Universiteit Leiden

ICT in Business and the Public Sector

Lightweight Requirements Engineering Metrics

Designing and evaluating requirements engineering metrics for software development in complex IT environments – Case study

Name: Jenny Yung
Student-no: s0952249

Date: 18/04/2019

1st supervisor: Werner Heijstek

2nd supervisor: Guus Ramackers

Abstract

The software development industry is relatively young and has matured considerably in the past 15 years. With several major transformations, the industry continues to seek ways to improve their delivery. A while ago, the waterfall method was the standard option for software development projects. Today, agile methodology is used more frequently. New development engineering practices are developed like standardization, continuous integration or continuous delivery. However, the consensus of being predictable and repeatable is not yet there. Agile software development and delivery are still not sufficiently reliable, resulting in expensive and unpredictable projects.

In this thesis, *measurements* in requirements engineering are introduced. This paper presents a maturity model to assess the requirements engineering in complex IT environments. Companies strive to improve their software production and delivery, and the study aims to find a lightweight requirements metric method. The method is compared with a library of abstract metric models. The case study is performed at three different companies, and experts evaluated the conceptual metric system. As a result, a Requirements Engineering Maturity scan (REMS) is created. The tool suggests appropriate generic requirements metrics to deal with the encountered project. The paper concludes a review with related work and a discussion of the prospects for REMS.

Keywords: Requirements Engineering • Quality Measurements • Metrics • ISO/IEC 9126 • Agile Methodology • Requirements Engineering Maturity Scan • Complex IT Infrastructure •

Acknowledgement

Anyone writing a thesis will tell you it is a lot of work, and they are right. It has been a rewarding journey where I learned a lot. The first contact with "Lightweight Requirements Engineering Metrics" started as a business case at Nederlandse Spoorwegen. As time passed, I was lucky enough to develop and implement a lightweight metrics system at a consultancy company and an international bank. I had the chance to put the theory into practice. It was great to see the conceptual metrics model in a corporate company.

There are not enough words to express the gratitude I feel to the people that have helped, inspired and supported me in writing this thesis.

First, I would like to thank my first supervisor, Werner Heijstek, and second supervisor, Guus Ramackers. In addition, I would like to thank the manager of NS, for giving me advice and input during, and even after the business case. Also, I would like to thank the consultancy company and international bank for giving me the chance to work with this delightful topic.

Finally, I would like to thank my parents, Chi Lung, Ka Ho, and all the participants who helped to review and kept me motivated for writing the thesis.

[Page intentionally left blank]

There's a saying in Dutch:

"Meten is weten,
gissen is missen"

Meaning: "measurement is the key to knowledge"

Table of content

Abbreviations

ASD	Adaptive Software Development
AUP	Agile Unified Process
CMMI	Capability Maturity Model Integration
DSDM	Dynamic Systems Development Method
IREB	International Requirements Engineering Board
GQM	Goal Question Metric
IT	Information Technology
KPI	Key Performance Indicator
NS	Nederlandse Spoorwegen
REMS	Requirements Engineering Maturity Scan
RUP	Rational Unified Process
SME	Subject Matter Expert
SRS	Software Requirements Specification
XP	eXtreme Programming

1. Introduction

The importance of software has grown tremendously during the last 40 years. Based on a McKinsey & Company research on large scale IT projects, data has shown that large IT projects run 45 percent over budget and 7 percent over time on average, while delivering 56 percent less value than predicted [McKinsey, 2012]. Cost estimation is a difficult and demanding activity [P. Armour, 2002], [Boehm 1981]. Requirements tend to change over time [N. Nurmuliani, D. Zowghi, S. Fowell, 2004], and cost estimation is a difficult and demanding activity [Boehm 1981].

Companies try to align business and IT the best they can. The demand for IT projects often starts with a business problem. When the business requires an IT solution for their problem, IT provides the necessary support. To improve the success rate of IT projects, regularly it is important to meet business satisfaction in minimal time and with the lowest cost possible. One way of achieving business satisfaction is to have adequate business requirements to work with. With requirements, IT will have a better understanding of what the business wants as a solution. Different methods are used for addressing business requirements in IT projects. The most common methods are the traditional methods and incremental methods.

In traditional methods during the software development life cycle (SDLC), projects often work in phases. When one phase ends, another one will be picked up until the project has ended. The Waterfall Method is an example of a traditional method, which assesses and builds on the users' needs, forming a complete analysis of user requirements [Royce, 1970]. Rational Unified Process (RUP) is an iterative software development process framework [IBM, 2003], which is an adaptable process framework. The Agile Unified Process (AUP) is created which is a simplified version of RUP [S. Amber, 2013].

The design of incremental development is to offer time-savings and to better handle risks [Boehm 1981]. Agile is an example of incremental development, and Agile development have become popular during the last few years. Several Agile methods exist: Adaptive Software Development, Crystal, Dynamic Systems Development, eXtreme Programming (XP), Feature Driven Development, Lean Software Development, Scrum [Beck et al., 2001]. Evolutionary methods like iterative development and the Spiral Model [Boehm, 1988] aim to better handle changing requirements and manage risk.

IT project developments evolve over time. In PRINCE2, a project is defined as "a temporary organization that is created for the purpose of delivering one or more business products according to an agreed Business Case." [M. Van Onna, A. koning, 2007]. There is a start and an end to a project. Ideally, the result should be a working product for the business or end users. There is a difference between gathering requirements in a traditional software development method and an incremental software development method. In traditional methods, requirements are gathered at the beginning of the project and the working product will be developed at the end. However, studies have shown that it is more expensive to adjust codes in the development process, than to rewrite sentences in a requirement document. In other words, when the product is ready to be delivered, the expenses to change something can be far costlier [Boehm, 1983]. Incremental software development methods tend to gather requirements during the IT project and evolves over time.

For enabling IT standardization and optimization, it is interesting to examine the measurement of IT project conditions in an early stage, where software delivery is required in a fast-moving environment and where the software will properly meet customers' changing needs. This is where the opportunity was noticed to do research in requirements engineering, as most IT projects start with defining requirements before entering software development. The empirical value of this research is relevant to academic fields and business practices. This study will be an empirical literature study on metrics in requirements engineering in a complex IT environment. The focus of this research is on companies with Agile development expertise. A case study conducted at the Dutch Railway company *Nederlandse Spoorwegen* (NS), a consultancy company and an international bank are provided in this study.

1.2 Problem statement

Every IT project is different. Requirements analysis is critical to the success of a systems or software project [A. Abran, J. W. Moore; P. Bourque, R. Dupuis, 2005]. While requirements engineering itself is not a new study, and metrics in requirements is not either. The current specification for software requirements may have a good outcome for traditional software development, but it is unknown if metrics for requirements engineering is a beneficial lightweight method. It is posing a challenge to find a lightweight method to measure the quality of requirements engineering.

Requirements engineering is a combination of understanding technology and human factors. It demands the expertise of hardware, software, and people skills. The common problems in software development are the lack of customer knowledge in technology, cultural issues, requirement changes during the project, timeline troubles, and inadequate communication [J. D. Herbsleb, D. Moitra, Lucent Technologies, 2003]. Meeting all these factors would be the ideal situation, but there is always a shifted balance between time, quality and money. See fig 1.

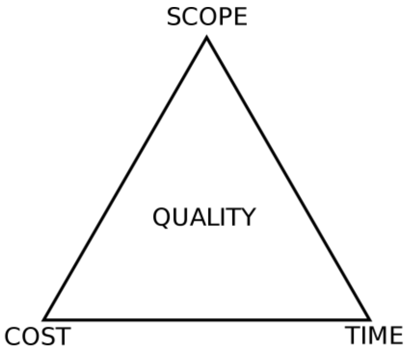


Fig. 1 - Triple Constraint, Iron Triangle [R. Atkinson, 1999]

In table 1 the frequency of requirements error is put together in a table. It shows the defect potential with the removal efficiency and the delivery effect.

Defect	Defect potential	Removal efficiency	Delivery effect
Requirements	1.00	77%	0.23
Design	1.25	85%	0.19
Coding	1.75	95%	0.09
Documentation	0.60	80%	0.12
Bad fixes	0.40	70%	0.12
Total	5.00	85%	0.75

Table 1: Frequency of requirements error. Source: Data derived from Jones [1994]

This thesis will present the benefits of metrics in requirements engineering and how metrics add value to a project.

1.3 Research Relevance

Research of requirements engineering will provide results that benefit academics as well as business. The research relevance of requirements engineering metrics can be divided into several levels. The relevance for development teams is to find quality in the intended features that will perform accordingly. The relevance for management is to save costs and have a monitoring tool. The research relevance on academic level is to provide theoretical support.

Business relevance

For companies, it is not desirable to have a misunderstanding between business and IT. It undermines the quality in the requirements. Reflecting upon the measurements of requirements engineering could potentially lead to cost reduction and improve quality in IT projects. This research aims at helping organizations analyze, assess and improve their quality in requirements engineering. The IT project can address the problem they need to solve, by connecting to the business on an early stage, and keeping them informed on the way they will work.

The IT projects team can track their “Way of Working” and how well their requirements are defined. First, the maturity level of requirements engineering will be captured and analyzed. The IT project team will have a Zero measurement to start with, so it is made easy to track challenges. Second, bottlenecks are identified, and improvements are made in the requirements engineering process. Finally, the current “Way of Working” is put next to the new conceptual model, and requirements are compared with the new lightweight method to observe the data.

For a better understanding of optimizing IT projects, finding a way to measure the quality of requirements engineering is relevant for research. There are advantages for having metrics in requirements. For example, it is possible to measure “quality” in requirements during projects, without changing codes or changing the whole IT infrastructure. Another advantage is: businesses can be involved more rapidly in the process. Development teams have more periods of receiving feedback. Moreover, assessment results show specific data that is relevant for the business, and it helps to make forecasts.

Studies performed at companies like IBM, HP, GTE have measured and assigned costs to errors occurring at various phases of the project lifecycle [Davis, 1993]. Studies have determined that fixing problems early in the SDLC is less expensive than to address the problems later. Changes in projects can be costly. See figure 2 for the cost of change at various phases of the SDLC. The conclusion of the studies is: if a unit cost of one is assigned to the effort required to detect and repair an error during the coding stage, then the cost to detect and repair an error during the requirements stage is between five to ten times less. When compared during the maintenance stage, the cost to detect and repair an error is many times more.

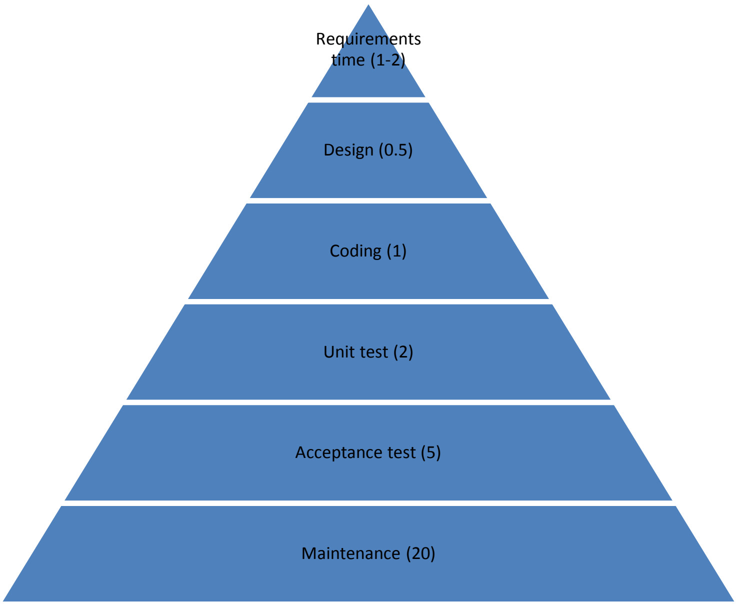


Fig. 2 - Relative cost to repair a defect at different lifecycle phases [Davis, 1993].

The research relevance also has management implications where risk management is concerned. "Risk Management is the application of tools and procedures to contain project risk within acceptable limits. Risk management provides a standard approach to identify and document risk factors, evaluate their potential severity and propose a strategy for mitigating them" [William, Walker, and Dorofee, 1997]. Risk management includes the following activities [McConnel, Steve, 1996]:

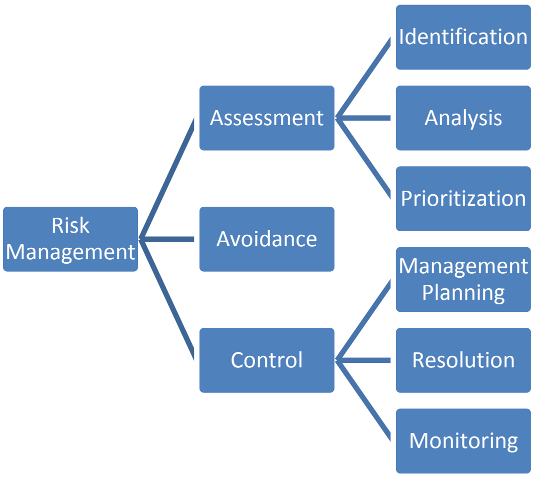


Fig. 3 - Elements of Risk Management

Academic relevance

Requirements engineering is widely used in practice, but scientific research in lightweight requirements metrics in complex IT environments is limited. This study uses a scientific perspective of forming requirements engineering measurements to make IT projects more effective. Moreover, the practical perspective from the case study will add value to existing case studies and current scientific research.
Furthermore, the measurement method for requirements engineering, can be interesting for relevant studies in the field of software engineering.

1.4 Research questions

To have a focus point for the study, a set of research questions are created. The research question and the sub-questions guide this thesis and the research study. The goal of this study is to find a solution for measuring the quality of requirements engineering in IT projects. Therefore, the following research question and subsequent questions have been formed.

Research question:

How to design a lightweight method to measure the quality of requirements engineering for software development in complex IT environments?

Sub questions

The research question will be answered with the following sub questions:

1. What are the main challenges in requirements engineering?
2. Which methods can be used to measure requirements engineering?
3. What are the criteria for lightweight requirements engineering?
4. What are the KPI's in requirements engineering?
5. How can a conceptual model be implemented at the business case?
6. How can we evaluate the implemented model?

1.5 Thesis scope

The scope of this study is to find a solution for requirements in a "complex IT environments". In this study, complex IT environments have multiple projects running, parallel IT architectures, parallel databases, parallel networks on a large scale. Due to a large number of organizations that fall into this scope, the outcome of the study aims to be as lightweight as possible.

While the researched companies are all familiar with Agile methodology, most of the companies were still employing methods in line with the traditional software development methodology. Due to this reason, this thesis will still have a small focus on the traditional methodology.

There are various ways to develop with Agile methodology. Scrum[1] will is considered for this study, as this is the most popular framework for implementing agile.

Assuming the complex IT environment have a matured requirements process, only parts of the Agile structure are considered for this study. The requirements written on Epic level, User Story level, and Subtask level are part of the study. See figure 4. The Initiative is left out, because this is a collection of epics to a business goal.

Furthermore, requirements that are gathered after the delivered developed product are left out of the scope of this study as well. Requirements from "the deliverable" is part of the final product of the SDLC, while this study aims to focus on the requirements before delivering. When requirements are still added to the product, this should be taken up by the maintenance of the project or a department like the service desk.

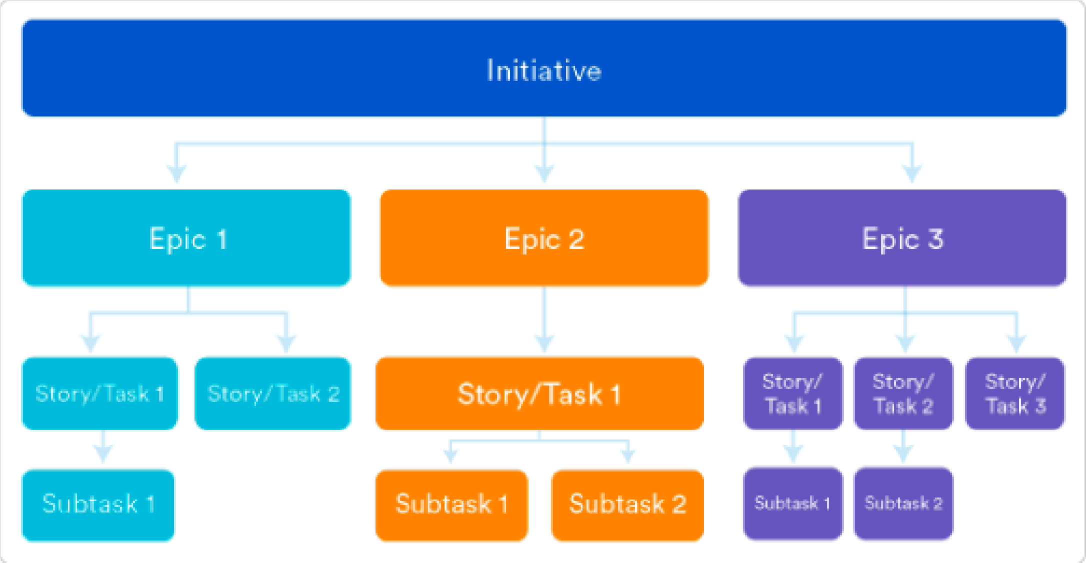


Figure 4 - Agile structure: Initiatives, epics, stories, subtask[2]

[1] https://www.atlassian.com/agile/scrum

[2] https://www.atlassian.com/agile/project-management/epics-stories-themes

1.6 Thesis overview

The outline of this thesis is presented in Table 2. Chapter 1 begins with an introduction of the research in a general sense. After that, it is explained how the research process was executed scientifically. The related work and methodology are described in the following chapters. In the final chapters, the empirical findings are discussed, leading to the conclusion and recommendations.

Chapter	**Outline**
1	This chapter describes an **overview** of the thesis. The problem statement, research relevance, research questions, thesis scope and thesis overview are presented.
2	The **methodology** of the research is explained, and it includes information about the case study, validity, data collection and data analysis.
3	A scientific review of the **related work** describes the essence of software development methods, requirements engineering, and metrics for quality.
4	This chapter presents the **results** regarding the case study overview, the design, implementation and evaluation of the conceptual method.
5	In chapter 5, the research questions are answered by combining the analysis of the gathered empirical data in the **discussion**.
6	In the final chapter, the **conclusion and recommendations** will present the conclusion, limitation of the research and recommendations for future work.

Table 2 - Thesis Outline

2. Research Methodology

This research has a deductive approach. A deductive approach begins by gaining theoretical knowledge first, followed by defining the research question and developing a model. The model is implemented in a case study at three different companies.

A pilot is created to test the hypothesis and to see what the challenges are in the sub-questions. The goal of the pilot is to find the optimal guideline of the Assessment flow. With the pilot, the Assessment evolves according to the received feedback. It is possible to find the team performance, the involvement of the people, actions that need to be taken and it also allows the teams to have a review of the outcome initiating the "who, what, when, why" is essential for future planning of the model. Finally, the results of the case studies and the results of the research are observed and concluded in this thesis.

Research method

A systematic literature review is used as the research method. The theoretical assumption is supported by existing theories from scientific papers and books. The scientific papers are collected from Google Scholar and carefully selected for analysis. The theoretical literature is also collected from libraries. The Leiden library and The Hague public library have a wide selection of books, and the information is collected from those books.

Research design

A collection of related concepts guides the empirical research by gaining knowledge of direct or indirect literature observations. Afterward, the information is analyzed qualitatively. With a qualitative analysis of the literature review and field research, the existing literature is evaluated. The research design of the study has a case study at different companies. During the case study, a survey is conducted, interviews are planned and performed with the selected experts, like IT project managers, business analysts, software developers, and other requirements engineering experts.

2.1 Research approach

A method for measuring requirements engineering is assessed and developed. To support the research, a case study is conducted at *Nederlandse Spoorwegen* (NS), a consultancy company and an international bank which are all based in The Netherlands.

Case study at	Organization size	Industry	Agile/traditional environment
Nederlandse Spoorwegen	33 000	Transport	Both[3]
International Bank	135 000	Financial Services	Both, but more Agile[4]
Consultancy company	>200 000	Consulting, Technology, Digital Transformation Services	Mainly Agile[5]

Table 3: Overview of the reviewed companies

2.1.1 Case study overview

Nederlandse Spoorwegen

The case study was done in 2012. It was a period of 6 months. NS is a Dutch company in the passenger railway operating business. NS provides rail services on the Dutch rail network, with 33,000 employees[6]. The core business of the researched company is not focused on software development, but they do have specialized teams for their IT projects. IT projects at NS are deployed at IT-Operations, which is a segment of NS IT. The requirements managers at IT-Operations are responsible for the development and sourcing of the IT projects.

The Dutch Railway company had a CMMI[7] maturity level of 1 in 2012. In practice, NS has its own processes for operating IT projects. Analyzing these processes are essential for the business case. The IT teams do not use independent test agencies to measure their requirements engineering processes. Requirements engineering is measured with a check dashboard. With this dashboard, the manager indicates the current requirements engineering score of how well they perform. To obtain valid requirements to achieve quality outcomes in their IT projects, the proposed method can support the IT teams.

[3] Data acquired from 2012
[4] Data acquired from 2016
[5] Data acquired from 2017
[6] NS, 2013
[7] https://cmmiinstitute.com/learning/appraisals/levels

International bank

The case study at the international bank was in 2016. It was a period of 1 year. The international bank has its headquarter in Amsterdam and is listed as one of the largest banks in the Netherlands. The bank has more than 135,000 employees and is nationalized by the Dutch government. The bank had a CMMI maturity level 2 in 2016.

The International bank has its own processes for operating IT projects. Most of the IT teams work with an Agile method and are adopting the Scrum method.

Consultancy company

The case study at the consultancy company was done between 2017 and 2018. The consultancy company has more than 200.000 employees worldwide. The CMMI maturity level is 1 or 2, and most of the employees have Agile knowledge. The consultancy company has its own projects within the company. The requirements experts of internal projects were asked to participate in this study.

2.2 Data collection

2.2.1 Interviews

During the business case, semi-structured-interviews are conducted. One set of the interviews were focused on how the company is currently working in teams and projects, how the requirements are gathered and processed, what would be interesting to include for their company and how they define quality in their requirements. Another set of interviews were focused on improving the Maturity Scan. The focus point of this interview set is to see what is necessary for teams to have in improving their current requirement process.

2.2.2 Survey

Due to time limitation for writing the thesis, the limited availability of experts, and the limited time to perform research at the companies, it was not possible to interview all the experts. Therefore, another solution was found to gather the most data from experts. Surveys were used to contact a more significant number of people at the companies. The survey was created in an online form, and it was sent out to participants. The survey was filled in by experts in requirements engineering, who were experienced with the Agile methodology and the people who work closely with requirements.

2.3 Data analysis

The outcome of the interviews and the surveys were collected and analyzed. The processed data was used as input for this study.

Interviews were conducted with experts in requirements engineering to gather in-depth information. Invitations were sent out by email, and the list of contacts were provided by the management. Among the interviewees, there were business analysts, software developers, architects, projects leaders, scrum team members, and scrum masters. Interviewees had a minimum experience of 3 years working in IT projects, Agile methodology and in requirements engineering.

2.4 Creating Requirements Engineering Maturity Scan

The First design of the Maturity Scan started in Microsoft Excel, where the Maturity Scan could easily evolve from. The final design of the Maturity Scan was created in a browser software.

2.4.1 Design

The design of the Maturity Scan started with a pilot scrum team, who works according to the current situation of the company. This pilot team helped with maturing the Maturity Scan. Surveys and interviews were conducted with the pilot team to gather feedback. After improving the first draft, the matured version was rolled out to another pilot team. And another set of surveys and interviews were done. In the end, around 25 scrum teams have been using the Maturity Scan.

Requirements for creating the Requirements Engineering Maturity Scan are:

- Lightweight metrics assessment for requirements engineering
- Applicable for both traditional and iterative software development methods
- The model should lead to cost reduction in the development process
- Suitable for complex IT landscapes

Lightweight metrics assessment for requirements engineering

The assessment should be generally understood by people in the project. When requirements are gathered, there is often still a need to talk to the business. The Maturity Scan should be as lightweight as possible, and jargon should be limited. It was useful to speak in business terms so that the business understands what the Maturity Scan is. The quality of the requirements will improve when stakeholders are on the same level.

Applicable for both traditional and iterative software development methods

The assessment should be designed in a way that it is applicable throughout the domains of the company whether they use the traditional or iterative software development method. However, the focus of this study was on the agile development method.

The model should lead to cost reduction in the development process

Rework is expensive. This assessment model helps projects to measure the quality of requirements, which potentially lead to cost reduction in the development process, because (hopefully) no rework is needed.

Suitable for complex IT landscapes

Some projects have complex IT landscapes. This assessment is suitable for complex IT landscapes and simple IT landscapes, with the assumption that if the assessment is applicable for complex IT landscapes, it will also be applicable for simple IT landscapes.

2.5 Validity

A method to validate the Maturity Scan Assessment is essential. Validation of the methods begins before creating and developing the REMS.

First, validation methods are found in literary studies to have a foundation for the Assessment. The literature study can be found in chapter 3 in this thesis. Second, validation of the methods is formed and confirmed by the experts by doing interviews and filling in surveys. The key findings of the interviews and surveys can be found in the discussion, results, and appendix. Finally, the validation method is used on the Maturity Scan, where people gave feedback on REMS. The tool will be evaluated by the people who used it.

Information-gathering techniques to validate REMS are the following:

	Nederlandse Spoorwegen	International bank	Consultancy company
Interviews	Yes	Yes	Yes
Facilitated requirements workshops	Yes	Yes	Yes
Document analysis	Yes	Yes	Yes
Surveys	Yes	Yes	Yes
Customer site visit	Yes	Yes	Yes
Business process analysis	Yes	Yes	Yes
Work flow and task analysis	Yes	Yes	Yes
Event lists	Yes	Yes	Yes
Competitive product analysis	No	No	No
Reverse engineering of existing systems	No	No	No
Retrospective performed on the previous project	No	Yes	Yes

Table 4 – Gathering techniques to validate REMS

3. Literature review

In this chapter, the literature review of measuring quality in requirements engineering is put into context. The following section describes the relevant and existing empirical knowledge about metrics and requirements engineering in complex IT environments. The context that is found on this topic is used during the interviews with the subject matter experts (SME). This way, the interviewees gave a more in-depth perspective on the subject.

The search for the literature was in Google Scholar for relevant papers, books, and grey literature. The literature found were mostly academic literature, and were sorted on relevance and it was selected to match the keywords in the abstract or the title.

The evaluation of the paper is done by asking the SME and the case studies that were used during the literature study. The first group of experts were asked to evaluate the conceptual maturity model and the quality of it. The second group of experts were asked to evaluate the conceptual maturity model after using it for 4 months.

With the survey, the outcome is:

- Finding the relevant maturity levels of requirements engineering;
- Finding the level of complexity in IT projects;
- Finding the way of working in project teams;
- Finding out whether the conceptual model was comprehensible;
- Finding out how pragmatic the conceptual model was;
- Finding out what should be improved in the conceptual model;
- Finding out if the conceptual model was adequate for complex IT projects.

Sub research question	Method	Discussion results
1. What are the main challenges in requirements engineering?	Literature review, interviews with experts	Chapter 3
2. Which methods can be used to measure requirements engineering?	Literature review, interviews with experts, case study	Chapter 3
3. What are the criteria for lightweight requirements engineering?	Literature review, interviews with experts, case study, survey	Chapter 3
4. What are the KPI's in requirements engineering?	Literature review, interviews with experts, case study	Chapter 3
5. How can a conceptual model be implemented at the business case?	Case study, survey	Chapter 4
6. How can we evaluate the implemented model?	Case study, survey	Chapter 5

Table 5: link between research question and research method

3.1 Traditional Software Development Methods

Various methods can be used within a traditional software development method to deliver software. The Waterfall model (Royce) had the following order: "Requirements", "Design", "Coding and unit test", "System integration" and "Operations and Maintenance". In this study, the search on Traditional Software Development methods were done in Scholar for scientific papers and books on Software Development. Keywords in Google Scholar were "traditional software development", "software development" and "waterfall software development".

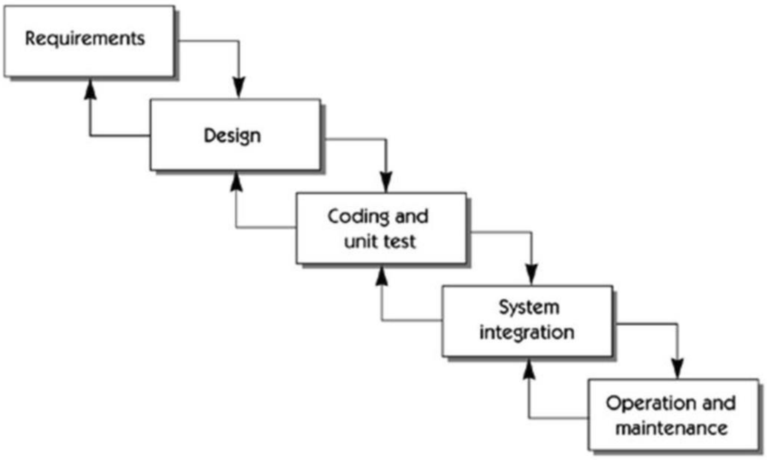


Fig. 5: Waterfall Model - Royce 1970

3.2 Agile Software Development Methods

The search on Agile Software Development methods was done in Scholar for scientific papers and books on Software Development. Keywords in Google Scholar were "Agile software development", "software development" and "scrum software development".

Agile Software Development

The Agile Manifesto [Fowler & Highsmith, 2001] was produced by 17 developers, the Agile Alliance, during an outing in 2013. According to agilemanifesto.org, they are "uncovering better ways of developing software by doing it and helping others do it." The Agile Alliance developed 12 principles of Agile. The focus of Agile is on individuals and interactions, working software, customer engagement and response in change. The iterative nature of Agile allows companies to adopt changes in an early phase of development improving quality and customer satisfaction. See table 6.

Individuals and interactions	**over**	**Processes and tools**
Working product	over	Comprehensive documentation
Customer collaboration	over	Contract negotiation
Responding to change	over	Following a plan

Table 6: The Agile Manifesto Principles[8]

Scrum Methodology

[8] www.agilemanifesto.org

Scrum is a framework within people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.[9] The Scrum-creators have written the "Scrum Guide" to explain Scrum. According to Scrum.org, Scrum is lightweight, simple to understand and difficult to master. The researched companies are all familiar with this framework.

Common Scrum terminology[10]

Term	Description
Daily Scrum	daily time-boxed event of 15 minutes, or less, for the Development Team to re-plan the next day of development work during a Sprint. Updates are reflected in the Sprint Backlog.
Done	a shared understanding of expectations that software must live up to in order to be releasable into production. Managed by the Development Team.
Increment	a piece of working software that adds to previously created Increments, where the sum of all Increments -as a whole - form a product.
Product Backlog	an ordered list of the work to be done in order to create, maintain and sustain a product. Managed by the Product Owner.
Product Owner	the role in Scrum accountable for maximizing the value of a product, primarily by incrementally managing and expressing business and functional expectations for a product to the Development Team(s).
Scrum Master	the role within a Scrum Team accountable for guiding, coaching, teaching and assisting a Scrum Team and its environments in a proper understanding and use of Scrum.
Scrum Team	a self-organizing team consisting of a Product Owner, Development Team and Scrum Master.
Sprint	time-boxed event of 30 days, or less, that serves as a container for the other Scrum events and activities. Sprints are done consecutively, without intermediate gaps.

Table 7: Common Scrum terminology

[9] www.scrum.org

[10] https://www.scrum.org/scrum-glossary

3.3 Requirements Engineering

With the many studies about Requirements engineering, it has gradually grown into a matured IT subject. Before getting to measuring requirements engineering, it is good to understand the different standards in requirements engineering.

The search on defining Requirements Engineering was done in Google Scholar for scientific papers and books on Software Development, with a focus on requirements engineering. Keywords were "requirements engineering", "requirements in software development" and "requirements models".

In the book of Software Requirements [Karl E. Wiegers, 2003], "Requirements" is defined as "Anything that drives design choices" [Lawrence, 1997]. The IEEE Standard Glossary of Software Engineering Terminology (1990) defines requirements as

1. A condition or capability needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
3. A documented representation of a condition or capability as in 1 or 2.

"Requirements management" is defined in *Managing software requirements* [D. Leffingwell, D. Widrig; 2007] as "a systematic approach to eliciting, organizing, and documenting the requirements of the system, and a process that establishes and maintains agreement between the customer and the project team on the changing requirements of the system."

3.4 Requirements Engineering Metrics

Defining "quality" in requirements engineering is complicated. There are certain standards to measure the quality in requirements. In this section, the criteria for defining quality in requirements are described. What criteria can be used for prioritization, and what metrics can be used for quality validation.

The search on metrics in requirements engineering was done in Google Scholar for scientific papers and books on Software Development. Keywords in Scholar were "requirements engineering metrics", "metrics in software development", "quality requirements", "maturity model requirements" and "measuring requirements".

In the below section, the definitions of the methods are explained. With the following metrics, it is made possible to measure requirements:

- Software Requirements Specification
- MoSCoW
- Capability Maturity Model
- ISO 9126
- Goal Question Metrics
- Bootstrap
- Requirements Elicitation

Software Requirements Specification

An IEEE guide for Software Requirements Specification (SRS)[11] was introduced, aiming to specify the requirements of software that is yet to be developed.

The SRS examines requirements on the following qualities:

Unambiguous	Electronically stored
Complete	Executable/interpretable
Correct	Annotated by relative importance
Understandable	Annotated by relative stability
Verifiable	Annotated by version
Internally consistent	Not redundant
Externally consistent	At right level of detail
Achievable	Precise
Concise	Reusable
Design independent	Traced
Traceable	Organized
Modifiable	Cross-Referenced

Table 8: SRS qualities

MoSCoW

For creating the Requirements Engineering Maturity Scan, requirements must be set to have a better understanding of the goal of this study. The MoSCoW method was applied for this set of guidelines.

The Maturity Scan is created with the MoSCoW method, together with the knowledge of experts. The MoSCoW method is used to prioritize importance in project management, business analysis, and software development. The MoSCoW is an acronym for "Must have", "Should have", Could have" and "Would have". This prioritization method creates a narrower direction for developing the Maturity Scan.

CMMI

The Capability Maturity Model Integration (CMMI) models provide guidance for developing or improving processes that meet the business goals of an organization. The model can also be used as a framework to appraise the process maturity of an organization. The model contains the essential elements of effective processes for one or more bodies of knowledge [Crosby 79, Juran 88, Deming 86, Humphrey 89]. CMMI has two maturity levels for requirements: requirements development and requirements management.

The Maturity Scan does not measure the CMMI of a company. However, companies are measured on their CMMI level for this study to have an initial starting point to measure the company and to see what the goal is of the company. Assumptions are made with the CMMI model, and companies can compare their CMMI level after the Maturity Scan.

[11] 830-1984 — IEEE Guide to Software Requirements Specifications. 1984. doi:10.1109/IEEESTD.1984.119205. ISBN 0-7381-4418-5.

ISO 9126

Requirements are divided by functional and non-functional requirements. Non-functional requirements can be measured by ISO/IEC 9126 [Van Zelst e.a., 1996] and the model is published in 2001. Internal metrics were published in 2003 and external metrics with measurement regulations in 2004. The ISO 9126:2001-norm has 27 quality characteristics and it is divided into six domains: functionalities, reliability, usability, efficiency, maintainability, and portability.

Goal Question Metrics

The Goal Question Metric (GQM) [Basili, 1994] support the traceable software engineering process. GQM has measurement model on three levels:

- Conceptual level (Goal)
  An object has a defined goal for certain reasons (e.g. various models from different perspectives).
- Operational level (Question)
  Questions are set and used to define models of that object and focuses on achieving a specific goal.
- Quantitative level (Metric)
  Metrics are created based on the models and is associated with measurable questions.

Basili described his six-step GQM process as follows:

1. Develop a set of corporate, division and project business goals and associated measurement goals for productivity and quality.
2. Generate questions (based on models) that define those goals as completely as possible in a quantifiable way.
3. Specify the measures needed to be collected to answer those questions and track process and product conformance to the goals.
4. Develop mechanisms for data collection.
5. Collect, validate and analyze the data in real time to provide feedback to projects for corrective action.
6. Analyze the data in a post mortem fashion to assess conformance to the goals and to make recommendations for future improvements

Bootstrap

Bootstrapping [B. Efron, R. Tibshirani, 1993] is a test or metric that relies on random sampling with replacement. Bootstrapping is a simple simulation method for frequentist inference. The bootstrapping method is useful when standard assumptions are invalid, standard problem has non-standard twist, complex problem has no (reliable) theory or (almost) anywhere else [Anthony Davison], 2012.

Requirements Elicitation

The requirements elicitation process is a set of activities that allow communication, prioritization, negotiation, and collaboration with the relevant stakeholders.

In 2004, Goldsmith suggested a "problem pyramid" of "six steps which must be performed in sequence" [Goldsmith, 2004].

1. Identify the real problem, opportunity or challenge
2. Identify the current measure(s) which show that the problem is real
3. Identify the goal measure(s) to show the problem has been addressed and the value of meeting it
4. Identify the "as-is" cause(s) of the problem, as it is the causes that must be solved, not the problem directly
5. Define the business "wants" that must be delivered to meet the goal measure(s)
6. Specify a product design how to satisfy the real business requirements

Comparison

Table 9 shows the comparison between the different requirements metrics with the usage of the method in the organization of study.

Method	Used at the organization	Characteristic	Qualitative/ quantitative	RE process
SRS	/	Description of software to be developed.	Qualitative	Specification
MoSCoW	NS, International Bank, consultancy company	Prioritization technique.	Qualitative	Analysis
Capability Maturity Model	NS, consultancy company	Defines process maturity levels.	Quantitative	Validation
ISO 9126	NS	International standard for software quality.	Qualitative	Validation
Goal Question Metrics	/	Software metric.	Qualitative	Analysis
Bootstrap	International Bank	Metric with random sampling with replacement.	Quantitative	Validation
Requirements Elicitation	NS, International Bank, Consultancy company	Capture requirements. Examples: interviews, questionnaires, use cases.	Qualitative	Elicitation

Table 9: Comparison between different requirements metrics

4. Results

The proposed method to measure the quality of requirements engineering for custom software development is called the “requirements engineering maturity scan”, in short, “REMS”. The first version of REMS is called REMS 1.0 and the second version is called REMS 2.0. In the future, the tool can be evolved to REMS 3.0 where DevOps can measure their requirements. See Fig. 6. Elaboration on REMS 3.0 can be found in chapter 6.3.

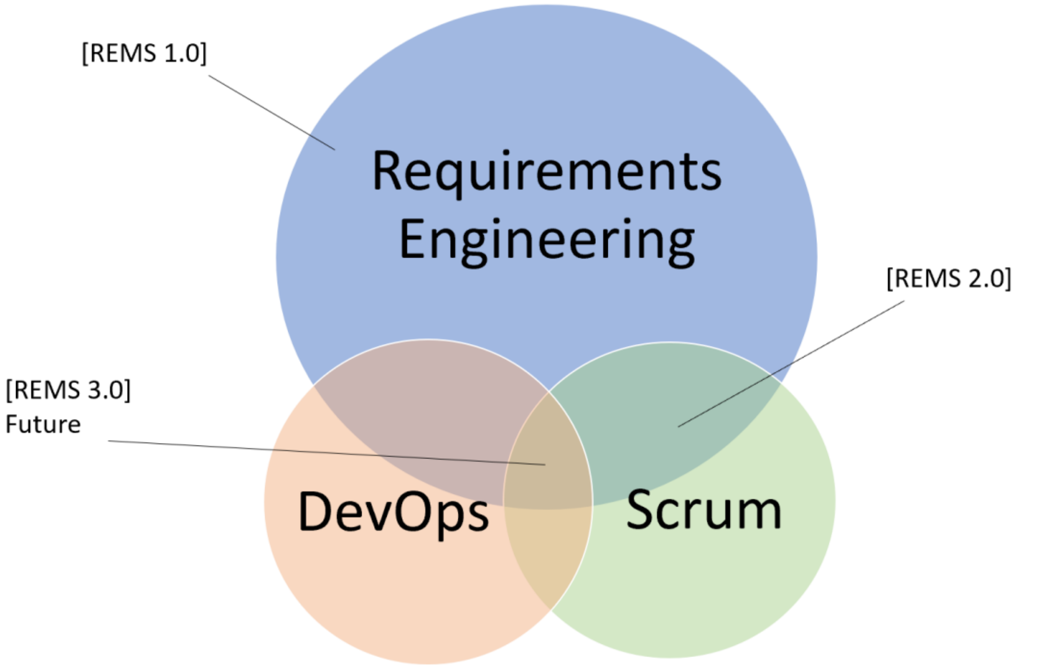


Fig. 6 - Relationship between REMS 1.0 and REMS 2.0

In the theoretical framework, the literature study supported the study to find metrics to measure “quality”, “lightweight”, and “requirements engineering”. Definitions were defined clearly, and a set of metrics were grouped to form the assessment. When the prediction of requirements activities is precise, it potentially results in cost saving or time efficiency for their IT projects.

The objective of the research is to find metrics for the Maturity Scan in IT projects. The initial design of the Requirements Engineering Assessment Template was created to be used during agile and traditional IT projects when the engineer is gathering requirements for the business case or the user story. The assessed metrics are derived from multiple sources which were based on theoretical findings, underpinnings and business developments.

The interview and survey results can be found in Appendix 2.

4.1 Design of the proposed Maturity Scan

Before designing the Maturity Scan, it was necessary to find out the way of working of the company. Some companies work more traditional and some companies work more Agile. Assessment questions support both methodologies.

During the development of the Maturity Scan pilot, the MoSCoW of the Maturity Scan was prioritized. Some items in the Maturity Scan were not applicable for a large scale of complex companies, and those items are left out intentionally. With the help of the experts, a shift in the MoSCoW was created, and the Maturity Scan was improved as the Maturity Scan matured.

The proposed method is composed of a list of questions that will measure the way team members are working, the quality of the requirements, and the team spirit. After filling in REMS, the team members, projects leaders and other stakeholders can evaluate the results. After a period of filling in the REMS, a trend is shown in the charts. The team members can see where the peaks are, or any bottlenecks and decide for future improvements.

Phase	**Goals**	**People/stakeholders**	**Scope/context**	**Measurement**
Orientation	Define deliverable	Manager	Set time and budget	Plan of approach (go/no go)
Preparation	Information gathering and analysis	Project manager, dev team, analysts, scrum master	Subject selection/scope	First version of REMS
Implementation	Implement REMS	Pilot team	Data gathering	REMS trend of pilot team
Feedback	Receive feedback from pilot team	Pilot team, project manager	Find bottlenecks and improvement areas	Results from survey and interviews
Closing project	Deliver REMS	Manager, project manager, dev team, analysts, scrum master	Improved REMS implemented at company	Feedback from company

Table 10: Phases of designing REMS

To have a sound measurement, a pilot team was measured first. After that, multiple projects were using the same assessment to track their performance. Results of the assessment were discussed during the retrospective session per team, to see if there are any improvements for the next sprint. Results of the retrospective, per team, were gathered in a dashboard to see the improvements in the project.

4.1.1. The relation between REMS 1 and REMS 2

REMS 1.0 was designed for NS and initially was meant as a generic assessment tool for more companies. The idea was to create a lightweight tool for the management to make informed decisions on a strategic or tactical level. NS worked with the traditional software development method that has an iterative cycle. There were some activities taken from the Scrum method, but it was not fully Agile. REMS 2.0 was created, because the second company that was studied needed more Agile and Scrum knowledge on their teams than NS.

4.1.2. REMS 1.0

The case study done for REMS 1.0 is found in this section. REMS 1.0 consists of two parts: Maturity Scan and Quick Scan. The Maturity Scan consists of the full list of questions, and the Quick Scan is meant to only measure the appropriate metrics for that specific team. The layout and framework are similar to each other. The only difference is the selection of questions.

The interview setup and survey setup can be found in Appendix 1. The delivered advisory document to the NS can be found in Appendix 3. This document includes the full list of the quality checklist of the table below.

REMS 1.0 started with selecting meaningful assessment questions. REMS 1.0 consists of quality questions that were important to the assessment and NS. In the table below, the subjects and the number of questions are listed. The full list of questions can be found in the advisory document in appendix 3. The original set of questions was created in Dutch.

Subject	**Amount of questions**
Problem analysis	Problem statement (5 Q)
	Goal statement (5 Q)
	Cause analysis (2 Q)
	System model (6 Q)
	Stakeholder analysis (5 Q)
	Constraints (3 Q)
	Actors (5 Q)
	Business Use Case Model (3 Q)
Users /stakeholders needs	Interviews (5 Q)
	User analysis (4 Q)
	Workshop (6 Q)
	Features (3 Q)
	Storyboard (3 Q)
System definition	Requirements (5 Q)
	Vision (3 Q)
	Use case identification (1 Q)
	Product Manager (3 Q)
	Commercial factor (1 Q)
Scope	Priorities (1 Q)
	Requirements (2 Q)
	Communication (3 Q)
	Expectations (2 Q)
System refinement	Use case models (13 Q)
	Use case specification (14 Q)
	Other specifications (5 Q)
	Ambiguous specification (2 Q)
	Technical methods (2 Q)
System analysis	Transition method (2 Q)
	Test case (3 Q)
	Use case traceability (4 Q)
	Change management (5 Q)
	Requirements method (4 Q)

Table 11: List of assessment questions REMS 1.0

Before creating REMS 1.0 in Excel and .NET, an Entity Diagram is developed in MS Visio. See figure 7 below. The Entity diagram was used to create an overview of the elements that are involved in creating the tool. The next step was to create the tool in .NET.

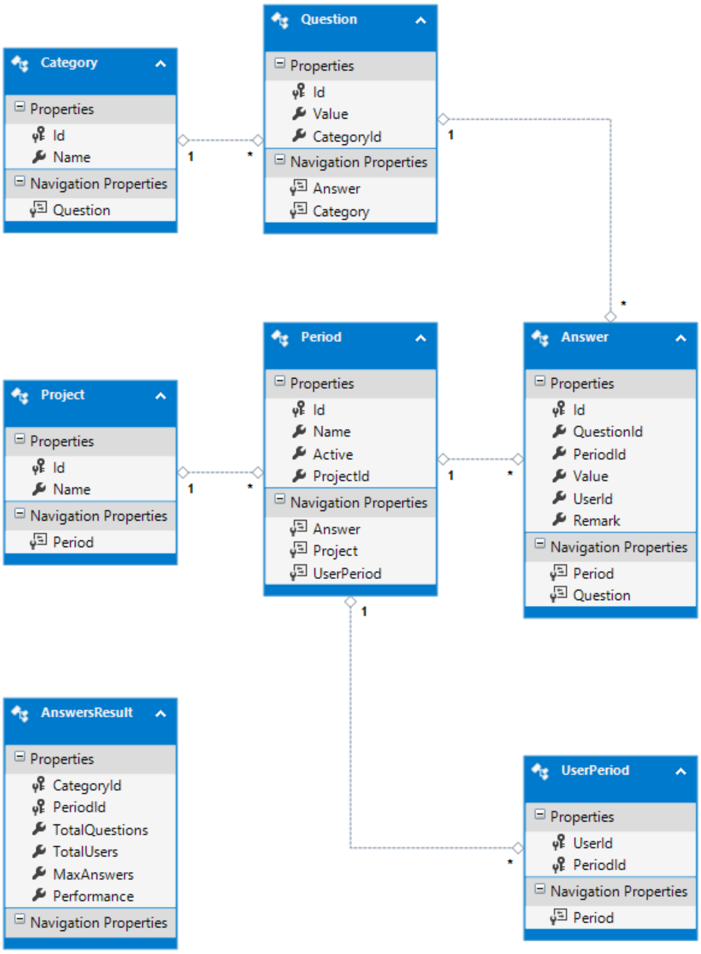


Fig. 7: Development of REMS 1.0. Entity Diagram

REMS 1.0 Quick Scan

REMS 1.0 is created in Excel and it has a front page with the NS logo. The screenshots of the Quick Scan and the Maturity Scan are shown in the section below. Both scans have a logon screen where people can register themselves as the user.

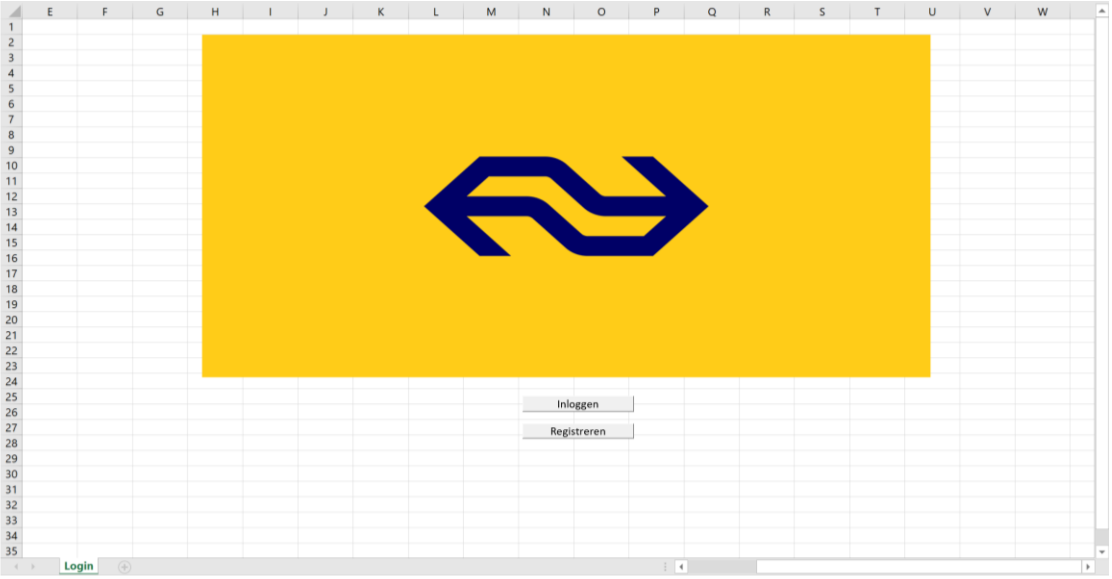
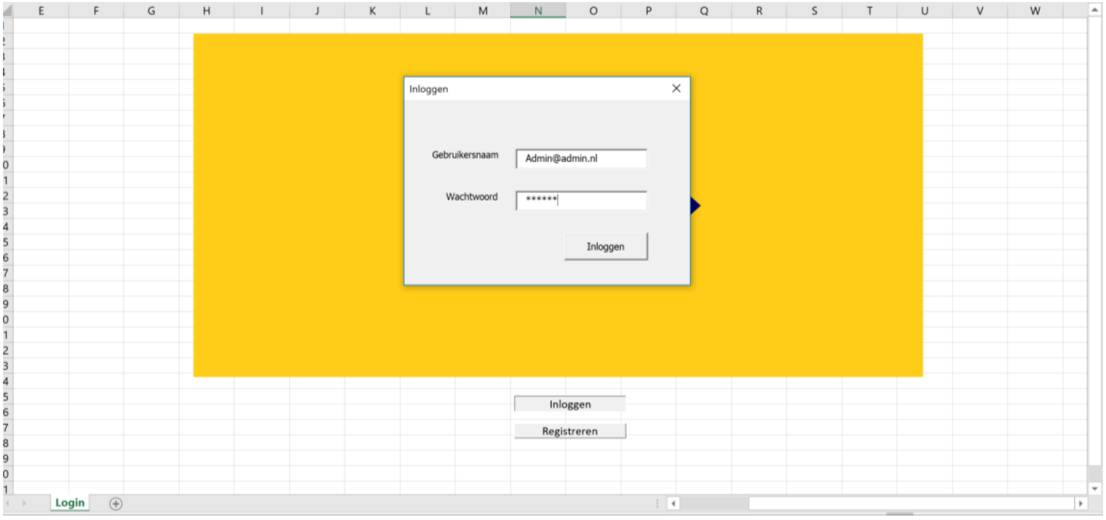


Fig. 8: Screenshot of REMS 1.0 – Quick Scan, front page


Inloggen

Gebruikersnaam Admin@admin.nl

Wachtwoord ******

Fig. 9: Screenshot of REMS 1.0 – Quick Scan, login page

REMS 1.0 Maturity Scan

The set of questions of the Maturity Scan is more comprehensive than the Quick Scan, but the layout is still the same.

The questions can be answered with a tick box and the results are found in the results-sheet.

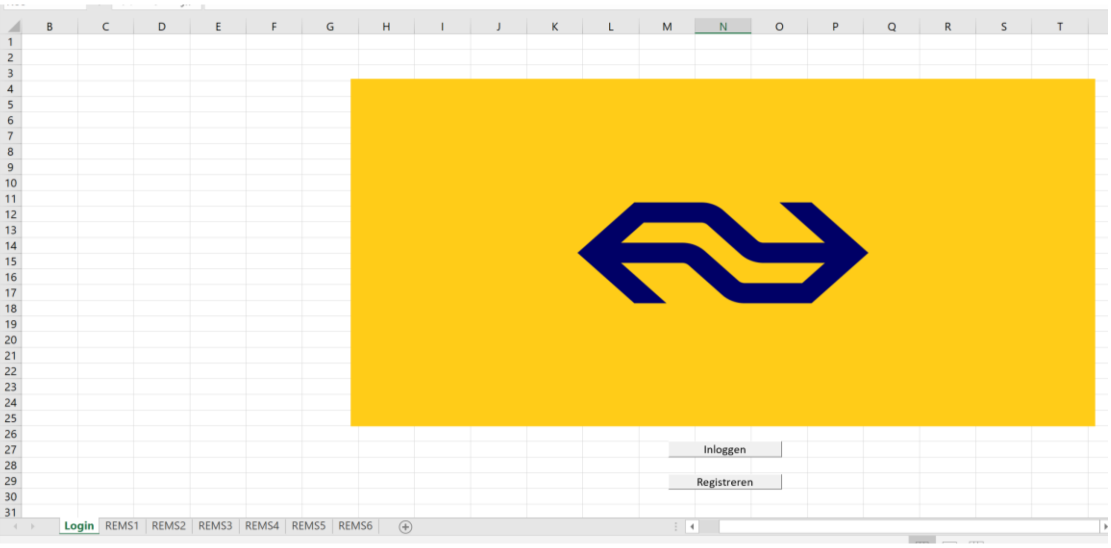
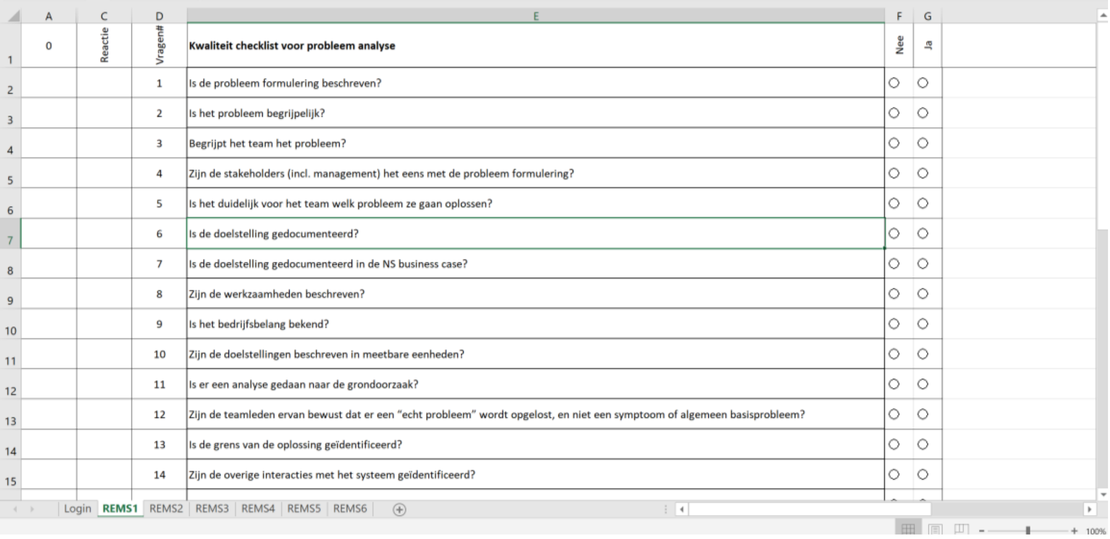


Fig. 10: Screenshot of REMS 1.0 – Maturity Scan, front page



0	Reactie	Vragen#	Kwaliteit checklist voor probleem analyse	Nee	Ja
		1	Is de probleem formulering beschreven?	○	○
		2	Is het probleem begrijpelijk?	○	○
		3	Begrijpt het team het probleem?	○	○
		4	Zijn de stakeholders (incl. management) het eens met de probleem formulering?	○	○
		5	Is het duidelijk voor het team welk probleem ze gaan oplossen?	○	○
		6	Is de doelstelling gedocumenteerd?	○	○
		7	Is de doelstelling gedocumenteerd in de NS business case?	○	○
		8	Zijn de werkzaamheden beschreven?	○	○
		9	Is het bedrijfsbelang bekend?	○	○
		10	Zijn de doelstellingen beschreven in meetbare eenheden?	○	○
		11	Is er een analyse gedaan naar de grondoorzaak?	○	○
		12	Zijn de teamleden ervan bewust dat er een “echt probleem” wordt opgelost, en niet een symptoom of algemeen basisprobleem?	○	○
		13	Is de grens van de oplossing geïdentificeerd?	○	○
		14	Zijn de overige interacties met het systeem geïdentificeerd?	○	○

Fig. 11: Screenshot of REMS 1.0, first screen “REMS1” – Maturity Scan

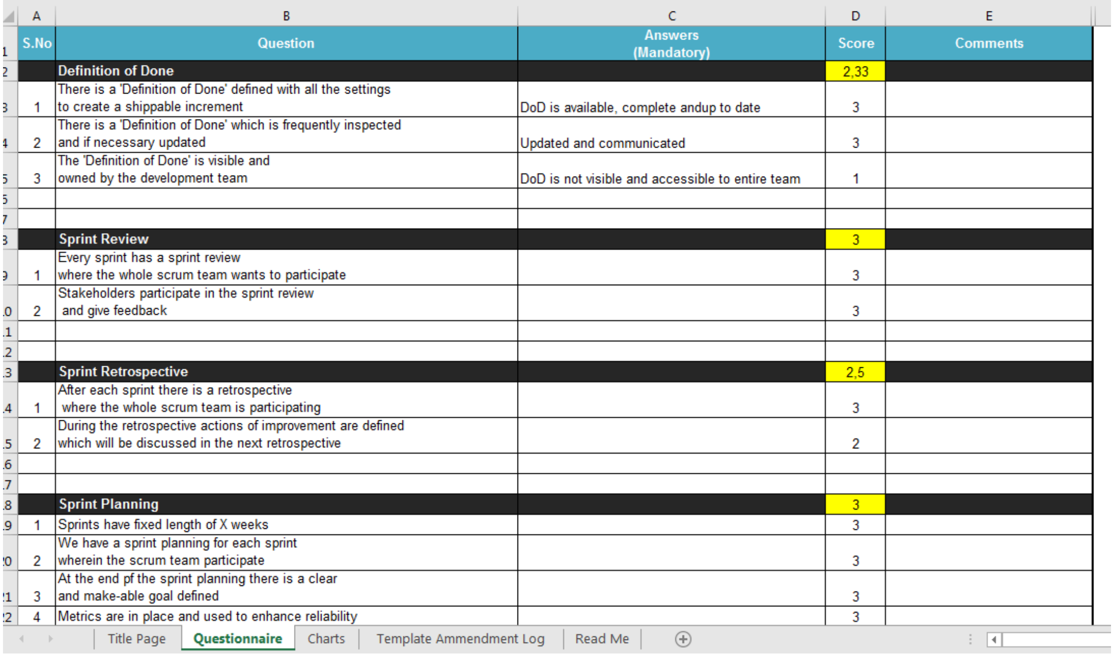
4.1.3. REMS 2.0

REMS 2.0 is created because the international bank needed a set for their Agile way of Working that has a linkage to their requirements. The full list of questions can be found in Appendix 5. REMS 2.0 is an extended set of metrics that can be used for companies that work mostly with the Scrum methodology. The assessment questions for REMS 2.0 are in the below table.

Subject	Amount of Questions
Definition of Done	3 Questions
Sprint Review	2 Questions
Sprint Retrospective	2 Questions
Sprint Planning	4 Questions
Daily Scrum	3 Questions
Development Process	4 Questions
Product Backlog & Product Owner	12 Questions
Cross-Functional & Happy Team	13 Questions
User Stories	4 Questions
Scrum of Scrum	6 Questions

Table 12: List of assessment questions REMS 2.0

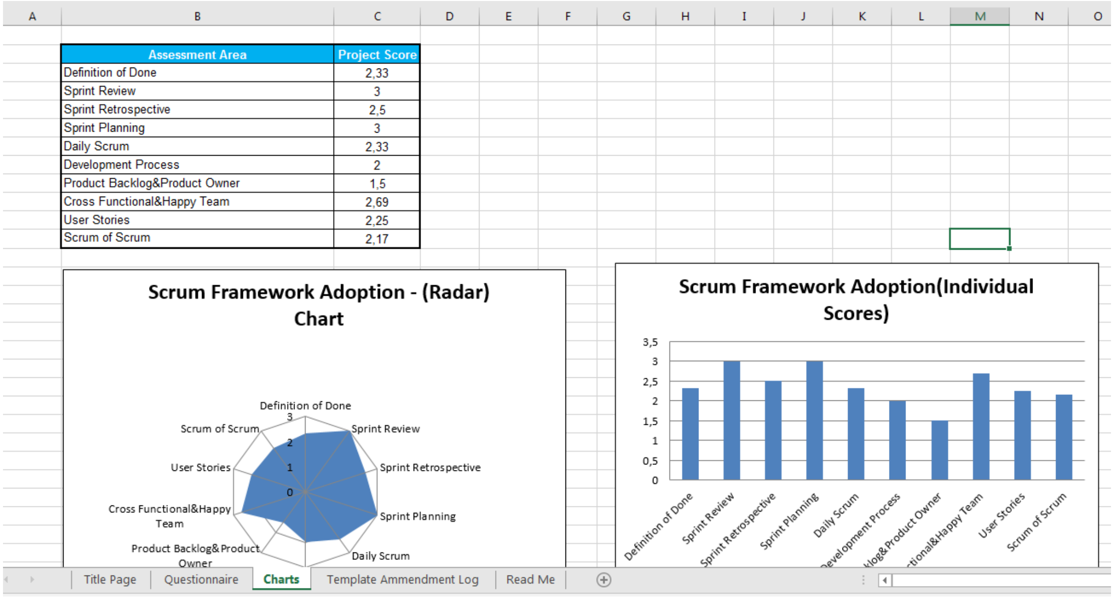
The assessment of REMS 2.0 is also created in Excel. Assessing is done with value scores from zero to three. Each question can score a minimum of 0 and a maximum of 3.



S.No	Question	Answers (Mandatory)	Score	Comments
	Definition of Done		2,33	
1	There is a 'Definition of Done' defined with all the settings to create a shippable increment	DoD is available, complete andup to date	3	
2	There is a 'Definition of Done' which is frequently inspected and if necessary updated	Updated and communicated	3	
3	The 'Definition of Done' is visible and owned by the development team	DoD is not visible and accessible to entire team	1	
	Sprint Review		3	
1	Every sprint has a sprint review where the whole scrum team wants to participate		3	
2	Stakeholders participate in the sprint review and give feedback		3	
	Sprint Retrospective		2,5	
1	After each sprint there is a retrospective where the whole scrum team is participating		3	
2	During the retrospective actions of improvement are defined which will be discussed in the next retrospective		2	
	Sprint Planning		3	
1	Sprints have fixed length of X weeks		3	
2	We have a sprint planning for each sprint wherein the scrum team participate		3	
3	At the end pf the sprint planning there is a clear and make-able goal defined		3	
4	Metrics are in place and used to enhance reliability		3	

Fig. 12: Screenshot of the REMS 2.0 Assessment

The total score will be consolidated in the Chart tab to show the result. The result is shown in a spider/radar chart and a bar chart. A trend can be formed when REMS is used repeatedly over a set of period.



Assessment Area	Project Score
Definition of Done	2,33
Sprint Review	3
Sprint Retrospective	2,5
Sprint Planning	3
Daily Scrum	2,33
Development Process	2
Product Backlog&Product Owner	1,5
Cross Functional&Happy Team	2,69
User Stories	2,25
Scrum of Scrum	2,17

Fig. 13: Screenshot of the REMS 2.0 charts

5. Discussion

In the discussion section, the research questions are answered with the knowledge gained from the literature review and the case study. REMS was used at three different companies, and REMS was studied on different projects.

In this chapter, the research questions will be discussed with the help of the interview input and the literature review. The research question of this study was to design a lightweight method to measure the quality of requirements engineering for software development in complex IT environments.

The research question will be answered with the following sub-questions in the following section:

1. What are the main challenges in requirements engineering?
2. Which methods can be used to measure requirements engineering?
3. What are the criteria for lightweight requirements engineering?
4. What are the KPI's in requirements engineering?
5. How can a conceptual model be implemented at the business case?
6. How can we evaluate the implemented model?

5.1 Lightweight method to measure the quality of requirements engineering for custom software development in complex IT environments

In traditional software development methods, the business analyst collects requirements through elicitation, analysis, verification, and validation. In business analysis, the critical task is to structure the requirements of the stakeholders. The business analyst invites stakeholders to give an account of the project's requirements. Depending on the software development method (e.g., Agile or Waterfall), it can vary how requirements engineering is processed. When the requirements are gathered, requirements can be identified as important by the stakeholders. The task of the business analyst is to discover what is necessary below the surface, and it is needed to find the deeper meanings by gathering facts, integrate data from different sources, find common ground between stakeholders and identify actual needs. To find a lightweight method, REMS was focused on the Agile way of working.

According to R. Tracey, "*complexity*" refers to a particular dynamic or movement in time that is paradoxically stable and unstable, predictable and unpredictable, known and unknown, certain and uncertain, all at the same time (Waldrop, 1992; Goodwin, 1994; Kauffman, 1995). In Fig. 14, the Stacey matrix shows the complexity that is defined by R. Stacey.

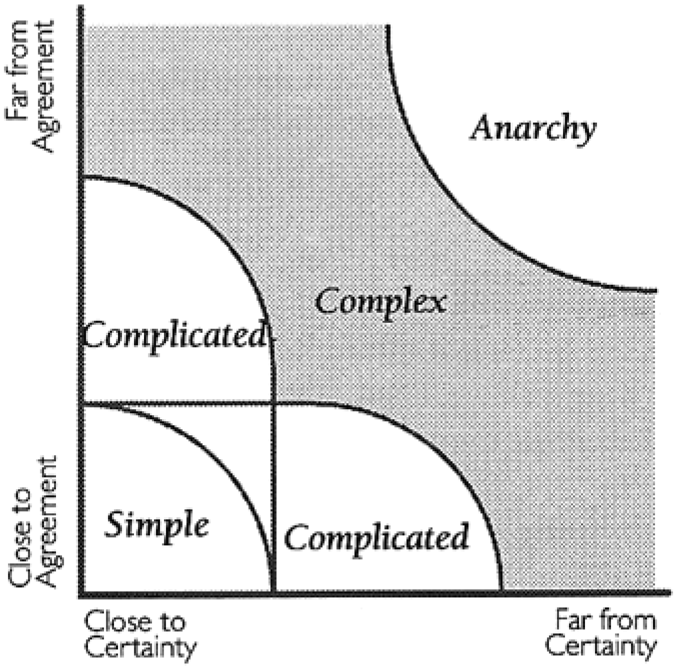


Fig. 14 - Stacey matrix on complexity

The definition of quality changes over time. The best time to use traditional methods for projects is where requirements change less than 1% per month (Boehm, 2002). When requirements change more often, an alternative would be to use iterative methods. The roadmap to develop quality requirements (on time, on budget, and does it meet the customers' real needs) must encounter the "requirements" of the customer, and not the "needs". The "nice-to-have" requirements are not the main goal whenever the development starts, and therefore the "must-have" requirements should be the focus.

When REMS 1.0 was created, most experts and interviewees were finding the maturity scan too long, not relevant and often not useful. In a practical sense, they did not want to use the REMS on a regular basis. The management, on the other hand, did find the REMS useful and they were very interested in the outcome of the results.

5.2 Main challenges in requirements engineering

Projects evolve rapidly, and new developments in IT projects may cause changes in the requirement. Requirements engineering is not limited to one phase of the project, especially working with the Agile method. Agile methods will stay in software development for the coming period, but they will not take over the traditional methods to gather requirements. It will be an evolution on gathering requirements. "Traditional software engineering can be enriched by paying attention to new ideas springing up from the field" (Glass, 2001).

Requirements Engineering authors Dorfman and Thayer [1990] defined Requirements Engineering as:

"1. A software capability needed by the user to solve a problem to achieve an objective.

2. A software capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documentation."

There are different types of requirements. On each level, we can divide them in functional and non-functional requirements. Functional requirements can be:

- Business requirements (what do companies want to achieve with the system)

- User requirements (what is the goal or task that the users have with the system)

- System requirements (what are the requirements or restraints of the system to achieve the business and user requirements)

Non-functional requirements can be measured with ISO/IEC 25010:2011[12]. This defines the following:

1. "A quality in use model composed of five characteristics (some of which are further subdivided into sub characteristics) that relate to the outcome of interaction when a product is used in a particular context of use. This system model is applicable to the complete human-computer system, including both computer systems in use and software products in use."

2. "A product quality model composed of eight characteristics (which are further subdivided into sub characteristics) that relate to static properties of software and dynamic properties of the computer system. The model is applicable to both computer systems and software products."

To have a streamlined requirement engineering process, the development of requirements often has the iterative process of the following activities: elicitation, analysis, specification, validation and realization. See figure 15.

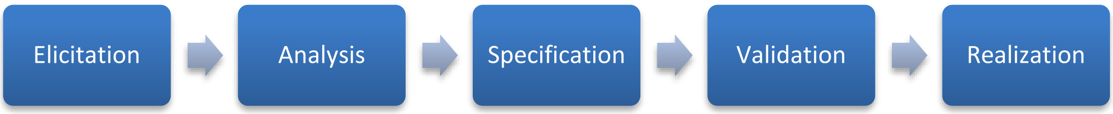


Figure 15: Activities of elicitation, analysis, specification, validation and realization

Getting the right requirements is considered as a vital but difficult part of software development projects [C. Jones,1996]. During the elicitation process, the requirements are gathered from the stakeholders. Gathering requirements can be done by techniques such as analyzing documentation or an existing system, interviews with stakeholders, workshops, observations, brainstorming, task analysis, prototyping or a combination of the techniques.

In the Analysis process, the requirements analyst will analyze these formed requirements with techniques like keeping a checklist, divide requirements into categories, or an interaction matrix (Kotonya e.a. 1998).

[12] https://www.iso.org/standard/35733.html

The results are captured and specified during the specification process, with techniques like scope models, structure models or system context diagrams. User requirements are often captured with Use Cases and have a business perspective. For prioritizing the requirements, techniques like MoSCoW, voting, content prioritizing or estimation are used.

In the validation process, the results of the specification are validated by checking the requirements. And in the Realization process, the requirements are put in practice.

The most common factors that caused projects to be "challenged" are according to the Standish Group [1994]:

1. Lack of user input (13%)
2. Incomplete requirements and specifications (12%)
3. Changing requirements and specifications (12%)

5.3 Criteria for lightweight requirements engineering

The requirement for creating a conceptual model was the criteria "lightweight assessment". It is therefore important to define "lightweight". In this study, "lightweight" means learning *what* needs to be measured and *where* to find this data to measure. Measurable units are broken down to pieces, and key metrics are identified. The tool for collecting data is also based on the essentials to avoid the details.

Most of the Agile practices are nothing new[13]. Incremental and iterative techniques focusing on breaking the development cycle into pieces evolved from the Waterfall model (Beck, 1999a), where the Waterfall method repeats itself until the end of the SDLC. However, requirements gathering in Agile Software Development Methods allow incrementing smaller functionalities throughout the SDLC, whereas projects in the Waterfall method will gather requirements in the beginning. To narrow the research, the scope will be on the Scrum Framework.

Scrum Framework

The Scrum Framework is an Agile method, created for effective team collaboration and complex products. Information about the Scrum Framework can be found in the Scrum Glossary[14] and the Scrum Guide[15], where the Scrum related terms are explained. Examples are the roles of the Scrum Team, Scrum Events, and Scrum Artifacts.

5.4 KPI's in requirements engineering

As organizations target moving to the next level of project performance, measuring their requirements engineering is a logical step. Some researched companies used CMMI as a metric tool for organizational growth. IT projects have not met their full potential yet and knowing where to improve their bottlenecks will allow an organization to forecast potential opportunities or threats.

[13] Cockburn and Highsmith, 2001

[14] https://www.scrum.org/resources/scrum-glossary?gclid=CjwKCAjw96fkBRA2EiwAKZjFTSeh-U7USiadNRgFAjPD4-MAjgw1-08t-KnG8ZLAZd0cOp9ruM1cgxoCLokQAvD_BwE

[15] https://www.scrum.org/resources/scrum-guide?gclid=CjwKCAjw96fkBRA2EiwAKZjFTZuIL2jQK4cGLdouEDH5_bQr-HX6_CA7gES3Sz_2Ai3sim369MMdyRoCcaIQAvD_BwE

Performance metrics translate business objectives to a set of operational metrics. The critical element of metrics are KPIs is to find quantifiable, understandable and meaningful measurements, that reflect the critical success factors of an organization. KPI's measure the success of an organization in a particular activity. With the KPI's, the most important metrics to the team and business are taken into account. The requirements KPI's will answer the question "Are the requirements effective?". KPI's are useful for management. When management know what the current situation is, it is made more accessible to steer in the right direction.

While changing requirements are not desirable, it is still part of software development. Indecision around requirements will probably never leave, so it is better to be prepared for the changes. The KPI's will figure out how well the team is at a consistent pace with the changing requirements. Effective requirements can be most effective measured at the beginning and end of an agile development cycle.

The following table visualizes possible KPI's that could be used to find meaningful measurements.

Project tracking	Manage tasks and bugs
Source control	Manage code and collaboration
Continuous integration	Generate builds and run tests
Deployment tools	Move code across environments
Application monitoring	Ensure everything is working

Table 13 - KPI's

5.5 Implementation of the Maturity Scan

The implementation of the Maturity Scan was done at three companies and in periods between 6 months and 12 months. Some organizations work with the traditional software development method and some work more Agile. It was not possible to implement the Maturity Scan without understanding the need of their IT projects.

In projects, customers often change their requirements, and this may affect the design and coding. When a project uses a Waterfall method, it will have a higher risk of changing codes than an Agile approach. During a Waterfall project, requirements are gathered in the beginning in the process, and the system will be built in another process. Changes in any requirement lead to an adjustment in the development. While using Agile methods, the requirements will evolve as the project proceeds. Making decisions in a later stage of the project will decrease the maximum amount of change and the risk that coding needs to be changed too frequently.

5.6 Evaluation of the Maturity Scan

In the business case, the results of a literature study are shown for the research question "How to design a method to measure the quality of requirements engineering for software development in complex IT environments?". A qualitative approach was used for the business case. Different measurement studies were analyzed, and a new method is proposed. Aside from the literature study, interviews were held with 15 employees and surveys were conducted by 32 employees. The result of the case study would lead to an advisory report and a presentation to interested parties with recommendations.

Overview
Assessment questions were carefully selected to cover the most general issues, so companies can choose from them to create a tailored assessment that works for them. The assumption was made that every company and project is different, therefore there was a need for a generic set of tools.

While developing the conceptual model, not all metric methods were suited. Various metric methods were considered, but not all were applied when creating REMS. After the pilot was done at the international bank, the Maturity Scan was evolved to use on other Scrum teams during their IT projects.
To have the most accurate result, the desirable time to start using the Maturity Scan is as early on the project as possible. The Maturity Scan is used during the early stages to the middle of the projects, and the scrum team will see a trend in their way of working. As a result, Scrum teams can improve their way of working and the quality in their delivery.

People involved
When evaluating the assessment model, the satisfaction of the user is measured. Satisfaction in general terms is defined as a positive affective response from a user, based on his or her experience with a system (Oliver, 1980). Satisfaction is a link between user expectation before the experience, and whether the actual experience meets or exceeds that expectation (Oliver, 1980). Studies have shown that satisfaction, built on prior experience, is related to future usage intention (Bloemer & Kasper, 1995; Heinrichs, Lim, Lim & Spangenberg, 2007; Oliver, 1980). The evaluation of the method is done with an anonymous survey and personal interviews.

Theoretically, implementing the assessment is not difficult. The management of the company that uses the assessment command their employees to use the REMS, and then the results can be gathered and analyzed.

However, the implementation of the assessment was not easy in practice. The main reason for the difficulty was the mindset of the people. The satisfaction of the REMS was not experienced on a positive level. Even though awareness was created by the management beforehand, toward the project leaders, scrum masters and the scrum teams. There was often a great division in the perspective of the REMS. Some people saw the benefits of the new development, but a lot of people were resistant of the extra work, the "feeling" of being checked and monitored by the management and the overall objective of the REMS was not understood by the teams.

Many people were involved in this thesis. People who were involved in the Maturity Scan had different opinions on the Maturity Scan. The reason is that they have another perspective on it. Some were positive, and some were negative.

The teams often felt they needed to do extra work for the management, on top of the usual business as usual (BAU). Teams also felt there was a lack of communication, and the teams did not know where the results will be kept and what the management will do with the results. Most people who are involved in the process were not pleased with the Requirements Engineering Maturity Scan. The people who used the Maturity Scan as a measurement of their work were not satisfied they had to apply an extra step in their daily work. Thus, they did not see the added value of the metric system.

The Management, on the other hand, saw the Maturity Scan as a useful tool. They can see how their employees are performing and how the scrum teams are improving themselves.

Some involved people saw the positive effect of measuring requirements. They are mostly willing to participate in the research and helping with interviews or surveys.

Metrics

According to a mathematical queuing theory, work processes work the best when small batches of work move through a system at a steady rate. Working in small, frequent iterations, will promote efficiency in the development process. Having metrics will help the business because a moment is created for reflection. Reasons for metrics can be to monitor requirements, have a better understanding of the project, improve the way of working, clarify impediments, expand transparency or enabling communication.

It was important to create useful metrics and not to create Vanity metrics. Vanity metrics are measurements but are not meaningful and easily manipulated or changed. The Maturity Scan is created to help create value to the business, to make decisions, and to help growth. Having requirements metrics allow the people involved to have a better understanding of the requirements, project tracking, source control, the build system, system monitoring and the team way of working. The output of the metrics should be an advantage for the project. Information from the metrics should measure the outcome of the teams and create a lean, minimal set of metrics that serve the purpose.

Starting with the metrics, there is a baseline needed. The baseline is the point where the measurements can lever up. When there are more results from the metrics, a trend is created. With the trend, it is made easier to steer, forecast certain activities or change the performances.

The metrics show an insight of the requirements of the teams and the way of working. Taking the assessment should be the responsibility of the team to obtain tracking and to invite communication. Interpreting the metrics can be done by the team and the management. The metrics can show how well the team understands the project, how fast the team is moving and how consistent the team is.

Process

The process of defining a set of requirements questions allowed evolvements in the case study. First, there was a need for a lightweight assessment on requirements, and then it needed expansion on the assessment to come to a more comprehensive assessment. The elaboration on the first set is a step closer to a more generic assessment where teams can tailor their own assessments. Tailoring the assessment will keep the assessment lightweight, as not all the questions will be of value to all teams. The REMS process is to identify assessment questions that matter to the team, set up the assessment and chart the activities. The result of plotting the activities will create a trend, and this can be analyzed for future references.

In this study, the focus is on conceptual integrity. The purpose of the assessment is to find the metrics on requirements to improve the quality of requirements. This means that the perceived integrity is not put in scope. Some companies have a more hierarchal structure in their business, which also plays a role in their decision of using REMS.

Conceptual integrity is judged by developers, while perceived integrity is judged by the customer's perception. When the conceptual integrity of the requirement is high, the functionalities have met a reliable, running system that fulfills its purpose. When the perceived integrity is high, the product of the requirement is of quality and value.

Results

The choice was made to visualize the results in charts. See Fig. 11. They function as an external memory aid by using arrangements that organize information in meaningful ways (Larkin & Simon, 1987). The teams that used the assessment have a trend in their way of working, and the teams that did not use the assessment did not have a track of their way of working. Tracking requirements can help with self-reflection and as a team.

When gathering requirements, the domain knowledge of the requirements engineer is not taken into account. This has no impact on the assessment. The result of the Maturity Scan is that the company needs a good Change Management Plan. The company should convince the employees to work with it and see the added value. Fig. 16 visualizes the cost of change at various phases of the SDLC.

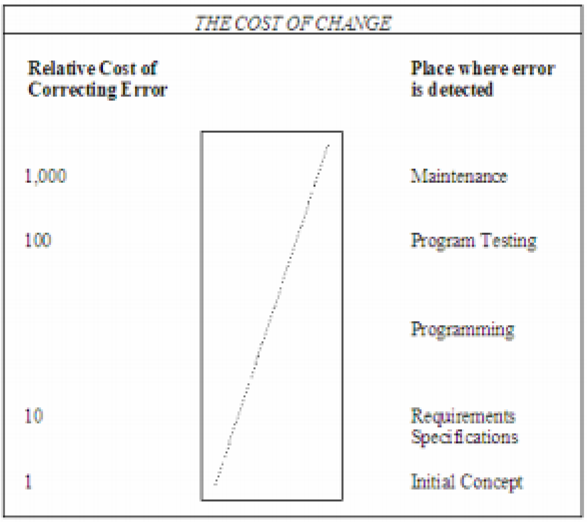


Figure 16: cost of change at various phases of the SDLC[16]

[16] Treasury Board of Canada Secretariat, “Systems Under Development (Audit Guide)”, Retrieved 1 March 2010, http://www.tbs-sct.gc.ca/pubs_pol/dcgpubs/TB_H4/systemssystemes03_e.asp

6. Conclusion and recommendations

This chapter will describe the conclusion and recommendations that were found during the project.

6.1 Conclusions

The study shows that large companies with complex IT environments often lack requirements metrics. There is not a standard methodology on gathering requirements and to measure requirements. The Maturity Scan is implemented at three different companies. Although the companies differ in size, area or environment, there are still some similarities. Sometimes, companies work in "Agile-in-name-only", where they combine the Waterfall method with the Agile methodology. Some projects are more Agile-oriented than other projects, even within one company.

The most accepted Agile methodology at the researched companies was the Scrum Framework. All the people who were interviewed and the people who filled in the survey were familiar with the Scrum methodology, or at least know some of its terminologies.

Gathering requirements with the Waterfall model allow stakeholders to think before they act, but in practice, projects have shown that users often change their minds. When designing a lightweight requirement assessment, there is no 'one-way fits all' to do so. Every company and project require different outcomes and parts of it must be tailored. The metrics itself may vary from company to company, and even from project to project. Investing in a tailormade metrics system can cost a lot, and it might take much time. The choice of investment is up to the company, and this can vary.

When the choice is to invest in a Maturity Scan, it is essential to have a lightweight assessment, because measuring requirements engineering is an extra step in the process which is not per definition a mandatory step to have working software. Most negative feedback from the involved people was the long, extra process of the assessment. Also, this extra step might not directly add more value to the company. The assessment is used to find requirement metrics on improving the way of working, team contentment and the general best practices. During the study, it was clear that when the Maturity Scan was lightweight and easy to use, there was less resistance from the project teams.

One way of making sure people use REMS in projects is to have the Maturity Scan pushed by the management to have it implemented in the software development process, i.e. the "top-down" approach. One of the researched companies used this approach but implementing a new system to projects with a top-down approach may trigger negative feedback. Project teams were very resistant to the new change, and they were often not willing to fill in REMS.

In the end, even though there were mixed tensions for REMS, doing the Maturity Scan assessment is still a human interaction. Having the right people in the metrics team is very important. The metrics team should be there to inform Scrum teams about the Maturity Scan and support teams to fill in REMS. Scrum teams were not always enthusiastic about REMS, so there is also a need for improvement to effectively implement REMS in projects. Starting a new metric system will take time, and people need to be educated about it.

To have people *wanting* to do a Maturity Scan, because they see value in it, Maturity Scans must be kept lightweight, agile, and easy to maintain. Knowledge transfer of performing REMS should be made as comfortable as possible. This ensures the quality of performing a Maturity Scan.

REMS will not change teams drastically, and it will not be a revolution for a company. REMS will be a conversation starter for structural project management improvements, where the assessment results are transparent to the team and where the team can decide to do anything with it. Multiple teams can use the tool which will lead to an improvement in collaboration as there is a common set of metrics to speak to same language. Forecasts can be made easier, and trends can be analyzed. REMS can be an asset to create a valuable assessment with an own set of metrics that can be used throughout the project.

Validity considerations

To achieve a better interpretation between business and IT, it is important to have clear, concise and complete requirements. The quality requirements lead to valuable improvements in IT projects, like risk reduction. Early identification of requirements engineering metrics will enhance IT projects in predictability and adjustability. The overview of where the project is at, and what the future status will be, allow project members to act upon it.

With a better time-management, it is easier to reduce costs in IT projects. Team members can focus on their core tasks. The success of REMS can be measured with Net costs, the speed of delivery, quality of the requirements, the reliability of the estimates, or even the possibility of automating certain repetitive tasks that are slowing down the team.

6.2 Limitation

There are limitations to the Maturity Scan that was created during the thesis. While the research study has a focus on companies with Agile expertise, some teams only work “Agile in name only”. For example, teams work with the traditional development method, but they added a daily meetup/scrum, and they call their way of working “Agile”. Companies should be clear of their Agile way of working and in what extent they use Agile or Scrum.

First, the Maturity Scan is only implemented and tested at three companies that are based in the Netherlands. It does not give a solid viewpoint of the whole context to create a general Maturity Scan that can be used in all complex IT environments. Also, the way the company worked at that point is only a snapshot of that period and specific projects that participated. REMS will add more value when the trend is shown and not only a snapshot. There is a possibility that some companies and projects evolve in their way of working. That means the researched content is not representable for that team or company anymore.

Second, the Maturity Scan has only been validated by the people who are interviewed and who filled in the survey for this study. It may not serve a larger group of people. Even though there were numerous people who gave feedback on the Maturity Scan while creating the Assessment, and many teams who worked with it, there is still a chance the research is biased from a certain point of view.

Third, the study used a particular set of methods, which may not be typical or suitable for other companies. IT projects have been around for decades and requirements engineering is a well-known term. Even though the study already made an elaboration in the assessment questions, from requirements engineering to a more Agile approach, it still will not cover a “generic lightweight assessment”. There is still room for optimizing the requirements engineering process. The used methods for this Maturity Scan may not have the most suitable content for a general measurement, but these are international best practices, used by many companies and proven methods. Moreover, the participated companies felt at ease to work with the chosen methods, and the reason why the chosen methods are used is that the human factor is also something that is taken in account for this study.

Finally, there was no guidance for the involved people on implementing the Maturity Scan. There was no governance in doing the assessment. The objective of the assessment was not communicated enough. When the Maturity Scan was implemented at the three companies, there was a certain resistance among some people and some projects. They did not know what to do with the Maturity Scan and why they needed to assess their requirements. While some people did have a positive attitude toward the Maturity Scan, adding an extra step to their way of working was not appreciated. People felt they had extra work added and they did not see the added value of measuring the requirements engineering process.

In general, teams often did not like the idea of being measured, and there was often a lack of communication with/toward the involved people. The Maturity Scan was only created with a particular group of people, the pilot, and the rest of the teams were not involved. Therefore, the teams that were not involved had no idea of a new metric system in their way of working. Also, when the Maturity Scan was implemented there was limited communication toward the people. The teams "just have to" fill in the assessment and the output was analyzed.

This might have influenced the development and validation of the Maturity Scan. Limitations set by the resistance of the people in adapting the scan caused the lack of participants who could have valuable input for the study. Above all, the people who were interviewed, and the people who did the survey may have a certain background that may have cause a disruptive perspective while creating the Maturity Scan. Even the validation of the Maturity Scan can be influenced by the people who filled out the survey. Experienced IT people often have knowledge on the way of working in projects, team-work, Scrum methodology, Waterfall methodology, their own involvements, the exposure of management or even previous participation of Maturity Scans.

6.3 Recommendations for future work

It was a challenge to implement a new routine for projects. Considering the limitations of this study, there are some recommendations for future work.

Overview

When implementing a change in the company, a very important issue to address is the change management department. The change management department often knows how the company is currently working and how changes are implemented throughout the company. Also, the change management department often has contacts of other departments or have recommendations on what to do best in that case. When there is no change management department, the recommendation is to find a department that works as a change management department.

While most of the teams that participated this thesis were working in some form of Agile project, the study was not always well received when the participants knew they might have to use REMS for a longer period. REMS does not work well with Agile projects, since there is already a regular check-up moment on the requirements. However, REMS can be used as a tool to measure the requirements whenever it is needed. REMS would have an added value in a Waterfall environment, where the requirements are not frequently reviewed, and REMS could give an insight for that.

People involved

Generally, people often do not like change or do not like to change. When a new method is created, it means that people must change *something* in their workforce. When a specialized department, like change management, can guide the people to change, it will help the people and the management to minimize any impact.

To create an effective and efficient implementation, it is recommended to have clear answers on the communication part. There was often a feeling that REMS is inefficient during the project because it was an extra step in the process and there was often a lack of commitment. It is crucial to communicate the reason behind measuring the requirements, and why improving the quality of requirements can help with the software development process. Teams should have a clear vision of the outcome of the measurement, and they should feel the need to participate in the new way of working. The teams should see the benefits of having quality requirements that it can improve the team spirit, the use of new technology and how the Maturity Scan can help them to move further.

Process

Assessments can be done during Scrum Retrospectives, and the trend of the assessments can be analyzed after a few times. It is recommended to set up a metrics team who are dedicated to creating the metrics, guide the team members doing the assessment and gather the results of the metrics. There is a need for governance, guidance, and structure. When there are questions, there should always be a metrics expert to help. Moreover, if companies need to extend the performance of the assessment, the company can get external resources to do the performance (outsourcing the assessment).

It is also recommended to have experienced stakeholders to tailor the assessment to have a more robust metric system to initiate the Maturity Scan. This will allow people to feel involved, and more willing to corporate. Furthermore, it creates awareness in teams, and there should be communication on both ends. Finally, have experienced members who know what the best way is to improve the metrics.

Metrics

It would be interesting to have one assessment that would measure the project tracking (manage tasks and bugs), source control (manage code and collaboration), continuous development (generate builds and run tests), development tools (move code across environments) and application monitoring (ensure everything is working).

A transformation of the REMS 2.0 metrics could be REMS 3.0. DevOps can be introduced in metrics. “DevOps is a set of software development practices that combines software development (Dev) and information technology operations (Ops) to shorten the systems development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives.” [Atlassian. April 2019] KPI’s can be metrics that measure the time of market product/services, business agility, delivery or productivity and metrics on operational service costs. Another evolvement is to measure Continuous Integration, Continuous Delivery or

Continuous Testing. There are different DevOps tools like Jenkins and Solarcube that allow measurements. KPI's need to be created and the results show the maturity of the deployment.

Technology

The current REMS is created in Excel. It would be interesting if the results are visualized in a dashboard that is running real-time. If the dashboard is real-time, it is possible to apply gamification to motivate the people.

Results

There are many possibilities for lightweight assessments in requirements. It is a shift of the mindset of the people, the availability of technology, the budget of a company and the willingness of management to push an assessment through. It is recommended to have a change management process with clear communication in future studies.

7. References

Study on large scale IT projects, 2012, McKinsey & Company and the University of Oxford.

Complexity and Management, fad or radical challenges to systems thinking? – Ralph D. Stacey, Douglas Griffin and Patricia Shaw, 2001

Arendsen, M., H.J.J. Cannegieter, A. Grund, P. Heck, S. de Klerk & J. Zandhuis (2010), *Succes met de requirements!*, 2^{e} herziene druk, Sdu, Den Haag.

Aydinli, O., E. Hendriks, J. Zandvliet (2013), *Smart requirements 2.0*, Sdu, Amsterdam.

IREB (2015), IREB Foundation syllabus, *Certified professional for requirements engineering* – Foundation level – Version 2.0, International Requirements Engineering Board.

Leffingwell, D., Widrig, D. (2007), *Manageing software requirements*, Pearson Education.

Alain Abran, James W. Moore; Pierre Bourque, Robert Dupuis, eds. (March 2005). *Guide to the software engineering body of knowledge* (2004 ed.).

Software requirements, second edition. Karl E. Wiegers, 2003.

Rapid Development: Taming Wild Software Schedules. McConnel, Steve 1996.

8. Appendices

Appendix 1: Interview setup

Interviews were conducted during the case studies. In this appendix the interview setup is described.

Standard email for invite:

> Dear colleague,
>
> My name is Jenny and I am an ICT in Business student from the Leiden University. Currently, I am working on a research about lightweight requirements engineering metrics. I would like to invite you for an interview of 60 minutes to deep dive in this topic. Your expertise would help a lot.
>
> Thank you in advance.
>
> Kind regards,
> Jenny Yung

Introduction:
Starting with a short personal introduction: name, study, reason why I conduct the interview, and what the goal is of the interview.

Explanation of the research outcome:
Interested in the requirements engineering at the company they work for.

Appendix 2: Key findings in interviews and survey

Interview questions for the experts – Before creating REMS 1.0:

1. Can you tell me a bit about your career background?
2. What is your experience with requirements engineering?
3. What is your experience in the Agile way of working?
4. Have you worked in a complex IT environment before?
5. How would you define a complex IT environment?
6. What is your experience with metrics in requirements engineering?
7. Should projects embed metrics in requirements engineering?
8. What are the challenges according to you, when we implement metrics in requirements engineering?
9. What is a good metric for the REMS?

Key findings of the experts:

	Expert Role	IT experience (years)	Complex IT environment	Metrics Requirements Engineering (summarized)	Challenge (summarized)
1	Business Analyst	5-10	3-8 projects	No experience but is very interested in the outcome.	-Technology -Quality of project
2	Business Analyst	5-10	8> projects	Experienced on a high level.	-People -Timing
3	Business Analyst	0-3	0-3 projects	No experience but is very interested in the outcome.	-Technology -Quality of project
4	Project Leader	10>	3-8 projects	Experienced on a high level.	-People -Purpose
5	Tester	5-10	3-8 projects	No experience but is very interested in the outcome.	-People -Purpose
6	Tester	0-3	0-3 projects	Experienced on a high level.	-People -Quality of project
7	Technical Architect	10>	3-8 projects	No experience but is very interested in the outcome.	-Timing -Quality of project
8	Enterprise Architect	10>	3-8 projects	No experience but is very interested in the outcome.	-Purpose -People
9	Software Developer	5-10	3-8 projects	Experienced on a high level.	-Purpose -Quality of project
10	Software Developer	5-10	0-3 projects	No experience but is very interested in the outcome.	-People -Quality of project
11	Software Developer	10>	3-8 projects	No experience but is very interested in the outcome.	-Purpose -People
12	Scrum Master	5-10	8> projects	Experienced on a high level.	-Purpose -People
13	Scrum Master	5-10	8> projects	No experience but is very interested in the outcome.	-Purpose -Quality of project

Survey questions before creating REMS 1.0:

1. What is your position?
2. What is your experience with requirements engineering?
3. What are the challenges within your IT projects?
4. What is the importance of requirements Engineering?
5. When is requirements engineering needed during a project?

Key findings of the experts:

	Position	**RE experience**	**IT challenges**	**Importance of RE**	**When RE is needed**
	Information analyst (73.91%)	10+ years (47.83%)	Communication with stakeholders (60.87%)	Very Important (78.26%)	During the project like the Agile method (43.48%)
	Business Analyst (17.39%)	5 – 10 years (34.78%)	Too many changes in requirements (39.13%)	Important (13.04%)	Whenever it is needed, like the Lean method (26.09%)
	Software developer (4.35%)	2 – 5 years (8.7%)	Inconsistency Business & IT (34.78%)	A little important (4.35%)	Within the project, like the Waterfall method (13.04%)
	Others (4.35%)	0 – 2 years (8.7%)	Hardware problems (30.43%)	Not important (0%)	Other (17.39%)

Survey questions for the experts – After the creation of REMS 1.0:

1. Can you tell me a bit about your career background?
2. What is your experience with requirements engineering?
3. What is your experience in the Agile way of working?
4. What is your experience with metrics in requirements engineering?
5. What do you think of REMS?
6. What is good about REMS?
7. What is bad about REMS?
8. What would you improve about REMS?

Key findings of the experts:

Expert Role	IT experience (years)	Agile way of working	REMS	Feedback
Business Analyst	5-10	Agile experienced	Positive	Could be nice to experiment with.
Business Analyst	5-10	Agile experienced	Positive	Could be nice to experiment with.
Business Analyst	0-3	Agile experienced	Positive	Could be nice to experiment with.
Project Leader	10>	Agile experienced	Positive	Could be nice to experiment with.
Tester	0-3	Agile experienced	Positive	Could be nice to experiment with.
Software Developer	5-10	Agile experienced	Neutral	Could be nice to experiment with.
Software Developer	5-10	Agile experienced	Positive	Could be nice to experiment with.
Scrum Master	5-10	Agile experienced	Neutral	Not interested in trying.

Interview questions for the experts – Before creating REMS 2.0:

1. Can you tell me a bit about your career background?
2. What is your experience with requirements engineering?
3. What is your experience in the Agile way of working?
4. What is your experience with metrics in requirements engineering?
5. What do you think of REMS?
6. What is good about REMS?
7. What is bad about REMS?
8. What would you improve about REMS?

Key findings of the experts:

Expert Role	IT experience (years)	REMS experience	Feedback
Business Analyst	10>	Nice tool. Good initiative to measure requirements.	REMS is too long. It should be shorter and easier to use.
Business Analyst	5-10	Good initiative to measure requirements.	REMS should be more user-friendly.
Business Analyst	5-10	Not useful for his project.	REMS should be easier to use.
Project Leader	10>	Good initiative to measure requirements.	REMS is too long. It should be shorter and easier to use.
Tester	5-10	Not useful for his project.	REMS is out of scope.
Software Developer	0-3	Not useful for his project.	REMS is too long. It should be shorter and easier to use.
Software Developer	10>	Good initiative to measure requirements.	REMS should be more user-friendly.
Scrum Master	10>	Good initiative to measure requirements.	REMS is too long. It should be shorter and easier to use.
Scrum Master	5-10	Good initiative to measure requirements.	REMS should be easier to use.

Interview questions for the experts – After the creation of REMS 2.0:

1. Can you tell me a bit about your career background?
2. What is your experience with requirements engineering?
3. What is your experience in the Agile way of working?
4. What is your experience with metrics in requirements engineering?
5. What do you think of REMS?
6. What is good about REMS?
7. What is bad about REMS?
8. What would you improve about REMS?

Key findings of the experts:

Expert Role	**IT experience (years)**	**REMS experience**	**Feedback**
Business Analyst	10>	Nice improvement, but he prefers not using it for his requirements.	-
Business Analyst	5-10	Good improvement. He likes it better like this.	Not sure if he will use this daily, but it is a nice tool.
Business Analyst	5-10	Good improvement. He likes it better like this.	Not sure if he will use this daily, but it is a nice tool.
Project Leader	10>	Good improvement, but not going to use it for requirements.	-
Tester	5-10	Nice improvement, but still not applicable for his project.	Not applicable.
Software Developer	0-3	Good improvement, but still not applicable for his project.	-
Software Developer	10>	Good improvement. REMS is shorter now.	Not applicable.
Scrum Master	10>	Good improvement. REMS is shorter now.	Not sure if he will use this daily, but it is a nice tool.
Scrum Master	5-10	Good improvement. REMS is easier to use now.	-

Appendix 3: REMS 1.0-NS Document

NS has its own process and its own way of working. The project document with the process is shown in this Appendix.

NS Case study

Requirements Engineering Maturity Scan

“Kwaliteit meten in Requirements Engineering”

Jennifer Yung
Afstudeerder
December 2015

Inhoudsopgave

Introductie

Er zijn verschillende perspectieven voor kwaliteit. Filosofisch gezien wordt kwaliteit benaderd als absoluut en universeel herkenbaar. Bij een waarde-gerichte benadering wordt de prestatie gekoppeld aan de prijs, en bij een productgerichte benadering wordt kwaliteit als een meetbaar variabele beschouwd. Bij een productiegerichte benadering wordt er van kwaliteit gesproken wanneer de vastgelegde productiespecificatie is getoetst. Met de gebruikers-georiënteerde benadering wordt er gekeken naar de tevredenheid van de gebruiker.

Voor dit onderzoek (case study) wordt er een instrument ontwikkeld om de kwaliteit van Requirements Engineering te meten. Eerst wordt de huidige situatie in kaart gebracht, en vervolgens is het instrument gebaseerd op de gegevens die tijdens het onderzoek naar voren werden gebracht. Er werd een enquête rondgestuurd naar de NS medewerkers en interviews werden gehouden om de huidige situatie beter te begrijpen.

Voor dit document wordt de literatuur van onder andere de boeken “Smart requirements 2.0” (Aydinli, et al 2013), “Precies volgens plan!” (Hoogveld, et al., 2011), “Succes met de requirements!” (Arendsen, et al., 2010),”Managing Software Requirements” (D. Leffingwell, 2003) en de syllabus van de IREB Foundation (IREB, 2012, 2015) als referentiekader gebruikt.

Achtergrondinformatie

Het doel van het model is om NS medewerkers, met name (project)managers, ontwikkelaars en informatie/business analisten, te ondersteunen bij het meten van de kwaliteit van hun Requirements Engineering. Er zijn verschillende eisen en methodieken gebruikt voor het ontwikkelen van het model. In dit hoofdstuk worden ze kort behandeld.

ISO 9216-norm

Om de niet-functionele requirements meetbaar te maken, wordt er met behulp van de ISO 9216-norm de requirements uitgewerkt. De ISO 9126-norm bestaat uit het model zelf, externe metrieken, interne metrieken en gebruikersgerichte metrieken.

Welke kwaliteitseigenschappen van belang zijn voor een specifiek systeem kan worden bepaald door meerdere technieken. Ongeacht de technieken, moet er eerst in kaart worden gebracht wie de stakeholders zijn.

Functionaliteit	**Betrouwbaarheid**	**Bruikbaarheid**
Compleetheid	Bedrijfszekerheid	Begrijpelijkheid
Juistheid	Bestendigheid	Leerbaarheid
Koppelbaarheid	Herstelbaarheid	Bedienbaarheid
Beveiligbaarheid	Naleving betrouwbaarheidseisen	Aantrekkelijkheid
Naleving functionaliteitseisen		Naleving bruikbaarheidseisen

Tabel 1 - De 27 kwaliteitseigenschappen van niet-functionele eisen volgens de ISO 916-1:2001 norm (1/2)

Efficiëntie	**Onderhoudbaarheid**	**Portabiliteit**
Snelheid	Analyseerbaarheid	Overzetbaarheid
Middelenbeslag	Wijzigbaarheid	Installeerbaarheid
Naleving efficiëntie-eisen	Stabiliteit	Beïnvloedbaarheid
	Testbaarheid	Vervangbaarheid
	Naleving onderhoudbaarheids-eisen	Naleving portabiliteitseisen

Tabel 2 - De 27 kwaliteitseigenschappen van niet-functionele eisen volgens de ISO 916-1:2001 norm (2/2)

Functionele en niet-functionele eisen

Businessrequirements, gebruikersrequirements en systeemrequirements kunnen worden onderscheiden door functionele en niet-functionele eisen. Hieronder volgen twee tabellen die gaan over de functionele en niet-functionele eisen. Deze eisen zijn belangrijk voor het vormen van het model. Er moet namelijk gekeken worden naar verschillende aspecten van eisen.

Functionele eisen:
Gedrag
Gegevens
Foutafhandeling
Dynamiek
Presentatie
Interfaces

Tabel 3 Voorbeelden van functionele eisen

Niet functionele eisen (ISO 9126):
Functionaliteit
Betrouwbaarheid
Bruikbaarheid
Efficiëntie
Onderhoudbaarheid
Portabiliteit

Tabel 4 Voorbeelden van niet-functionele eisen

Eisen per requirement

Requirements hebben bepaalde eisen. In de onderstaande lijst kun je de “eisen” vinden die horen bij requirements:

- Uniek identificeerbaar
- Atomair
- Eenduidig
- Vrij van implementatiedetails
- Traceerbaar
- Testbaar/verifieerbaar
- Voorzien van prioriteit

MoSCoW-methode
Met de MoSCoW-methode[17] wordt er gekeken naar de prioriteit in software engineering. De stakeholder bepaalt de prioriteit van bepaalde requirements. MoSCoW is een afkorting van het volgende (Van Vliet, 2008):

- **M** - must haves: deze eisen (requirements) moeten in het eindresultaat terugkomen, zonder deze eisen is het product niet bruikbaar;
- **S** - should haves: deze eisen zijn zeer gewenst, maar zonder is het product wel bruikbaar;
- **C** - could haves: deze eisen zullen alleen aan bod komen als er tijd genoeg is;
- **W** - won't haves (ook wel would haves genoemd): deze eisen zullen in dit project niet aan bod komen maar kunnen in de toekomst, bij een vervolgproject, interessant zijn.

Huidige situatie

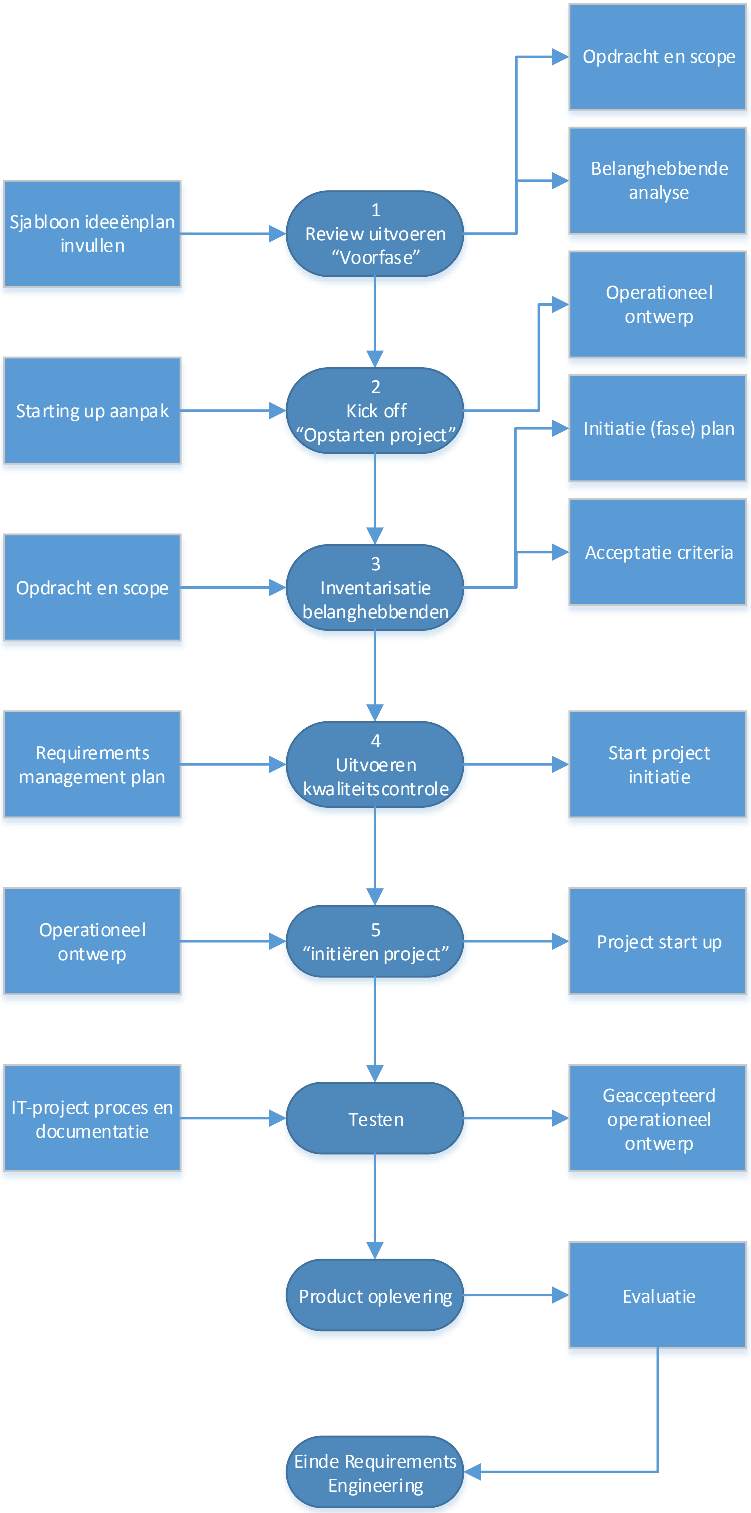
Uit de case study enquête is gebleken dat 78.26% (18 van de 23 personen) van de NS IT-afdeling requirements zeer belangrijk in een project. Momenteel is er een maandelijkse checklist voor het meten van de requirements. Deze lijst wordt bijgehouden om te zien welke status een bepaald project heeft.

Tooling
Bij de NS wordt gebruik gemaakt de tool Enterprise Architecture. Via TOPAAS zijn de templates te vinden. Documenten die in TOPAAS worden opgeslagen zijn onder andere:
- *Ideeën plan*
Het Ideeën plan is de eerste concrete stap om een project te starten. In de template van "het ideeën plan" staan redenen voor een doorvoering van een idee.
- *Operationeel ontwerp*
In het operationeel ontwerp wordt het gewenste systeem, omgeving, toekomstige gebruikers en de eisen voor het systeem beschreven.
- *Systeem eisen*
In het systeem eisen document is het overzicht van de requirements beschreven die aan het gewenste systeem zijn gesteld.
- *Systeem ontwerp*
De belangrijkste ontwerp beslissingen voor het systeem worden in dit document in kaart gebracht.
- *Project Start Architectuur*
In het Project Start Architectuur (PSA) wordt de architectuur beschreven van een specifieke situatie van het project.
- *Eindrapport*
Het eindrapport bevat de management samenvatting, evaluatie van het Testobject, productrisico's en strategiebijstelling, vrijgaveadvies, evaluatie van het testproces, overdracht, aanbeveling voor toekomstige test, ervaringscijfers en de kosten en baten.

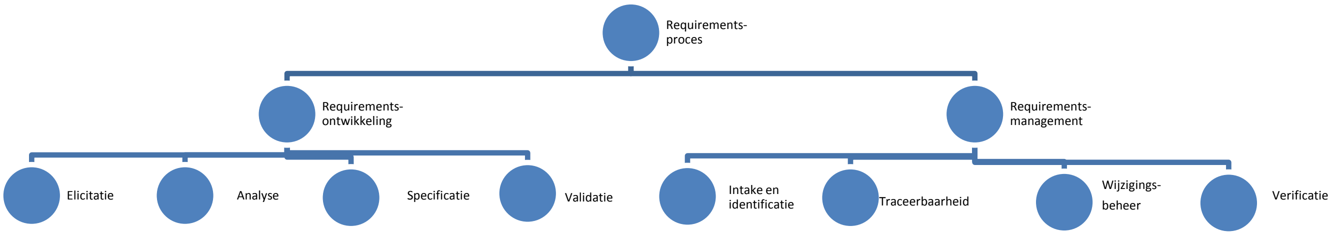
Use case
Use cases zijn een veelgebruikte techniek voor het vastleggen van gebruikersrequirements. Bij de NS worden er ook use cases gemaakt tijdens een project. Dit wordt gedaan met de methodiek van "*use-case 2.0*".
Requirements proces

[17] Hans van Vliet, Software Engineering: Principles and Practice, third edition, Wiley, Chichester (UK), 2008, p. 63

Met de volgende afbeelding wordt het requirements proces weergegeven. Dit is de versimpelde versie van de huidige atletiekbaan.


Opdracht en scope
Belanghebbende analyse
Sjabloon ideeënplan invullen
1 Review uitvoeren "Voorfase"
Operationeel ontwerp
Starting up aanpak
2 Kick off "Opstarten project"
Initiatie (fase) plan
Opdracht en scope
3 Inventarisatie belanghebbenden
Acceptatie criteria
Requirements management plan
4 Uitvoeren kwaliteitscontrole
Start project initiatie
Operationeel ontwerp
5 "initiëren project"
Project start up
IT-project proces en documentatie
Testen
Geaccepteerd operationeel ontwerp
Product oplevering
Evaluatie
Einde Requirements Engineering

Figuur 1 Requirements proces



Figuur 2 Onderverdeling van het requirements proces

Het model

Voor het model is er rekening gehouden met de drukte van informatie analisten en de informatie die de manager nodig heeft om een overzicht te krijgen over het requirements engineering proces. Zo is er de mogelijkheid om voor de Quick Scan of de Maturity Scan te kiezen. Beide zullen een overzicht geven over de "volwassenheid" van de Requirements Engineering.

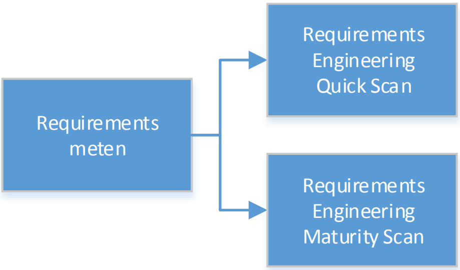

Requirements meten
Requirements Engineering Quick Scan
Requirements Engineering Maturity Scan

Figure 3 Het model in grote lijnen

In figuur 3 is weergegeven hoe men kan kiezen tussen de Requirements Engineering Quick Scan en de Requirements Engineering Maturity Scan. Bij beide scans, worden de stappen van Elicitatie, Analyse, Specificatie en Validatie doorlopen (zie figuur 4). Het verschil van de beide scans zitten in de uitgebreidheid van de vragen. De Quick Scan heeft algemene vragen over de Requirements Engineering, en de Maturity Scan heeft diepgaande vragen.

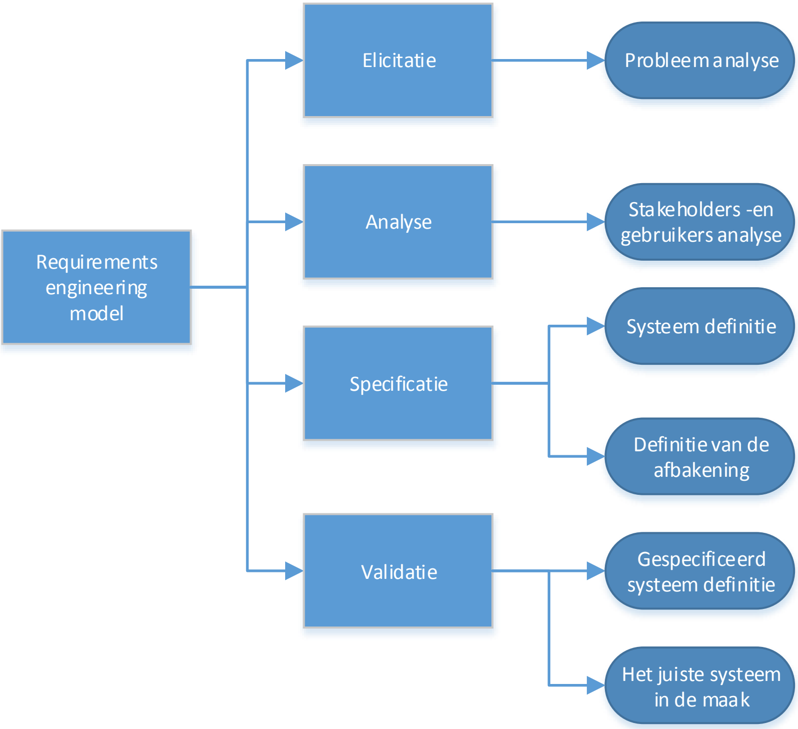

Requirements engineering model
Elicitatie
Probleemanalyse
Analyse
Stakeholders -en gebruikers analyse
Specificatie
Systeem definitie
Definitie van de afbakening
Validatie
Gespecificeerd systeem definitie
Het juiste systeem in de maak

Figure 4 - Requirements Engineering

Het doel van elk requirements proces

Voor elk requirements proces is er een doel. Deze wordt in de volgende lijst weergegeven:

Proces	Doel	Benodigdheden
Elicitatie	Probleem analyse	Probleem formulering Doelstelling Grondoorzaak analyse Systeem model Lijst van stakeholders en gebruikers Lijst van design en development beperkingen Lijst van actoren Business use case model
Analyse	Stakeholders-en gebruikers analyse	Interviews Gebruikers analyseren en begrijpen Workshops Lijst van geprioriteerde features Storyboards
Specificatie	Systeem definitie	Requirements organiseren Vision document Identificatie van initiële use cases Machtigen product manager Definitie van commerciële factoren
	Definitie van de afbakening	Prioriteren van verwachte features Basislijn requirements Herkennen en communiceren van haalbare afbakening Besproken verwachtingen
Validatie	Gespecificeerd systeem definitie	Use case modellen Use case specificaties Aanvullende specificaties Dubbelzinnigheid en specificatie overwegingen Technische methodes
	Het juiste systeem in de maak	Transitie methode (van design naar code) Test case (terugkoppelend naar de use case) Requirements traceability Requirements change management proces Requirements methode

Tabel 5 – doelen van requirements engineering

Requirements Engineering Quick Scan

Voor een snelle "scan" van de requirements, zijn de vragen opgesteld om de requirements engineering in een snel overzicht te kunnen meten. In de onderstaande lijst zijn er vragen ontwikkeld op basis van de volgende stappen: elicitatie – analyse- specificatie – validatie.

Elicitatie

- Zijn er verschillende groepen?
- Zijn de stakeholders bekend?
- Wie is de vertegenwoordiger van de groep?
- Is het domein duidelijk voor de requirements?
- Ondersteunt het systeem het bedrijfsproces?
- Is de noodzakelijke voorwaarde van de requirements juist?
- Is de noodzakelijke voorwaarde van de requirements volledig?
- Is er genoeg capaciteit uit de operationele processen?
- Biedt het systeem ondersteuning aan het bedrijfsproces?
- Zijn de requirements geaccepteerd door de gebruikers?
- Zijn de gebruikers betrokken geweest bij het project?
- Zijn de wensen geïnventariseerd?
- Zijn de eisen geïnventariseerd?
- Zijn de requirements expliciet gemaakt?
- Is de informatie analist actief betrokken geweest tijdens het proces?
- Zijn de requirements vastgelegd in een brondocument?
- Zijn de gedetailleerde requirements herleid van de oorspronkelijke requirements?

Techniek	Domein	Stakeholders	Heden	Toekomst
Documentatie studie	x		x	x
Analyse bestaand systeem	x		x	
Interviews afnemen		x	x	
Workshop		x		x
Brainstorm sessies houden		x		
Observatie van lopende projecten	x		x	
Taakanalyse	x		x	
Prototypen		x		x
Scenario's		x	x	x

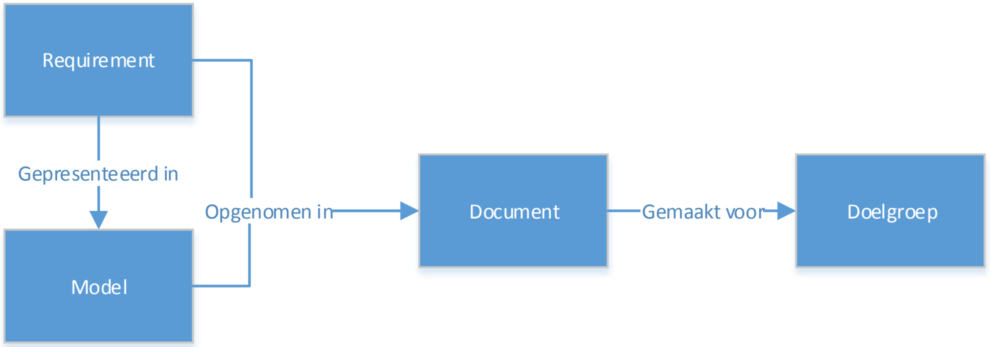
Tabel 2 Lijst van elicitatie technieken

Analyse

- Is de essentie van een specifiek requirement vastgesteld?
- Is de formulering van de requirements geanalyseerd en geconfirmeerd met de stakeholder?
- Heeft de informatie analist overlegd met de stakeholder, wat de essentie is van elk requirement?
- Is het hogere doel van een requirement vastgesteld? (wat is de reden dat dit requirement relevant is?) Is het hoger doel niet vastgesteld, verwijder het requirement.
- Zijn de requirements relevant? (als het hoger doel is niet vastgesteld)
- Zijn de business requirements vastgelegd? (om het hoger doel te benaderen en voor de traceerbaarheid van de requirements)
- Is het vervullen van een requirement de enige zinvolle manier om het hogere requirement in te vullen?

Specificatie

- Zijn de resultaten van de analyse stap vastgelegd?
- Is er een eenduidige formulering gebruikt?
- Zijn de use cases gemaakt?
- Is een datamodel gemaakt?
- Zijn de modellen gevalideerd?
- Zijn er wijzigingen?
- Is het document goedgekeurd door de stakeholder?

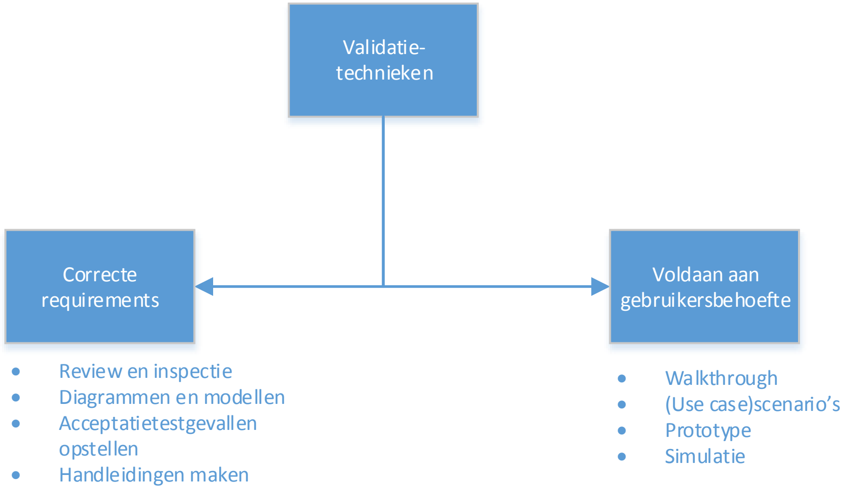


Figuur 5 Relatie tussen requirements, modellen en documenten

Validatie

Het doel van de validatie van requirements:
- zijn de requirements op een correcte manier opgeschreven?
- Verwoorden de requirements alle behoeften van de gebruiker?

- Zijn de resultaten van de specificatie gecontroleerd?
- Zijn de desbetreffende requirements door de stakeholders beoordeeld?
- Zijn de juiste requirements beschreven?
- Zijn de requirements volledig?
- Zijn de requirements consistent?



Figuur 6 Overzicht van validatietechnieken

Requirements Engineering Maturity Scan (REMS)

In REMS worden de vragen voor Requirements Engineering uitgebreid behandeld. De applicatie REMS heeft een inlogscherm, waar men kan inloggen met zijn/haar naam. Er wordt gevraagd naar het project waar men mee bezig is. Vervolgens worden de vragen gesteld van REMS.

In de "voorfase" en "opstarten project" van de NS atletiekbaan is het nog niet nodig om de requirements te meten. Vandaar dat de Requirements Engingeering Maturity Scan (REMS) pas wordt gebruikt bij "initiëren project" en "project oplevering". REMS kan op elk moment worden gebruikt om te volwassenheid te bepalen van de requirements.

REMS vragenlijst

Kwaliteit checklist voor probleem analyse	
Probleem formulering	Is de probleem formulering beschreven?
	Is het probleem begrijpelijk?
	Begrijpt het team het probleem?
	Zijn de stakeholders (incl. management) het eens met de probleem formulering?
	Is het duidelijk voor het team welk probleem ze gaan oplossen?
Doelstelling	Is de doelstelling gedocumenteerd?
	Is de doelstelling gedocumenteerd in de NS business case?
	Zijn de werkzaamheden beschreven?
	Is het bedrijfsbelang bekend?
	Zijn de doelstellingen beschreven in meetbare eenheden?
Grondoorzaak analyse	Is er een analyse gedaan naar de grondoorzaak?
	Zijn de teamleden ervan bewust dat er een "echt probleem" wordt opgelost, en niet een symptoom of algemeen basisprobleem?
Systeem model	Is de grens van de oplossing geïdentificeerd?
	Zijn de overige interacties met het systeem geïdentificeerd?
	Is het systeem onderverdeeld in subsystemen?
	Zo ja, zijn alle subsystemen gedefinieerd?
	Zijn de grenzen van elk subsysteem begrepen?
	Is er een plan voor het identificeren en tot het komen van requirements?
Lijst van stakeholders en gebruikers	Zijn alle gebruikers van het systeem geïdentificeerd?
	Zijn alle stakeholders geïdentificeerd?
	Is er een gedegen stakeholderanalyse geweest?
	Heb je buiten de set van de huidige gebruikers en stakeholders gekeken, bijvoorbeeld de personen die te maken hebben met de administratie, installatie, support of training?
	Weten de teamleden dat alle stakeholders zijn geïdentificeerd?
Lijst van design en development beperkingen	Heeft het team alle beperkingen van het systeem beschreven?
	Heeft het team de ontwikkel beperkingen geïdentificeerd?

	Zijn de beperkte bronnen (zoals budget, productie kosten, politiek of contractuele requirements, systeem requirements, omgeving factoren, regulaties, personeel, software proces en tooling) in aanmerking gebracht?
Lijst van actoren	Heb je alle actoren gevonden die interactie hebben met het systeem?
	Is elke actor betrokken bij tenminste één use case?
	Zijn er actoren die een vergelijkbare rol hebben in relatie met het systeem? Zo ja, probeer het onder één actor te brengen.
	Is er een actor die het systeem op meerdere manieren gebruikt? Zo ja, dan moet je verschillende actoren beschrijven.
	Hebben de actoren intuïtieve en beschrijfbare namen? Kunnen gebruikers en klanten de namen begrijpen?
Business use case model	Zijn de functionaliteiten van de business use case begrijpelijk van het voorgestelde systeem?
	Is het business object model begrijpelijk voor de entiteiten in het betrokken business proces?
	Heeft het team begrepen wat de specifieke functionaliteiten zijn van het voorgestelde systeem?

Kwaliteit checklist voor gebruikers begrijpen en stakeholder behoeften	
Gestructureerde interviews	Is er een gestructureerd interview afgenomen?
	Heeft het interview alle belangrijke aspecten van het systeem behandeld? (Denk aan: product requirements, het doel, het gebruik, betrouwbaarheid, in gebruik neming, beheer)
	Was het aantal geïnterviewde gebruikers en stakeholders voldoende?
	Zijn de interviews vrij van tegenstrijdigheid om de kwaliteit te kunnen waarborgen?
	Zijn de andere hoofd-invloeden begrepen door het team?
Gebruikers analyseren en begrijpen	Begrijp je wie de gebruikers zijn en wat de geschiktheid moet zijn om de applicatie te gebruiken?
	Heb jij de hoofdgebruiker ontdekt die nodig is om gericht het product te ontwikkelen?
	Zijn de hoogste prioriteiten geconvergeerd na de redelijke interviews?
	Zijn de gegevens over gebruikers, behoeften, en andere gesuggereerde features samengevat voor toekomstige referentie?
Workshops	Is er een workshop georganiseerd met de vereiste stakeholders?
	Was de workshop zo georganiseerd dat de stakeholders werden aangemoedigd om input te geven?
	Resulteerde de workshop tot een eenduidig begrip van het toekomstige systeem?
	Was het development team betrokken om redelijke technische-en projectmatige haalbaarheid te verzekeren?

	Is er een brainstormsessie gehouden voor het verzamelen van requirements?
	Worden de ideeën en resultaten van de brainstormsessie gedocumenteerd?
Lijst van geprioriteerde features	Bestaat er een prioriteiten lijst van features?
	Is er een ruwe schatting gemaakt van de inspanning van het development team?
	Is de opgenomen informatie gedocumenteerd voor verdere referentie?
Storyboards	Als de applicatie innovatief is, heb jij middelen ontwikkeld om de applicatie te demonstreren aan de gebruiker?
	Was de reactie van de gebruikers in acht genomen, en gereflecteerd op het huidige begrip van het systeem?
	Kun je een paar voorbeeld use cases beschrijven over “hoe het systeem gebruikt zal worden”?

Kwaliteit checklist voor systeem definiëren	
Requirements organiseren	Heb je het plan opgericht voor de requirements van de organisatie?
	Heb je begrepen welke tool je gebruikt voor het beheren van het proces?
	Heeft je organisatie systeem alle typen requirements opgenomen?
	Ben je nog op zoek naar design beperkingen?
Vision document	Heb je een vision document voor het project?
	Heeft het vision document relevante bronnen (auteur, stakeholders, experts, development team) over de aspecten van het project (systeem requirements, beperkingen, andere systemen en applicaties, concurrerende producten)?
	Is het vision document beschreven in een gevestigd template voor bepaalde doeleinden?
Identificatie van initiële use cases	Heb je de basis use cases geïdentificeerd (benoemd en beschreven)?
Machtigen product manager	Is er een product manager of project verdediger die het team ondersteunt?
	Is hij/zij de officiële persoon van feature-level veranderingen?
	Weet je hoe je het product beschrijft aan de buitenwereld?
Definitie van commerciële factoren	Heb je de requirements/het beleid gedefinieerd en beschreven voor documentatie? (denk aan: installatie, prijs, configuratie, beheer, licentie, training voor eindgebruiker, product benaming, merk, label)

Kwaliteit checklist voor afbakening	
Prioriteren van verwachte features	Heb je de risico’s voor de features ingeschat en geprioriteerd?

Basislijn requirements	Heb je de baseline requirements vastgesteld voor de release waar je aan werkt?
	Begrijp je welke features kritisch zijn voor de release?
Herkennen en communiceren van haalbare afbakening	Heeft het project een haalbare afbakening?
	Heb jij de mogelijke -en niet mogelijke beslissingen genomen voor het project voor de afbakening van het project?
	Zijn de hoofdmanagers en stakeholders het eens met de afbakening?
Besproken verwachtingen	Zijn de verwachtingen voor de huidige release door het team begrepen?
	Zijn de verwachtingen besproken en is er een akkoord gekomen met de stakeholders buiten het team (inclusief de eindgebruiker/klant)?

Kwaliteit checklist voor systeem verfijning	
Use case modellen	Zijn de use cases gemaakt volgens de theorie van use case 2.0?
	Als het systeem subsystemen heeft, reflecteert het use case model juist op de subsystemen?
	Heb je voor alle use case modellen een reflectie gedaan?
	Zijn de use case modellen uniek en intuïtief?
	Heeft de use case modellen begrijpbare namen, zodat er in een later stadium geen verwarring komt?
	Zijn alle nodige systeem features geïdentificeerd in één of meer use cases?
	Begrijpen klanten en gebruikers de namen en beschrijvingen van de use cases?
	Als je kijkt naar het use case model, kun je een goed beeld vormen van de systeem functionaliteiten en hoe ze zijn gerelateerd?
	Komen de uitgewerkte use cases overeen met alle functionele requirements?
	Heeft het use case model overbodigheden?
	Heeft het use case model meer functionaliteiten dan er staat beschreven in de requirements?
	Heeft het model geïdentificeerde externe relaties?
	Kan het model worden versimpeld met alternatieve relaties?
Use case specificaties	Is elke use case betrokken bij tenminste één actor?
	Geeft de beschrijving een goed beeld over de use case?
	Is het duidelijk wie de use case zal uitvoeren? Is het doel van de use case vastgesteld?
	Hebben de uitgebreide use cases de nodige secties en geschikte inhoud voor namen, actoren, korte beschrijving, primaire en alternatieve event flows, pre- en post condities, en speciale requirements?
	Is het duidelijk hoe en wanneer de use case event flows starten en eindigen?

	Is elke use case onafhankelijk van elkaar?
	Zijn er use cases die vergelijkbare gedragingen/flows hebben?
	Is een deel van de use case event flows al gemodelleerd in een andere use case?
	Moeten de event flows van één use case in een andere event flow worden geplaatst?
	Voldoen de use cases aan de requirements voor de besturing? Zijn de use case specificaties gerefereerd aan de non-functionele requirements waar het nodig is?
	Is de frequentie van communiceren tussen de actor en de use case in overeenstemming met de gebruikersverwachting?
	Is er een beschrijving van wat er gebeurt als een conditie niet is voldaan?
	Zijn er use cases die te complex zijn?
	Is de interactie tussen de actor en informatie uitwisseling duidelijk?
Aanvullende specificaties	Heb je een geschikt template gebruikt voor de specifieke doelen?
	Zijn de functionele requirements, inclusief het use case model, gereflecteerd in de aanvullende specificaties?
	Zijn de non-functionele requirements (zoals bruikbaarheid, betrouwbaarheid, performance, en beheerbaarheid) geïdentificeerd en beschreven?
	Zijn de passende ontwerp beperkingen geïdentificeerd en beschreven?
	Zijn aanvullende requirements gelinkt aan de use cases (waar nodig is)?
Dubbelzinnigheid en specificatie overwegingen	Over het algemeen, heeft het team het juiste niveau van de specificaties voor het project bereikt?
	Hoe weet je dat het juiste niveau is bereikt?
Technische methodes	Zijn er voldoende technische methodes gebruikt om de dubbelzinnigheid te verwijderen? (Vooral in gevallen waar er geen miscommunicatie kan plaatsvinden)
	Als er voldoende technische methodes zijn, zijn deze methodes begrepen door de stakeholders?

Kwaliteit checklist voor het juiste systeem bouwen	
Transitie methode (van design naar code)	Is er een use case realisatie (samenwerking) voor alle use cases in het use case model?
	Zijn er andere realisaties voor andere functionele requirements?
Test case (terugkoppelend naar de use case)	Heb je de use cases gebruikt voor de test case ontwikkeling?
	Heb je de NS processen gebruikt om te testen?
	Zijn er één of meerdere test gevallen voor elke use case?
Requirements traceerbaarheid	Heb jij een plan gemaakt voor de requirements traceerbaarheid?
	Heb je voldoende tooling geïdentificeerd en geïmplementeerd?

	Heb jij een specifiek traceerbaar model geïdentificeerd en gevolgd voor dit project?
	Heb je de traceerbaarheid zoveel mogelijk benut?
Requirements change management proces	Begrijp je de bron veranderingen en dynamische veranderingen van het project?
	Heeft de project/product manager de controle over dit project?
	Is er een passende change control board opgericht en is dat functioneel voor het project?
	Kun je veranderingen vastleggen en effectief beheren met de geïmplementeerde tooling?
	Heb je een manier om defecten van het project vast te leggen en traceren?
Requirements methode	Heb je de juiste requirements methode gebruikt?
	Reflecteert het de hoofd prioriteiten van kritische aspecten en veiligheid van het project?
	Verwijdert de methode de onnodige documentatie?
	Ondersteunt de tooling de gekozen methode voldoende?

Aanbeveling

Op basis van de interviews, enquête en literatuur onderzoek is dit model ontworpen. Het REMS model is een conceptueel model wat de requirements metrieken vastlegt.

Er is een uitgebreide versie ontworpen en een kortere versie voor eventuele tijdnood tijdens het project. Dit is natuurlijk afhankelijk van de wensen en eisen van de informatie die nodig is.

Veel vragen kunnen van toepassing zijn voor de scans. In de Maturity Scan is er gekozen voor diepgaande vragen over Requirements Engineering. Hierdoor is de mogelijkheid om de requirements breed te onderzoeken. Het nadeel kan zijn dat het lang kan duren om REMS in te vullen. Daarom is er een optie om de checklist af te gaan als “quick scan”. In de quick scan is er gekozen voor vragen die belangrijk leken voor NS.

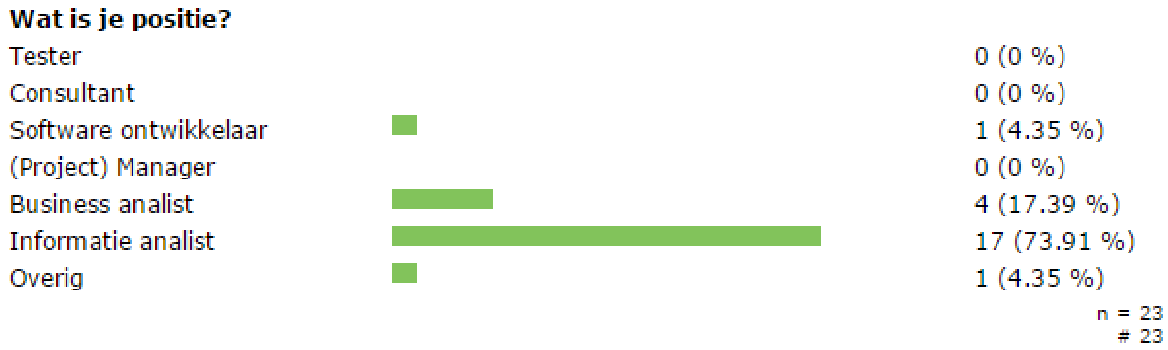
In verband met tijdnood en ontbrekende kennis, is het niet meer mogelijk geweest om de functionaliteit toe te voegen om de applicatie dynamisch te maken. Het idee is om antwoorden waar men “ja” op heeft geantwoord, te laten verdwijnen in de volgende keren wanneer men inlogt op eigen gebruikersnaam. Dus men zal alleen de vragen zien waar ze “nee” op hebben geantwoord, en daarop ingaan.

Literatuur

- Arendsen, M., H.J.J. Cannegieter, A. Grund, P. Heck, S. de Klerk & J. Zandhuis (2010), *Succes met de requirements!*, 2^{e} herziene druk, Sdu, Den Haag.
- Aydinli, O., E. Hendriks, J. Zandvliet (2013), *Smart requirements 2.0*, Sdu, Amsterdam.
- IREB (2015), IREB Foundation syllabus, Certified professional for requirements engineering – Foundation level – Version 2.0, International Requirements Engineering Board.
- Leffingwell, D., Widrig, D. (2007), *Manageing software requirements*, Pearson Education

Bijlagen

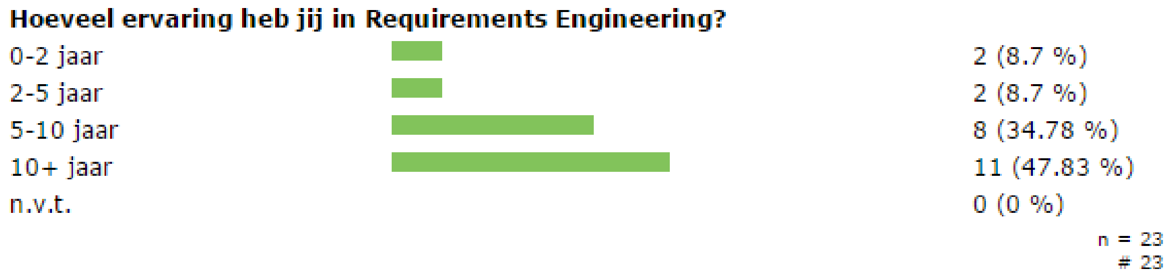
Bijlage - Enquête resultaten



Wat is je positie?

Tester	0 (0 %)
Consultant	0 (0 %)
Software ontwikkelaar	1 (4.35 %)
(Project) Manager	0 (0 %)
Business analist	4 (17.39 %)
Informatie analist	17 (73.91 %)
Overig	1 (4.35 %)

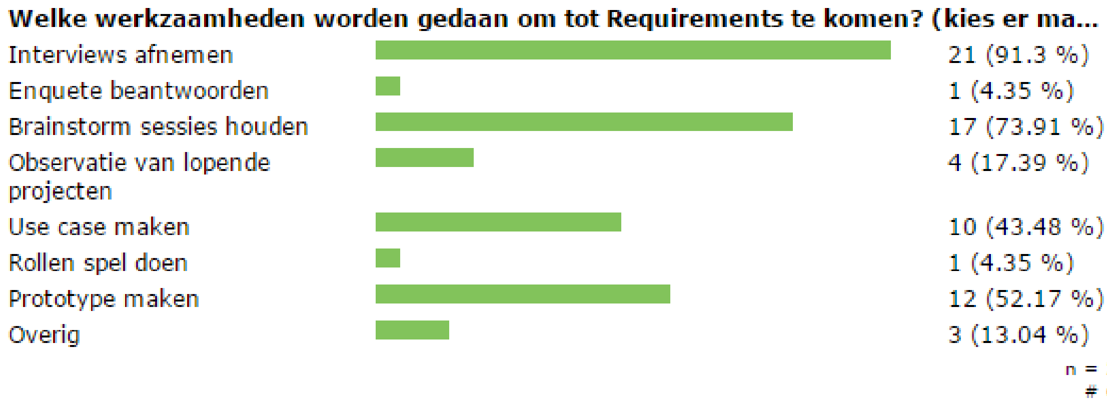
n = 23
# 23



Hoeveel ervaring heb jij in Requirements Engineering?

0-2 jaar	2 (8.7 %)
2-5 jaar	2 (8.7 %)
5-10 jaar	8 (34.78 %)
10+ jaar	11 (47.83 %)
n.v.t.	0 (0 %)

n = 23
# 23

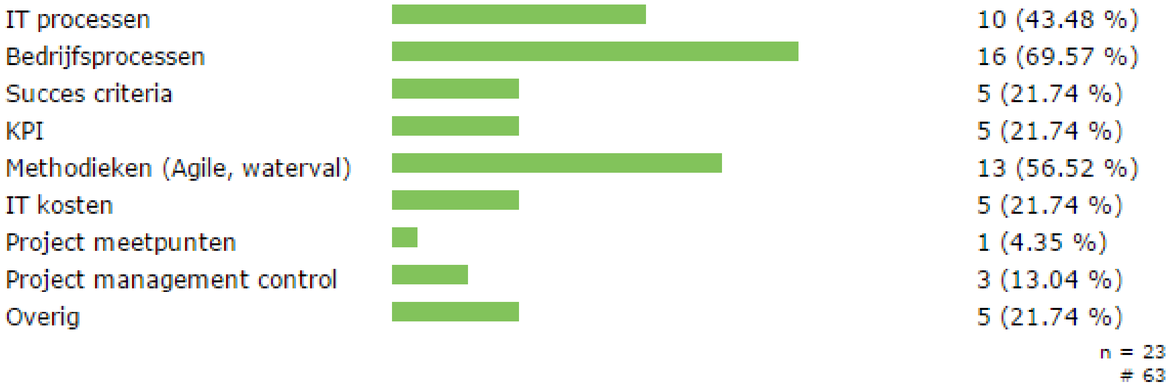


Welke werkzaamheden worden gedaan om tot Requirements te komen? (kies er ma...

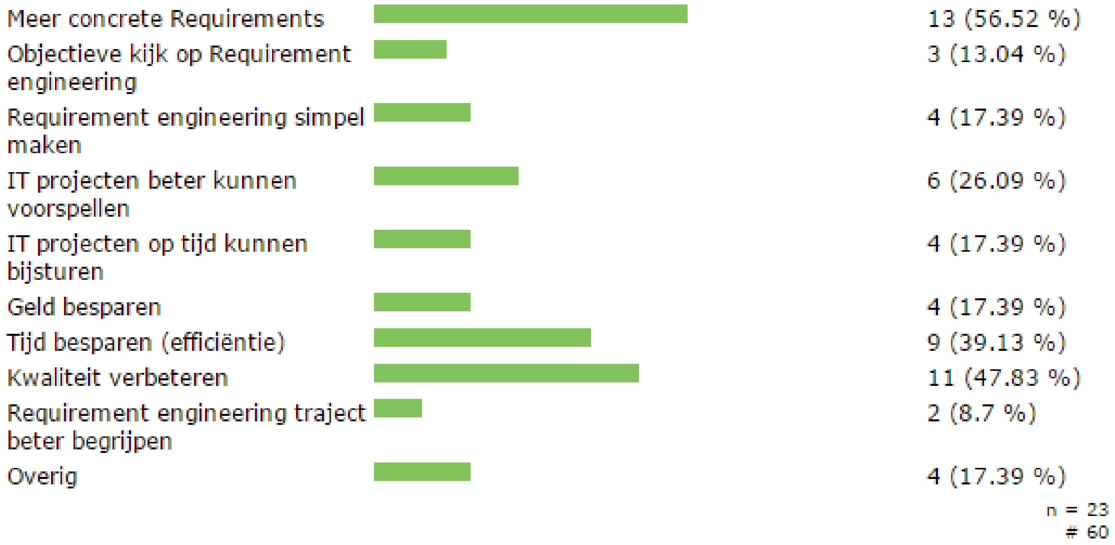
Interviews afnemen	21 (91.3 %)
Enquete beantwoorden	1 (4.35 %)
Brainstorm sessies houden	17 (73.91 %)
Observatie van lopende projecten	4 (17.39 %)
Use case maken	10 (43.48 %)
Rollen spel doen	1 (4.35 %)
Prototype maken	12 (52.17 %)
Overig	3 (13.04 %)

n = 23
# 69

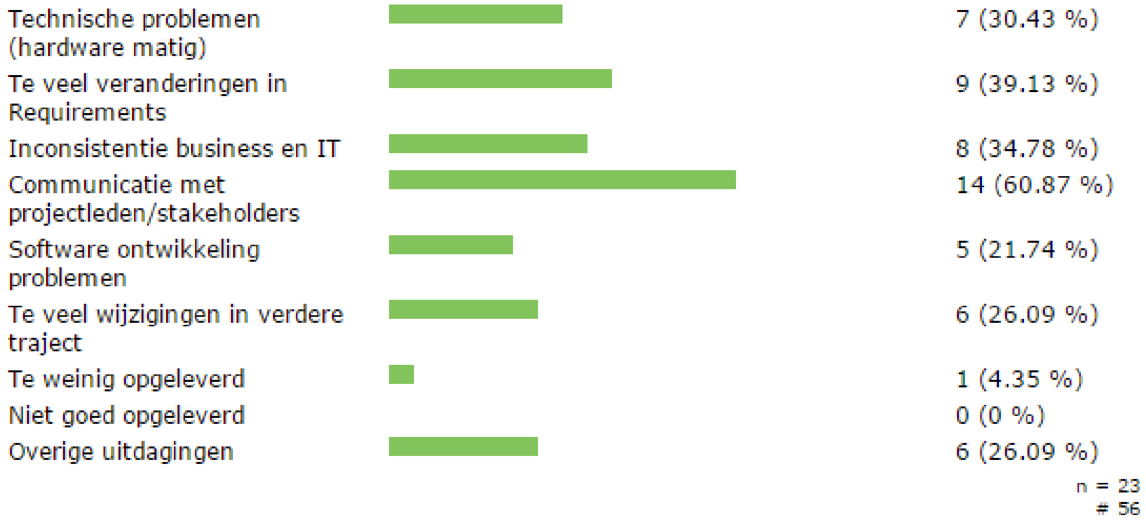
Welke onderdelen zijn belangrijk voor de uitkomst van een project?



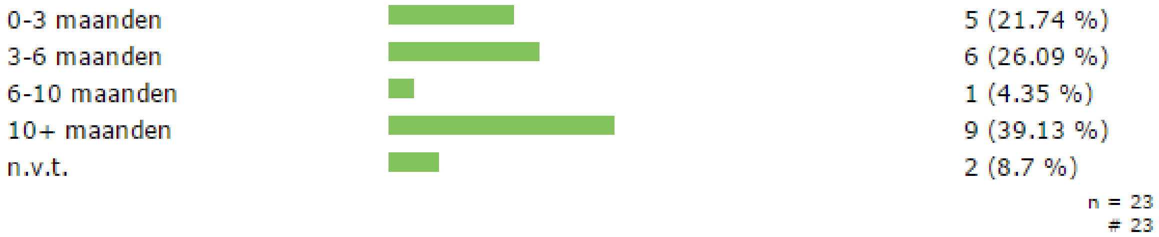
Welk resultaat zou een verbetering van de Requirements engineering activite...



Welke uitdagingen zitten er in jouw IT projecten? (kies er max. 3)



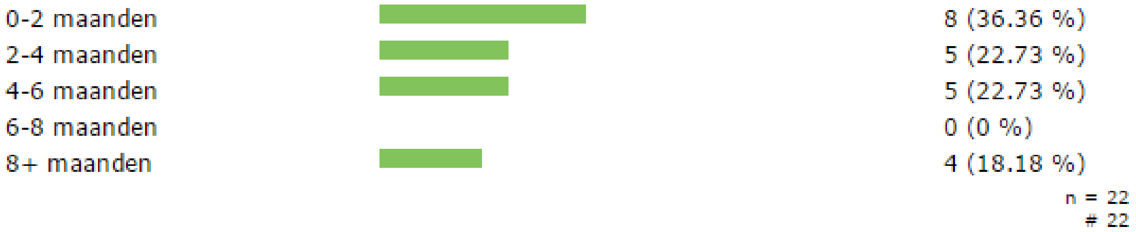
Hoe lang hebben jouw projecten (gemiddeld) geduurd, die te maken hadden met...



0-3 maanden	5 (21.74 %)
3-6 maanden	6 (26.09 %)
6-10 maanden	1 (4.35 %)
10+ maanden	9 (39.13 %)
n.v.t.	2 (8.7 %)

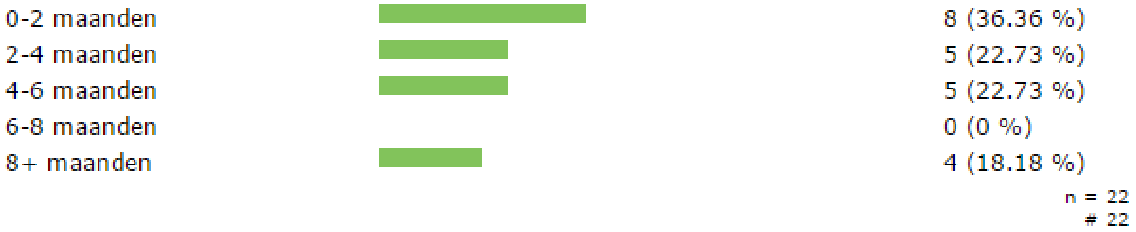
n = 23
23

Hoe lang duurde het Requirements engineering traject van die projecten?



0-2 maanden	8 (36.36 %)
2-4 maanden	5 (22.73 %)
4-6 maanden	5 (22.73 %)
6-8 maanden	0 (0 %)
8+ maanden	4 (18.18 %)

n = 22
22

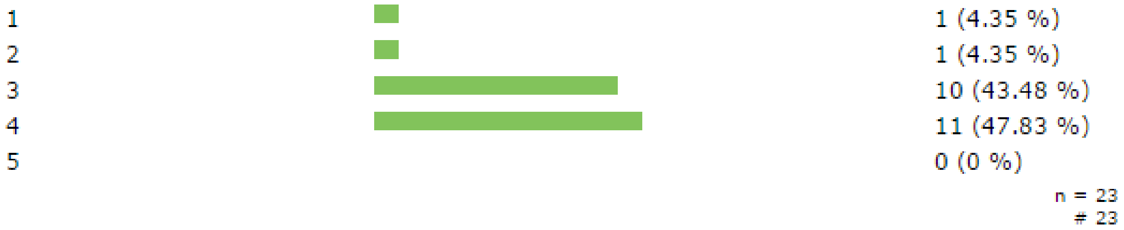


Hoe lang duurde het Requirements engineering traject van die projecten?

0-2 maanden	8 (36.36 %)
2-4 maanden	5 (22.73 %)
4-6 maanden	5 (22.73 %)
6-8 maanden	0 (0 %)
8+ maanden	4 (18.18 %)

n = 22
22

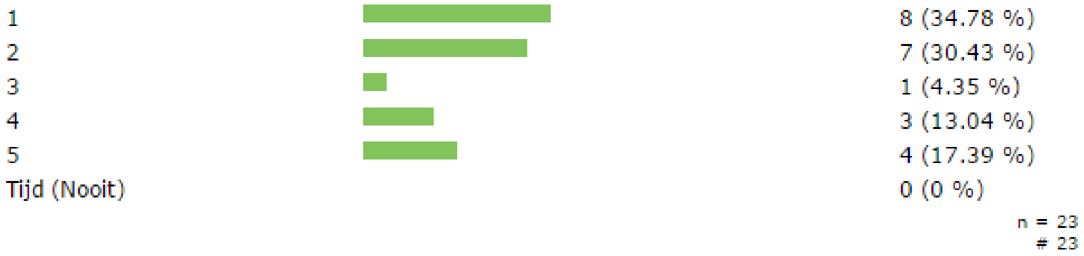
Hoe verlopen Requirements Engineering activiteiten in de huidige projecten?
Beoordeling activiteiten (Onvoldoende - Goed)



1	1 (4.35 %)
2	1 (4.35 %)
3	10 (43.48 %)
4	11 (47.83 %)
5	0 (0 %)

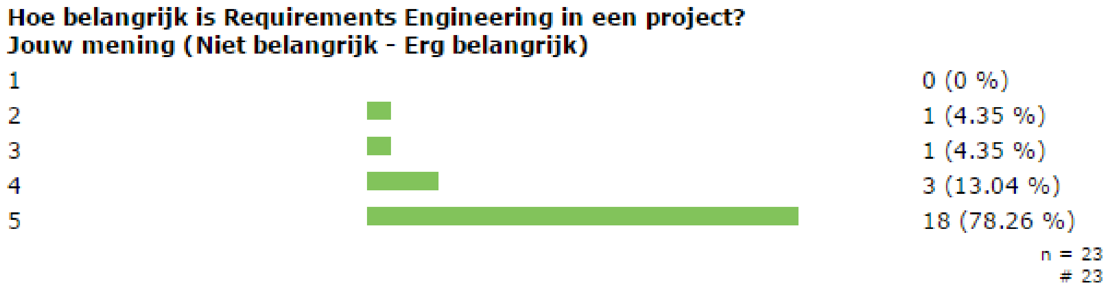
n = 23
23

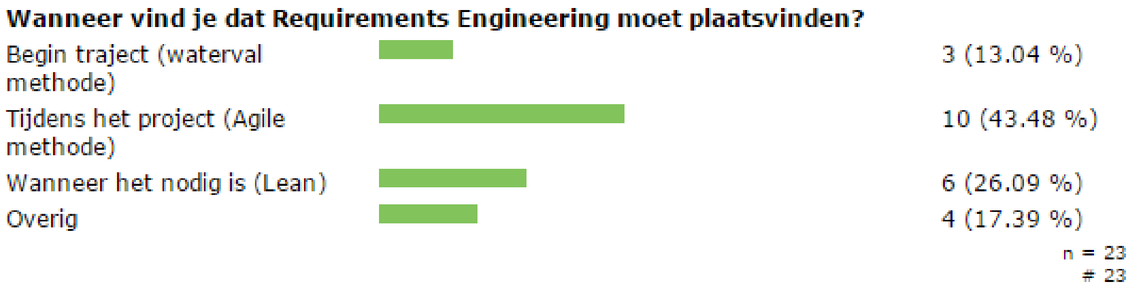
Hoe vaak heb jij te maken met Requirements Engineering?
Tijd (Dagelijks - Maandelijks)

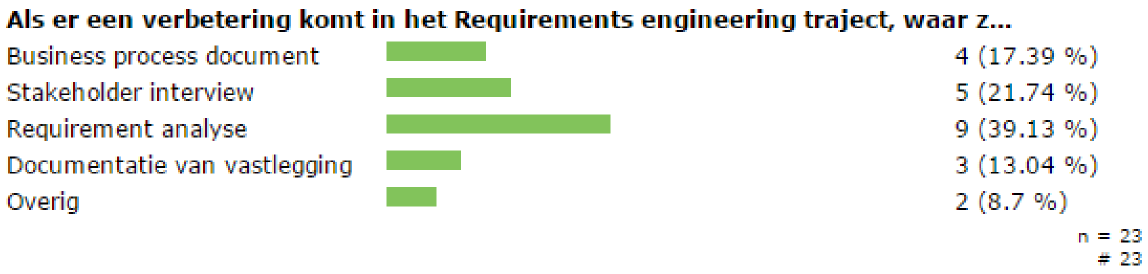


1	8 (34.78 %)
2	7 (30.43 %)
3	1 (4.35 %)
4	3 (13.04 %)
5	4 (17.39 %)
Tijd (Nooit)	0 (0 %)

n = 23
23







Bijlage 2 – Lijst met "verboden woorden" in de kwaliteitscontrole

Tijdens documenteren worden er bepaalde woorden gebruikt. Dit zijn woorden die niet te vinden moeten zijn in een document. Dit zijn de zogenaamde "verboden woorden" in de kwaliteitscontrole.

… op zich …	… zo … mogelijk…	… sommige …
… optimaal …	… bijna altijd …	… deels …
… meestal …	… bijna allemaal …	… weinig …
… in het algemeen …	… normaliter …	… beperkt …
… in principe …	… zelden …	… voldoende …
… etc., etc. …	… vrijwel net als …	… eigenlijk …

Appendix 4: REMS 2.0 interview

Main points of the interview:

- Who is the owner of the RE measurements?
- What are the decisions after measuring?
- What is the value of measuring?
- What is the goal of measuring?
- What is the impact?

Minutes of the interview:
It's good to know who is asking for the measurement of RE, and what type of decisions you will do after the RE is measured. In the current situation, it is a proactive initiative.

Keep in mind what type of decisions you will take after the RE is measured. What are the goals? And who is making the decisions.
In the end you want to know if the RE is relevant. And how can you effectively measuring up "time"?

In the meeting with the expert, we discussed what practice knowledge or parameters will fit in the context. The expert has experiences in metrics and knows what adds value.

The same question returned: what do you mean with "measure RE"? That is the hardest part, if you can measure this, then you can measure every KPI. The optimal number is a factor that influences the RE. But how do you know the optimal number. What we need to know is: what would you like to achieve?

Teams often don't recognize what the management will do with the results of the measurements. We should have measurements created that helps teams to grow in maturity.

Heart of the matter: who will be taking decisions after the measurements.
If it doesn't matter to the team, you're not going to achieve anything.
Visualize how effective a team is. Is it going into your direction? And be clear about the usefulness of dashboards.

The bank has a centralized management. They want a dashboard they want. We want to leverage the experience. On portfolio level it means: how much are we achieving and what are the RE.

What another team is doing: they are asking the teams (business partner) to express their value of what they deliver (features and realization). Are you doing the right stuff (release). You report what value you realized for which business/ program/ capabilities. The benefit is not described by teams, but business partners around the teams.

Another expert sees the risks of KPI's and measurements. Organization around the teams are doing a lot of measurements. They report to management, but it never gets back to the team. Teams don't see management supports the team's decision. This is more "Command and control", but not lean leadership. (Management will probably say the opposite.)

What is the best way to do portfolio management? That should be the "Value driven approach". What value do we get out, with the money we put in. Quality and reliability are important.

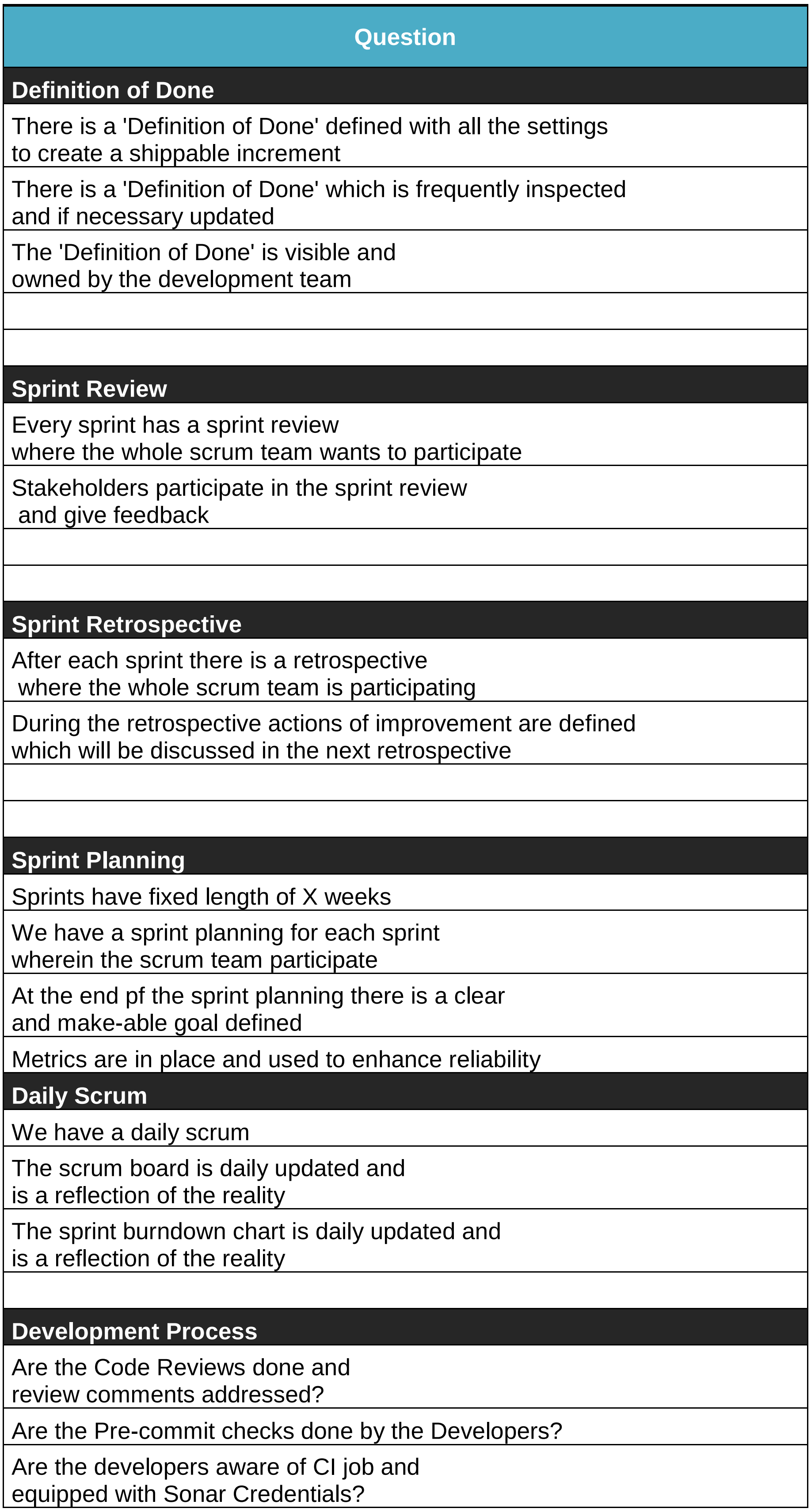
There are different kind of approaches. But we should recognize that there's not a "one size fits all" solution. It differentiates. It's not "one measurement".

Way to measure RE: contributors and weightages to each and there's an impact. Normalize the contributors, and then add up. This will be the RE metrics indication. So, what is the real value (what do you want to get out of this)?

There's a reason why teams are offshore. How do you want to deal with that. What's the target you want to achieve? You'll have an ideal team, and how do you measure the teams. What do percentages mean, and what is the 100%? Maybe more important: how do you get there?

Productivity: condensate and the team sitting together. Do they have the capability? Team composition is not the same as measuring RE. When the team is sitting together, that's a composition.

Appendix 5: Questions in REMS 2.0



Question
Definition of Done
There is a 'Definition of Done' defined with all the settings to create a shippable increment
There is a 'Definition of Done' which is frequently inspected and if necessary updated
The 'Definition of Done' is visible and owned by the development team
Sprint Review
Every sprint has a sprint review where the whole scrum team wants to participate
Stakeholders participate in the sprint review and give feedback
Sprint Retrospective
After each sprint there is a retrospective where the whole scrum team is participating
During the retrospective actions of improvement are defined which will be discussed in the next retrospective
Sprint Planning
Sprints have fixed length of X weeks
We have a sprint planning for each sprint wherein the scrum team participate
At the end pf the sprint planning there is a clear and make-able goal defined
Metrics are in place and used to enhance reliability
Daily Scrum
We have a daily scrum
The scrum board is daily updated and is a reflection of the reality
The sprint burndown chart is daily updated and is a reflection of the reality
Development Process
Are the Code Reviews done and review comments addressed?
Are the Pre-commit checks done by the Developers?
Are the developers aware of CI job and equipped with Sonar Credentials?

Is a licensed IDE used for development?
Product Backlog& Product Owner
The product backlog items with the highest priority are refined
The product backlog is prioritized
The stakeholders/product owner participate in the refinement
The refinement is used for knowing the functional and technical background of a user story
The product owner is available for the development team
The product owner has a operational available backlog for 2 sprints
The product owner leads the sprint review
All the work for a team is described in one single backlog
The most important stakeholders are involved in the product backlog refinement meeting and the administration of the product backlog
The product backlog is visible for users and customers and is expressed in a user friendly way
The stakeholders which are participating the refinement are delivering explicit input
Is there a reference for Non Functional Requirements(NFRS) and whether that has been agreed with Functional Maintenance Team?
Cross-Functional&Happy Team
The team is self-organized, there is no formal boss
We are a cross functional scrum team where people are working within their strength and share knowledge to get the team to a higher level
Management focus is on the results and not the way of working in the team
I think my manager is a good manager
Team members are actively committed
The Scrum team is active in finding solutions for existing problems and constantly searching for improvements
The Scrum team member are actively participating in sprint events
Team members take actively ownership of team tasks
Our team is customer focused
We are actively working on personal development
Managers actively participating in and are involved with the personal development of the team members
There is a team culture to improve our selves
(senior)Management is actively participating in solving impediments
User Stories
Are functional requirements expressed as User Stories?
Are the requirements readable and understandable by development, test, support and stakeholders?
Are acceptance criteria identified for the requirements?

Are for each user story / requirement / use case the attributes tracked, such as estimate of how long it will take to implement, and who will implement the requirement, and the traceability to test cases.
Scrum of Scrum
Is there a Scrum-of-Scrums?
Do the Teams have the same Sprint Length?
Do the Teams review an integrated result?
Has every Team all Scrum roles?
Are specialists shared across the Teams without impeding single Teams?
Is the Product Backlog suitable for multiple Teams (theming, grouping notions)?