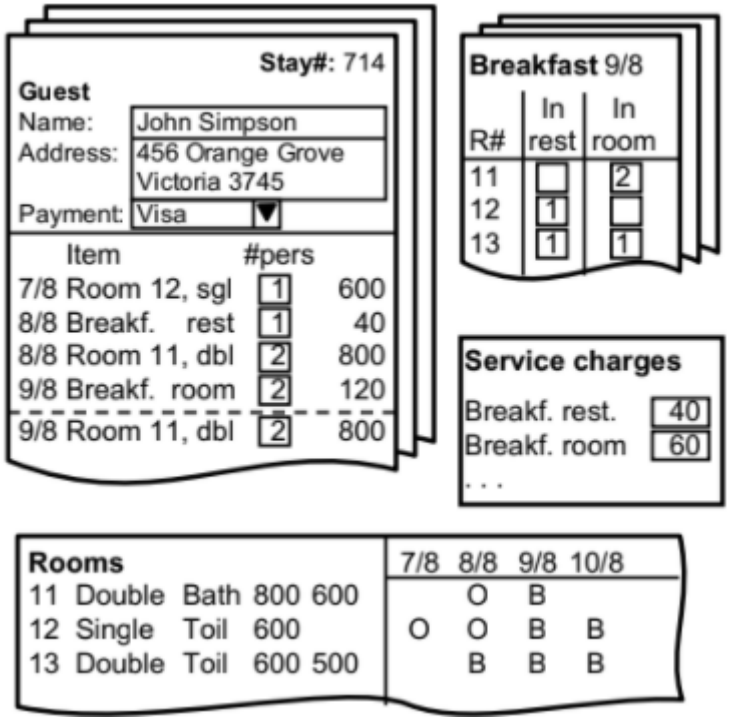


Figure 10: Virtual windows

8.11 Checklist forms

8.11.1 Checklist 1

Project: Requirement strategy		Date:050521 Who: Scrum master & Senior tester
Content check	Observations - Found and missing	Problem?
1- Intermediate work products	ok	How long would this take ? Risk of having stakeholders commitment Use to commit to late to the process
2- Elicitation using Design Thinking		
2-1 Empathize	ok	

Method 1: Stakeholder analysis	ok	
Method 2: Focus group	ok	
2-2 Define		
Method 1: Task demonstration	ok	
Method 2: Scenarios	ok	More methods
2-3 Ideate	ok	
Method 1: Focus group		
Method 2: Brainstorming	ok	Who are the invited people
Method 3: Design workshop	ok	
2-4 Prototype	ok	
Method 1: sketching		
2-5 Test		
Method 1: Pilot experiments		
Method 2:Observation		
3- Documentation	ok	Categorize which is a user story or functional requirement and NFR
3-1 User stories		
3-2 Functional requirement		
3-3 Non-functional requirement		
4- Verification & Validation	ok	All documentation should be verified
5- Communication	ok	Invaluable meeting and communications
6- Requirement management	ok	

Project: Template		Date: 050521 Who: Scrum master & Senior tester
Content check	Obeservations- Found and missing	Problem?
Template 1	ok	
Template 2	ok	
Template 3	ok	
Template 4	ok	

8.11.2 Checklist 2

Project: Requirement strategy		Date:140521 Who: Scrum master & Senior developer
Content check	Observations - Found and missing	Problem?
1- Intermediate work products	ok	No
2- Elicitation using Design Thinking	ok	
2-1 Empathize		
Method 1: Stakeholder analysis	Ok	In-dept It people
Method 2: Focus group	OK	Change the Mindset to focus more on rep analysis Lack of stakeholders availabilities
2-2 Define	ok	

Method 1: Task demonstration	ok	References
Method 2: Scenarios	ok	
2-3 Ideate		System requirement
Method 1: Focus group	ok	
Method 2: Brainstorming	ok	
Method 3: Design workshop	ok	Architecture
2-4 Prototype		
Method 1: sketching	ok	
2-5 Test		
Method 1: Pilot experiments	ok	
Method 2:Observation	ok	
3- Documentation	ok	
3-1 User stories		
3-2 Functional requirement		
3-3 Non-functional requirement		
4- Verification & Validation	ok	No real time given for analyses
5- Communication	ok	
6- Requirement management	ok	Convince the management how we can do that

Project: Template	Date: 050521 Who: Scrum master & Senior developer

Content check	Obeservations- Found and missing	Problem?
Template 1	Add links for the future project also to analyses that have been done	
Template 2	Add links for the future project also to analyses that have been done	
Template 3	ok	
Template 4	ok	

8.11.3 Checklist 3

Project: Requirement strategy		Date:140521 Who: Application specialist
Content check	Observations - Found and missing	Problem?
1- Intermediate work products	OK	Time frame
2- Elicitation using Design Thinking		Stress to deliver No time to communicate Lack of resources
2-1 Empathize		
Method 1: Stakeholder analysis	ok	
Method 2: Focus group	ok	
2-2 Define		
Method 1: Task demonstration	ok	
Method 2: Scenarios	ok	

2-3 Ideate		
Method 1: Focus group	ok	
Method 2: Brainstorming	ok	
Method 3: Design workshop	ok	
2-4 Prototype		
Method 1: sketching	ok	
Method 2: Mockups or wireframes	ok	
2-5 Test		
Method 1: Pilot experiments	ok	
Method 2:Observation	ok	
3- Documentation	ok	Need to know what to use from elicitation and which format to use to document
3-1 User stories		
3-2 Functional requirement		
3-3 Non-functional requirement		
4- Verification & Validation	ok	Lack of resources
5- Communication	ok	
6- Requirement management	ok	Need support from management

Project: Template		Date: 050521 Who: Application specialist
Content check	Obeservations- Found and missing	Problem?

Template 1	ok	
Template 2	ok	
Template 3	ok	
Template 4	ok	