

Crowd-Centric Requirements Engineering: A method based on crowdsourcing and gamification

R. Snijders

A. Özüm

S. Brinkkemper

F. Dalpiaz

Technical Report UU-CS-2015-004
March 2015

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands
www.cs.uu.nl

ISSN: 0924-3275

Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands



Crowd-Centric Requirements Engineering

A method based on crowdsourcing and gamification

Abstract

Requirements Engineering (RE) is an essential process in the continuous development of software products. User involvement has a large potential for improving the quality of RE and thereby the quality of software. Shortcomings of current RE methods and threats in the complex environment of Software Product Organizations (SPO) trigger the need for an innovative method. Crowdsourcing and gamification are two emerging trends that provide opportunities to overcome the shortcomings and threats. In this thesis, the Crowd-Centric Requirements Engineering (CCRE) method, which guides SPOs in involving all stakeholders in the RE process, is described. A prototype, Refine, is built to demonstrate the method. Although some obstacles to large-scale adoption exist, the evaluation of this demonstration shows that CCRE can provide a useful process, useful requirements, engaged stakeholders and valuable interaction among those stakeholders. Through these aspects, the method has shown potential to improve requirements engineering in software production.

Snijders, Remco

r.snijders@uu.nl

13-02-2015

Contents

Table of figures	4
Table of used tables	5
Used abbreviations	6
Acknowledgements	7
1. Introduction	8
2. Research Approach	11
2.1. Research Questions	11
2.2. Research Method.....	12
2.3. Validity & Reliability.....	16
3. Theoretical Foundation.....	17
3.1. Requirements Engineering	17
3.2. Software Product Management	19
3.3. Combining RE and SPM.....	21
3.4. The context of RE	28
3.5. Customer and user involvement in RE.....	31
3.6. Crowdsourcing	33
3.7. Gamification	37
3.8. Conceptual model.....	39
3.9. Method Engineering	40
4. Expert Interviews	42
4.1. Introduction	42
4.2. Results.....	43
4.3. Summary.....	49
5. The CCRE Method	50
5.1. Development	50
5.2. Method structure	50
5.3. Phases and activities.....	52
5.4. Concepts	66
5.5. CCRE drivers.....	67
5.6. Excluded and modified advices	68
6. The prototype: <i>Refine</i>	69

6.1. Introduction	69
6.2. Technical specifications	71
6.3. Gamification elements.....	71
6.4. Pages.....	73
7. Demonstration	76
7.1. Introduction	76
7.2. Instantiating the method.....	77
8. Evaluation	89
8.1. Observation	89
8.2. Participant questionnaire	90
8.3. Product management interview	95
8.4. Expert evaluation.....	95
8.5. Aggregated findings.....	100
9. Conclusions	103
9.1. Discussion	103
9.2. Future work	105
10. References	107
Appendix A – Expert interview questions.....	114
Appendix B – Product Management interview questions	115
Appendix C – Expert evaluation statements and questions	116
Appendix D – Interview Summaries.....	117
Appendix E – CCRE Method.....	139

Author: Remco Snijders

E-mail: r.snijders@uu.nl

Student number: 3469743

Program: Business Informatics

Internship period: September 2014 - February 2015

Research organization: KPMG Netherlands - MC ITA - Technology Advisory

Company supervision: Atilla Özüm, Senior Manager

Research supervision: dr. Fabiano Dalpiaz & prof. dr. Sjaak Brinkkemper

Table of figures

Figure 1-1: The domain positioning of the CCRE method.....	10
Figure 2-1: The DSRM process model (Peffer et al., 2007)	12
Figure 2-2: The research activities and deliverables.....	12
Figure 2-3: The relation of the four evaluation techniques to RE quality	15
Figure 3-1: The SPM competence model (Bekkers et al., 2010).....	20
Figure 3-2: Visualization of RE and SPM combined, based on current literature.....	22
Figure 3-3: An example of requirements management in Jama software	24
Figure 3-4: Offline (paper prototyping) and online (UserVoice) needs gathering.....	25
Figure 3-5: The outline of a MDRE repository (Regnell & Brinkkemper, 2005).....	26
Figure 3-6: Requirements in the V-Model (Hull et al., 2011).....	27
Figure 3-7: The refinery from a vision towards a standard backlog item (Vlaanderen et al., 2011).....	30
Figure 3-8: Alternating sprints in combining Scrum and SPM (Vlaanderen et al., 2011)	31
Figure 3-9: An example of the application of iRequire (Seyff et al., 2010).....	32
Figure 3-10: A screenshot of Get Satisfaction	33
Figure 3-11: Facebook crowdsources translation.....	34
Figure 3-12: The use case diagram of CrowdREquire	36
Figure 3-13: Gamification in Stack Overflow through reputation and badges.....	37
Figure 3-14: Gamification in the Zombies Run Game	37
Figure 3-15: The effects of social factors on the attitude towards and use of gamified services.	38
Figure 3-16: A screenshot of iThink	39
Figure 3-17: Conceptual model of CCRE-related concepts.....	40
Figure 3-18: An example of a PDD (van de Weerd & Brinkkemper, 2008).....	41
Figure 5-1: The simplified version of the CCRE method	51
Figure 5-2: The scope of the Microsoft Word example, Design & Page Layout features.....	52
Figure 6-1: The CD of <i>Refine</i>	69
Figure 6-2: The FA of the crowdsourcing phase of CCRE and the Crowd Involvement module.....	70
Figure 6-3: The needs overview of <i>Refine</i>	74
Figure 6-4: A need details page on <i>Refine</i>	75
Figure 6-5: A user profile on <i>Refine</i>	75
Figure 7-1: A screenshot of Qubus 7.....	76
Figure 7-2: A part of a weekly <i>Refine</i> update	82
Figure 7-3: The need for loading screens, as shared on <i>Refine</i>	83
Figure 7-4: The need for a lower number of clicks needed, as shared on <i>Refine</i>	83
Figure 7-5: The comments on "Minimize number mouse clicks needed" on <i>Refine</i>	84
Figure 7-6: The fifth most popular need, as shared on <i>Refine</i>	84
Figure 7-7: A less popular need, as shared on <i>Refine</i>	85
Figure 7-8: The role for every answer set, a design option for requirement #2	86
Figure 7-9: The repositioned Continue-button, a design option for requirement #3	87
Figure 8-1: The distribution of points among the participants (ordered by earned points, descending)	89
Figure 8-2: Frequencies of the comparison of feedback experiences	91
Figure 8-3: Frequencies of the responses on the CCRE process.....	92
Figure 8-4: Frequencies of responses on the <i>Refine</i> platform.....	93
Figure 8-5: Frequencies of responses on the game elements.....	94

Figure 8-6: Frequencies of the responses on the usefulness of *Refine*'s functionality.....94

Table of used tables

Table 1-1: A SWOT analysis of current RE techniques.....	9
Table 2-1: The mapping of the research activities on the sub questions	15
Table 3-1: The activities within RE	18
Table 3-2: The RE-related concepts of SPM and their definitions (Bekkers, van de Weerd et al., 2010).....	20
Table 3-3: the classification of different types of software (Xu & Brinkkemper, 2007)	21
Table 3-4: The pros and cons of RE and SPM.....	22
Table 3-5: Explanation of RE and SPM combined, based on current literature.	22
Table 3-6: The difference between needs, market requirements and product requirements	23
Table 3-7: Situational factors in SPM (Bekkers et al., 2010)	28
Table 3-8: Benefits and challenges of Agile RE best practices (Cao & Ramesh, 2008)	29
Table 3-9: The differences between existing concepts and crowdsourcing (Schenk & Guittard, 2009)	34
Table 3-10: the definitions of ME-related terms according to Brinkkemper (1996)	41
Table 4-1: Interviewed experts and their backgrounds.....	42
Table 5-1: Examples of method suggestions and results.....	50
Table 5-2: Mapping of the CCRE phases on the processes of RE/SPM	51
Table 5-3: Situational factors for the CCRE method	53
Table 5-4: Overview of existing interactive platforms.....	56
Table 5-5: The concepts and their definitions in the CCRE method	66
Table 6-1: Mapping of gamification elements on social factors that should be triggered.....	71
Table 6-2: Resources and points that are spent and earned with various actions.....	72
Table 7-1: The situational factors for the demonstration	78
Table 7-2: The effects of implementation and relative priority of the four identified requirements	85
Table 8-1: The activity of the different stakeholder groups on <i>Refine</i>	90
Table 8-2: frequency that the respondents gave feedback to Qubus or software in general.....	91
Table 8-3: The response on the statements in the first expert evaluation	96
Table 8-4: The response on the statements in the second expert evaluation	97
Table 8-5: The response on the statements in the third expert evaluation.....	99
Table 8-6: The colored and combined responses to the statements in the expert evaluation.....	100
Table 8-7: The mapping of the main findings on the evaluation	101
Table 9-1: Some requirements for the next version of <i>Refine</i>	105

Used abbreviations

AMT – Amazon Mechanical Turk
B2B – Business-To-Business
B2C – Business-To-Consumer
CCRE – Crowd-Centric Requirements Engineering
CD – Class Diagram
COTS – Commercial-off-the-shelf
CTO – Chief Technology Officer
DSRM – Design Science Research Methodology
FA – Functional Architecture
GRC – Governance Risk and Compliance
MDRE – Market-Driven Requirements Engineering
ME – Method Engineering
RE – Requirements Engineering
SEBoK – Software Engineering Body of Knowledge
SF – Situational Factor
SPM – Software Product Management
SPO – Software Producing Organization
SWOT – Strengths, weaknesses, opportunities and threats
TAM – Technology Acceptance Model
XP – Extreme Programming

Acknowledgements

This thesis could not have been the thesis it is without a number of people. First and foremost, I want to thank Fabiano for his constant enthusiasm, quick responses and innovative ideas to make the thesis something to be proud of. Thanks to my other two supervisors, Sjaak and Atilla, for their valuable insights and perspectives to further improve the quality of the work.

Some other people at KPMG have made a significant contribution to make the study possible. Peter got all required systems up and running; Sipke released the first version of Qubus and Joost put some urgency on the demonstration.

Finally, the 13 experts in the interviews and 18 participants in the demonstration allowed me to show actual results. Thanks!

Remco

1. Introduction

According to Nuseibeh and Easterbrook (2000), the degree to which a software system meets the purpose for which it was intended, is the primary measure of the system's success. This purpose can be represented by requirements, 'things' that mandate that something must be accomplished, transformed, produced or provided (Harwell, Aslaksen, & Hooks, 1993). According to Chemuturi (2013), there is more misunderstanding than right understanding about requirements; a large part of the high software project failure rate can be related to poorly defined and understood product requirements. Software requirements are established in a process called Requirements Engineering (RE). Corresponding with the previous statement on failure, Nuseibeh and Easterbrook (2000) identify ineffective RE as one of the factors for the many delivered systems that do not meet their customers' requirements.

Two strands of research that focus on software requirements and their establishment can be distinguished. Requirements Engineering is mostly concerned with the elicitation and specification of requirements and their relation to system functionality, goals and constraints. Software Product Management (SPM) includes Requirements Management but takes a broader perspective and focuses on how requirements are organized for and implemented in a release. Releases are versions of products, which are part of a SPO's product portfolio.

The benefits of involving customers and users in RE have been widely acknowledged. Already in 1975, Zand and Sorensen showed that user participation helps in overcoming resistance to change (Zand & Sorensen, 1975). Projecting this resistance on software, user involvement in RE can lead to improved acceptance of a system (Kujala, 2003). Other benefits include more accurate requirements, less unnecessary features (Kujala, 2003), higher requirement quality, a higher chance on project success (Emam, Quintin & Madhavji, 1996; Kujala, Kauppinen, Lehtola & Kojo, 2005), greater system understanding by the user (Damodaran, 1996), improved customer loyalty and a broadened market (Kabbedijk, Brinkkemper, Jansen & van der Veldt, 2009). Judging from these benefits, it is not surprising that the Standish CHAOS Report (The Standish Group, 2009) – an annual report on software success and failure – identifies user involvement as the most important success factor for IT projects.

Research from two decades ago has shown that requirements were usually invented instead of elicited (Potts, 1993) and customers had little involvement in verifying requirements (Thayer & Dorfman, 1997). Nowadays, shortcomings in RE and user involvement still exist. While end-users are the most important stakeholders – they experience the system's quality (Kujala et al., 2005) – the SPM competence model (Bekkers, van de Weerd, Spruit, & Brinkkemper, 2010) depicts a narrow concept of customers and users by only showing an undefined connection between the company and its customers and market. In addition, involvement is often focused on early stage requirements elicitation (Hosseini, Phalp, Taylor & Ali, 2014b; Kujala et al., 2005; Lim & Finkelstein, 2012). This has two effects: other processes are still performed without user involvement (and therefore jeopardize achieved benefits from the elicitation phase) and the importance of requirements change management after product deployment is not acknowledged (Chemuturi, 2013; Nuseibeh & Easterbrook, 2000).

At the same time, the context of software usage is changing. Stakeholders are often geographically distributed (Cheng & Atlee, 2007) and technical and social environments are uncertain (Hosseini et al., 2014b). It is often hard to determine what kind of users will use a system, in which environment and on which device. Involving users who are not known in a context that is not known is problematic. The changing

context makes traditional involvement techniques expensive and complex and makes it harder to incentivize stakeholders to be involved.

An opportunity for user involvement in RE that might be applied to address these issues is *crowdsourcing*. Howe (2006) defined crowdsourcing as “the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call”. A famous application of crowdsourcing is Amazon Mechanical Turk (AMT), in which small tasks are performed by leveraging the human intelligence of a crowd of users. Similarly, tasks in the process of RE might be outsourced to stakeholders of the software product.

Little research has been conducted on the application of crowdsourcing in RE. Lim, Damian and Finkelstein (2011) have created StakeSource 2.0, a tool that uses “a crowdsourcing approach to identify and prioritize stakeholders and their requirements”. The StakeRare method (Lim & Finkelstein, 2012) uses this tool in a method that supports requirement elicitation. Hosseini, Phalp, Taylor and Ali (2014b) are in the early stage of researching the use of crowdsourcing in RE and more specifically in requirements elicitation. Adepetu, Ahmed, Abd, Zaabi and Svetinovic (2012) describe the crowdsourcing platform CrowdREquire, which supports requirements engineering. What all these approaches are missing is a way to motivate stakeholders to be and remain a member of the crowd.

A recent phenomenon that can be linked to crowdsourced RE and can increase both motivation and quality (Eickhoff, Harris, de Vries, & Srinivassan, 2012) is *gamification*. Deterding, Dixon, Khaled and Nacke (2011) have defined gamification as “the use of game design elements in non-game contexts”. The mechanism can be used to motivate users and increase user activity and retention. According to Gartner (2011), “by 2015, more than 50 percent of organizations that manage innovation processes will gamify those processes”. Fernandes et al. (2012) have made an attempt to apply gamification to requirements elicitation by developing the game-based collaborative tool ‘iThink’. The tool aids in collecting new requirements and gaining feedback on existing requirements. Two case studies indicate enhanced user involvement in requirements elicitation.

Table 1-1: A SWOT analysis of current RE techniques

Strengths	Weaknesses
<ul style="list-style-type: none"> • Stakeholder involvement in elicitation • RE improves software quality • User involvement has a high potential impact 	<ul style="list-style-type: none"> • Narrow concept of customers and users • Little room for requirements negotiation • Lack of user involvement in prioritization • Invisibility of requirements change management • Lacking incentives
Opportunities	Threats
<ul style="list-style-type: none"> • Crowdsourcing • Gamification 	<ul style="list-style-type: none"> • Geographically distributed stakeholders • Uncertain technical and social environments

Table 1-1 lists the discussed strengths and weaknesses of current RE techniques, as well as the opportunities and threats of the changing environment. The high potential of crowdsourcing and gamification is clear – the previous paragraphs briefly described their potential to involve and motivate stakeholders – but the best way to apply them still has to be found. Current applications are limited to requirements elicitation and are applied in an ad hoc manner. Cheng and Atlee (2007) address the need for a methodology that describes the use of RE techniques for multiple problems.

We therefore ask the following question:

“How can crowdsourcing be used to improve requirements engineering in software production?”

To answer this question, this thesis proposes and evaluates a method for Crowd-Centric Requirements Engineering (CCRE), which covers all phases and combines SPM, RE and crowdsourcing. Figure 1-1 displays this position in a Venn diagram where CCRE is the overlap of the other three areas. The positioning of the concepts is specific for this thesis; some elements of RE, SPM and crowdsourcing are taken into account in CCRE, while others are out of the scope. Examples of elements that are not part of CCRE are requirements specification languages (RE), portfolio management (SPM) and problem solving tasks (crowdsourcing). The relative size of the areas and their intersections does not represent their importance or relevance.

Gamification is used to make the method more effective. The CCRE method emphasizes the importance of high quality requirements and extensively involves users through crowdsourcing. Instead of being focused on the initial stage of software development, the method can be applied throughout the whole software development lifecycle and is agile itself.

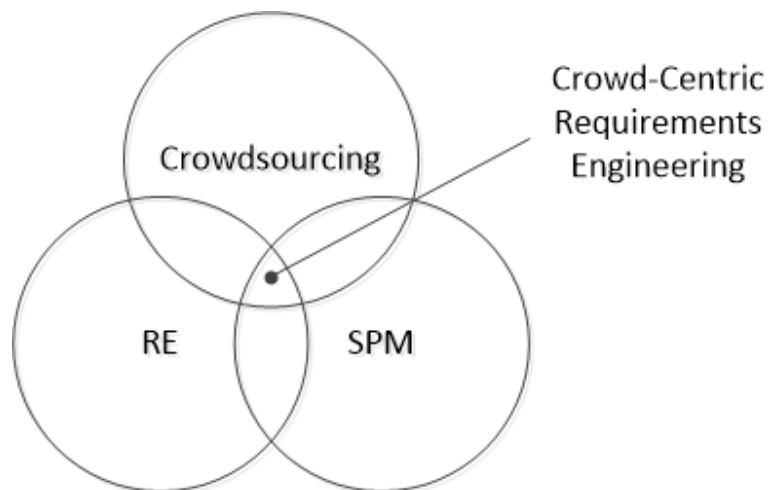


Figure 1-1: The domain positioning of the CCRE method

The CCRE method is relevant for the scientific community, by addressing the need for a well-structured crowdsourced method to perform the whole RE process, as well as the industry, by providing a means to involve users and eventually improving software quality. Evaluation of the method shows that stakeholders find it more useful and engaging than alternative methods, resulting requirements are useful and the transparent interaction gives a broad variety of perspectives. Some risks and suggestions for improvement are also identified.

This thesis is organized as follows. Chapter 2 describes the approach that is used to answer the research question. Chapter 3 builds the theoretical foundation for this study in the form of a literature review in the fields of RE, SPM, crowdsourcing and gamification. Expert interviews, which were used to gather input for the method, are outlined in Chapter 4. Chapter 5 describes the actual CCRE method. A prototype – *Refine* – that supports part of this method is introduced in Chapter 6. The demonstration of the method and prototype can be found in Chapter 7. Chapter 8 describes the evaluation of this demonstration. The thesis ends with conclusions, a discussion and opportunities for future work in Chapter 9.

2. Research Approach

In this chapter, the research approach of this thesis is outlined. The research questions are listed, the method is described and finally, the validity and reliability of this study are discussed.

2.1. Research Questions

The main research question of this thesis is formulated as follows:

RQ. *“How can crowdsourcing be used to improve requirements engineering in software production?”*

Six sub questions are identified. First, a clear understanding of RE and its elements needs in the relevant context to be given:

SQ 1. *What are the phases of requirements engineering in SPOs?*

As research in SPM shows, every SPO, development process and product environment is unique (Bekkers, Spruit, van de Weerd, van Vliet & Mahieu, 2010). A number of factors influence the execution of RE:

SQ 2. *What situational factors influence requirements engineering in SPOs?*

Throughout the process of RE, crowdsourcing might be applicable in different activities and ways. If crowdsourcing could be leveraged, the way in which it can be implemented is described:

SQ 3. *How can crowdsourcing support requirements engineering activities?*

The crowd needs to be motivated to perform certain tasks. Gamification is an emerging mechanism to engage and motivate people (Deterding et al., 2011). Its applicability to RE and the combination with crowdsourcing therefore has to be studied.

SQ 4. *To what extent does gamification help in increasing crowd involvement?*

The first four sub questions have focused on outlining the AS-IS situation and identifying opportunities for the TO-BE situation. The next step is the realization of this TO-BE situation and the implementation in practice.

SQ 5. *What are the activities and concepts of a crowd-centered method that supports requirements engineering for software products?*

In the end, the created method should lead to a higher quality of requirements engineering and thereby to a higher quality of software. Since a disadvantage of crowdsourcing is low quality input (Wais, Lingamneni, & Cook, 2010), significant attention needs to be given to this issue and the quality of the method's outcomes should be validated:

SQ 6. *To what extent can crowdsourcing increase the quality of requirements engineering?*

2.2. Research Method

Potts (1993) suggests that in software engineering research, “industry-as-laboratory” should be preferred above “research-then-transfer”. In “research-then-transfer”, laboratory research often leads to failure in addressing significant problems and undervaluation of technology transfer by researchers. In “industry-as-laboratory”, researchers closely collaborate with the industry. Supporting this rationale and aiming for utility, we decided to use design science as the research method.

As Hevner, March, Park and Ram (2004) state, design science “creates and evaluates IT artifacts intended to solve identified organizational problems”. Where behavioral science aims to find the truth, design science aims to create utility. March and Smith (1995) propose *building* and *evaluation* as design processes. In addition, they identified four artifacts that can be created with design science: *constructs*, *models*, *methods* and *instantiations*. Both the design processes are applied in this study. The CCRE method is the main artifact that is being built and evaluated.

In order to get more detailed guidance, the Design Science Research Methodology (DSRM) of Peffers, Tuunanen, Rothenberger and Chatterjee (2007) is followed. Figure 2-1 presents the six steps of the methodology. The following subsections describe how each of these steps is followed. Figure 2-2 shows a more practical visualization of the research method, with the deliverables on top and the activities at the bottom.



Figure 2-1: The DSRM process model (Peffers et al., 2007)

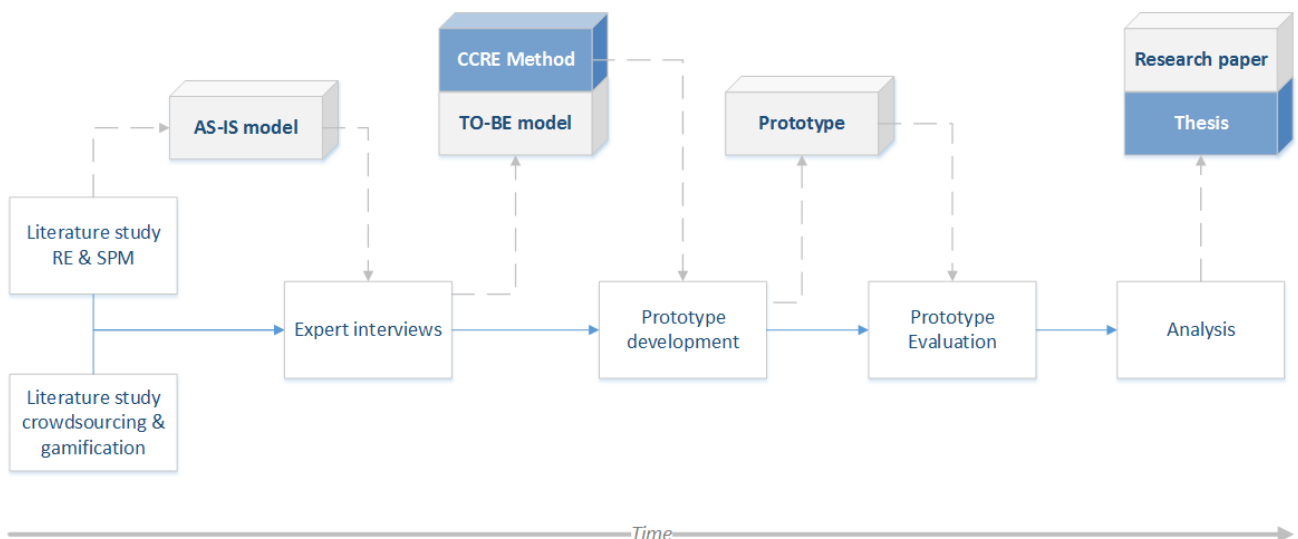


Figure 2-2: The research activities and deliverables

2.2.1. Problem Statement

In the introduction of this document, the problem that drives this study has been outlined. Requirements Engineering is an essential discipline with a big impact on software quality. In previous years, the importance and potential of user involvement became clear, but the current approaches have several shortcomings. Combining the RE perspective with SPM, we have identified the weaknesses of a narrow concept of customers and users, little room for requirements negotiation, a lack of user involvement in prioritization, the invisibility of requirements change management and lacking incentives. In an environment where stakeholders are geographically distributed and the environment of the user is uncertain, crowdsourcing and gamification have the potential to address these weaknesses and improve Requirements Engineering at Software Producing Organizations.

2.2.2. Objective

The overall objective of this study is to create and validate the Crowd-Centric Requirements Engineering method. Several sub-objectives are identified:

- **Scientific rationale should support the method** – Besides the practical contribution that we want to create for SPOs, we want to make a scientific contribution. A good scientific foundation is therefore essential;
- **The method needs to be agile and fit within an agile environment** – The people-centric and dynamic nature of agile development (Nerur & Balijepally, 2007) make it an ideal methodology to utilize in this study. Cao and Ramesh (2008) have studied several agile RE practices and found that the agile approach of these practices might lead to improved communication and understanding of customer needs. We want the CCRE method to be iterative and be useful in design-time RE as well as run-time RE, for which an agile environment is required;
- **The method should be adjustable** - Harmsen, Brinkkemper and Oei (1994) state that in reality, methods are tuned to a specific situation. The authors therefore describe the configuration of situational methods. The CCRE method is generic, but should be highly configurable. To aid in potential adjustment, we list a set of situational factors (SFs) and include an analysis of feasibility and context in the method.
- **The method has the ultimate goal to improve the quality of requirements engineering** – As Pacheco and Garcia (2012) state that “success and quality in software products depends to a great extent on requirements specification quality”, this is where our method should add value. This quality is not be dependent on the extent that a requirement adheres to a specified model, but rather on the coverage of an actual stakeholder need, the clarity of the requirement and the feasibility to convert this into a system functionality. In other words, the resulting requirements should be useful. Since the method is only successful when it can be continuously applied, the involvement and motivation of stakeholders is a second measurement for CCRE’s success.

2.2.3. Design & Development

In order to create the CCRE method, conceptual frameworks of the AS-IS and TO-BE situation of RE and SPM are created. The AS-IS framework emerged from a literature study and is a concise overview of the combination of RE and SPM. This literature study took a snowballing approach (Jalali & Wohlin, 2012). Relevant topics (i.e. Crowdsourcing, Requirements Engineering, Software Product Management) combined with suggestions from peers served as a starting point for finding papers. By forward snowballing, articles that cite the initial article were searched. Backward snowballing means that relevant articles were searched in the reference list of the initial article.

The TO-BE framework was also created based on the literature study but was used in the context of expert interviews, in which findings of the literature study were used as input for the questions. The TO-BE framework helped in the categorization of advices that were given by the experts. The categorized advices, complemented with knowledge created during the literature review, served as input for the CCRE method.

The CCRE method enables a SPO to operate in the TO-BE situation. Method Engineering (ME) was used in creating the method. Brinkkemper (1996) defines ME as “the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems”. The concept of ME is further described in Section 3.9.

2.2.4. Demonstration

To demonstrate the CCRE method, most of the method is instantiated and a prototype is developed to support a part that instantiation. The instantiation is presented as a case study. The prototype is not integrated in a software product, but has its own external environment. The prototype supports gamified elicitation, negotiation and prioritization of needs. The output of the prototype is a list of needs that were elicited, negotiated and prioritized by a set of stakeholders, who form the crowd. The needs will go through the subsequent phases of the CCRE method to be converted to a final list of requirements that can be put on a Product Backlog.

2.2.5. Evaluation

The demonstration of the CCRE method should lead to improved requirements engineering quality, which is in this thesis represented by a useful set of requirements (to improve the software product) and engaged stakeholders (to ensure continuity). No literature on the composition of RE quality was found, but the related studies from Lim and Finkelstein (2012) and Fernandes et al. (2012) evaluated stakeholder motivation, amusement, perceived ease of use (which add to engaged stakeholders), requirement completeness, accurate prioritization (which add to requirement usefulness) and usefulness of the approach (which adds to both engagement and requirement usefulness). Additionally, useful requirements can be seen as the logical output of a RE process and engaged stakeholders can be seen as an important determinant for feasibility and continuity.

Based on the two goals of engaged stakeholders and useful requirements, the results of the demonstration are evaluated in four ways (Figure 2-3).

First, users are asked about their experience after the demonstration. The motivation of the participants is measured as well as the perceived ease of use and usefulness of the prototype. This leads to qualitative as well as quantitative data, which provides insight in the participant perspective in the CCRE method.

Second, an observation of the demonstration leads to results about the engagement of stakeholders, the usefulness of requirements and the general feasibility and relevance of the method. By closely monitoring each activity and the interaction on the platform, we can determine whether the method works.

Third, the process and the list of CCRE-created requirements are discussed with the product management team of the case study organization. This provides the perfect perspective for measuring the perceived usefulness of the method and prototype, since these people have actually used the method to improve their software product.

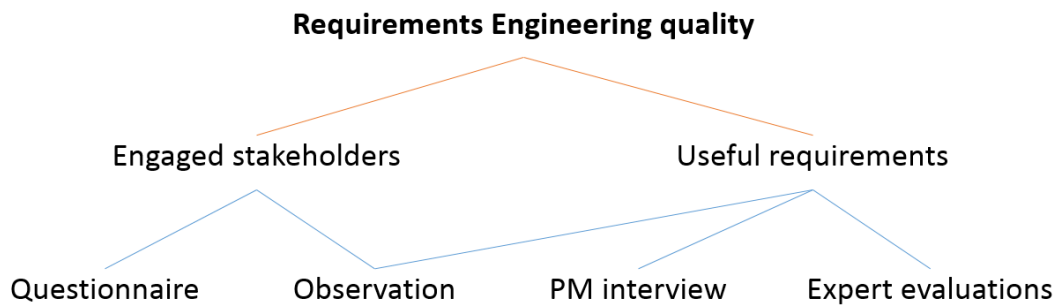


Figure 2-3: The relation of the four evaluation techniques to RE quality

Fourth, the method, prototype and samples of the outcome are presented to a group of experts from external organizations. The experts give feedback on the applicability of crowdsourcing and gamification and the quality of the resulting requirements. The results of this evaluation make clear whether the usefulness of the CCRE method can be generalized to Software Producing Organizations, or was specific for the evaluated context.

The evaluation provides suggestions for further research into Crowd-Centric Requirements Engineering and improvements to the prototype.

2.2.6. Communication

In Chapter 0 of the thesis, the results of the evaluation are presented. The expert discussion leads to a list of strong and weak points of the method. The user ratings and feedback lead to conclusions about the user experience and the CCRE method in general. Combining these findings, we determine the quality of the RE process.

Table 2-1 shows the mapping of the various research activities on the sub questions and thereby clarifies the relevance of the research activities. All evaluation techniques for the demonstration add value to the answers on sub questions 3 to 6. The corresponding section (8.5) will show which evaluation technique led to which findings.

In summary, the artifacts that are created are:

- Conceptual frameworks of the AS-IS and TO-BE situation;
- The CCRE method, which is the main artifact;
- A prototype that demonstrates the crowdsourced part of the CCRE method;
- A final thesis and research paper that describe the conducted research project.

Table 2-1: The mapping of the research activities on the sub questions

	Literature study	Expert interviews	Method Design	Demonstration
SQ 1	X	X		
SQ 2	X	X		
SQ 3	X	X	X	X
SQ 4	X	X	X	X
SQ 5		X	X	X
SQ 6		X	X	X

2.3. Validity & Reliability

In order to make a high-quality contribution to both science and society, the validity and reliability of the study has to be guaranteed. This section describes the important criteria and which decisions are made to meet these criteria.

2.3.1. Internal validity

Internal validity refers to the authenticity of causal relationships. As described by Brewer (2000), variation in the dependent variable should be caused by the independent variable and not by other forces. In our study, this principle is applied to determining the quality of requirements engineering and assuring that this quality is improved by the CCRE method. While it will always remain impossible to isolate the method from an external and influencing context, the context is described as precise as possible and the method is carried out rigorously. This reduces the chance of unreported coincidences that influence the findings.

2.3.2. External validity

External validity deals with the generalizability of results. By clearly defining the domain in which the results can be expected (SPOs) and clearly explaining the effects of situational factors, we strive to achieve this type of validity. Additionally, we interview the external experts and explicitly ask them for the contexts in which the method would and would not be useful.

2.3.3. Reliability

Reliability emphasizes the importance of consistency and repeatability of research. A clear description of the complete research process is given to ensure reliability. In creating the method, the method requirements and their origin are clearly documented and traceable through codes. In the demonstration, the context and prototype are outlined. Getting the same results in a repetition will however be impossible, since the created artefact and related outcomes are dependent on situational factors. This is an essential part of the study.

3. Theoretical Foundation

As a first step in this research, the theoretical fundamentals of Requirements Engineering and Software Product Management have been studied in order to create an overview of the AS-IS situation. The following sections include this literature study, which leads to a combination of both research fields in Section 3.3.

3.1. Requirements Engineering

According to Harwell, Aslaksen and Hooks (1993), a requirement is a ‘thing’ that mandates that something must be accomplished, transformed, produced or provided. The description in the Software Engineering Body of Knowledge (SEBoK) is similarly abstract: “A property that must be exhibited by something in order to solve some problem in the real world” (Bourque & Fairley, 2014, pp. 32–33). Sommerville (2011) differentiates between 1) *user requirements*, which are statements of what services the system is expected to provide and under which constraints it must operate and 2) *system requirements*, which are more detailed descriptions of the system’s functions, services and operational constraints. A second distinction is made between 1) *functional requirements*, which focus on the services and functions of the system and 2) *non-functional requirements*, which are constraints on those services and functions.

3.1.1. Definitions

The development of these requirements is studied in the field of Requirements Engineering (RE) A universally accepted definition of RE is missing, but several definitions can be found in literature:

- “The science and discipline concerned with establishing and documenting software requirements” (Thayer & Dorfman, 1997);
- “The systematic process of developing requirements through an iterative co-operative process of analyzing the problem, documenting the resulting observations [...] and checking the accuracy of the understanding gained” (Loucopoulos & Karakostas, 1995);
- “The process of discovering [the] purpose [of a software system], by identifying stakeholders and their needs, and documenting these in a form that is amenable to analysis, communication, and subsequent implementation” (Nuseibeh & Easterbrook, 2000, p. 1). The authors also emphasize the importance of describing the system’s environment;
- “The identification of the goals to be achieved by the envisioned system, the operationalization of such goals into services and constraints, and the assignment of responsibilities for the resulting requirements to agents such as humans, devices, and software” (Lamsweerde, 2000, p. 5);
- “The branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families” (Zave, 1997, p. 315).

The five definitions have very different scopes and list a variety of elements, which we have ordered in a logical manner:

- Identification of stakeholders, needs and goals;
- Analysis of problems and the system’s environment;
- Operationalization of goals into services and constraints;
- Establishment of requirements;
- Assignment of (human and system) responsibilities;

- Relations between goals, functions and constraints to software specifications;
- Documentation of needs and requirements;
- Communication;
- Verification;
- Implementation;
- Evolution over time and across software families.

Pohl (2013) classifies RE as an early stage of the software development life cycle. However, RE can be relevant in software maintenance, which can be classified in fault repairs, environmental adaptations or functionality addition (Sommerville, 2011). In this thesis, we also approach Requirements Engineering as a continuous activity, which is conducted before, during and after the development of a software product.

We do not use a specific definition of RE and do not attempt to develop our own definition, since that is outside the scope of this thesis.

3.1.2. Activities

In their book on Software Requirements Engineering, Thayer and Dorfman (1997) identify five processes in RE:

1. Software requirements elicitation – Discovering, reviewing, articulating and understanding the users' needs and the constraints on the software and its development;
2. Software requirements analysis – Analyzing these needs to define the software requirements;
3. Software requirements specification – Developing a document that precisely describes each of the requirements;
4. Software requirements verification – Ensuring that the specification is consistent with the requirements, adheres to standards and is useful;
5. Software requirements management – The planning and controlling of the previous four activities.

Other researchers have not always agreed with this set of processes, which are also named *tasks* and *activities*. Table 3-1 shows the various interpretations of the activities within RE. Within the iterative activity of elicitation and analysis, Sommerville (2011) lists the sub-activities discovery, classification & organization, negotiation and documentation. Elicitation is an activity that comes back in all descriptions, while other activities (e.g. feasibility study, documentation) are only mentioned by one of the reviewed papers.

Table 3-1: The activities within RE

Thayer and Dorfman (1997)	Nuseibeh and Easterbrook (2000)	Lamsweerde (2000)	Cheng and Atlee (2007)	Sommerville (2011)
<i>Processes</i>	<i>Activities</i>	<i>Activities</i>	<i>Tasks</i>	<i>Activities</i>
Elicitation	Elicitation	Domain analysis	Elicitation	Feasibility study
Analysis	Modelling & analysis	Elicitation	Modeling	Elicitation & analysis
Specification	Communication	Negotiation & agreement	Analysis	Specification
Verification	Agreement	Specification	Validation & verification	Validation
Management	Evolution	Specification analysis	Management	Management
		Documentation		
		Evolution		

In general, we see that the RE process optionally starts with a contextual analysis, which is followed by requirements elicitation. The resulting requirements are subject to some form of analysis, which might be followed up by negotiation and communication with stakeholders. The requirements are subsequently well-formulated in a specification, which is validated or verified in varying ways. After optional documentation of the process, evolution of requirements is managed. State-of-the-art research on the processes is described in Section 3.3.

According to the authors of the SEBoK, a systematic breakdown of activities creates the risk of a waterfall-like process. While the elicitation, analysis, specification and validation are outlined, a successful requirements process “must be considered as a process involving complex, tightly coupled activities (both sequential and concurrent), rather than as a discrete, one-off activity performed at the outset of a software development project” (Bourque & Fairley, 2014, p. 32).

3.2. Software Product Management

A second perspective from which the development of software requirements can be perceived is Software Product Management (SPM). Ebert (2007, p. 850) defines SPM as “the discipline and role, which governs a [software] product (or solution or service) from its inception to the market/customer delivery in order to generate biggest possible value to the business.” The author states that good product management has a substantial impact on the product’s success. While the research field is very young and immature (Maglyas, Nikula, & Smolander, 2011), Ebert and Brinkkemper (2014) link the excellence of product management to the difference in business success between companies like Netscape and Nokia on the one hand, and Apple, Google and SAP on the other. SPM is a complex task, due to the complexity of products, large variety of stakeholders, long lists of requirements and rapidly changing environment (Vlaanderen, Jansen, Brinkkemper, & Jaspers, 2011).

The structure of SPM was visualized in the reference framework of van de Weerd, Brinkkemper, Nieuwenhuis, Versendaal and Bijlsma (2006), further refined into the SPM competence model by Bekkers, van de Weerd, Spruit and Brinkkemper (2010). The SPM competence model is displayed in Figure 3-1. This model consists of four business functions: Portfolio management, Product planning, Release planning and Requirements management. Each of the functions has several focus areas, which are represented by the smaller boxes within the business functions.

Table 3-2 lists the business functions and focus areas in SPM that are relevant to RE and shows the definitions according to Bekkers, van de Weerd et al. (2010). The model and corresponding definitions make the difference with RE apparent. SPM spans the entire product-life cycle, connects with other business functions (e.g. marketing, support) and distinguishes the portfolio, product and release layers to allow for iterations. RE is mainly focused on initial requirements development, is described as a linear process and stops at detailed documentation (Thayer & Dorfman, 1997) or implementation in a specific product release (Nuseibeh & Easterbrook, 2000).

Recently, Ebert and Brinkkemper (2014) have interviewed fifteen companies in order to identify success factors of SPM. The top four consisted of:

- The existence of a core team with product, marketing, projects and operation managers that carry accountability for success;
- A standardized product life-cycle with clear criteria;
- Understanding of the customer voice and a clear business vision, including well-documented requirements that allow for technical, market and business judgment;

- Portfolio management and roadmapping, containing risk management and communication.

All these success factors could be useful in improving the requirements-related aspects of SPM and are therefore relevant for this study.

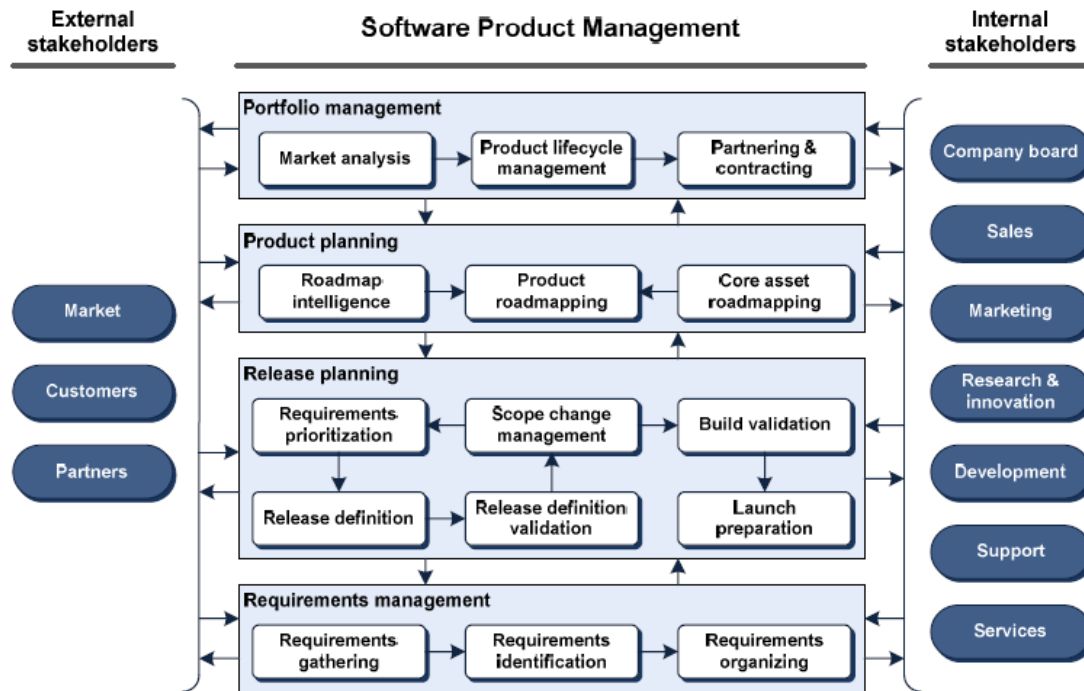


Figure 3-1: The SPM competence model (Bekkers et al., 2010)

Table 3-2: The RE-related concepts of SPM and their definitions (Bekkers, van de Weerd et al., 2010)

Concept	Definition
Requirements management	The continuous management of requirements outside of releases.
Requirements gathering	Acquisition of requirements from both internal and external stakeholders.
Requirements identification	Rewriting the Market Requirements to understandable Product Requirements and connecting similar requirements.
Requirements organizing	Structure requirements throughout their entire lifecycle and describe dependencies between Product Requirements.
Release planning	Capabilities needed to successfully create and launch a release.
Requirements prioritization	Prioritize identified and organized requirements
Release definition	Select requirements that will be implemented in the next release and create a release definition based on the selection.
Release definition validation	Validate the release definition internally.
Scope change management	Handle different kinds of scope changes during the release development.
Build validation	Validate the built release before the launch.
Product planning	The gathering of information for, an creation of a roadmap for a product or product line and its core assets.
Product roadmapping	The actual creation of the product roadmap.

Due to the immaturity of SPM, many research opportunities still exist. In a systematic mapping study on the existing knowledge on SPM, Maglyas, Nikula, and Smolander (2011) find only 25 articles and argue that SPM knowledge is fragmented and the definition is not well established.

3.2.1. Product software and Software Producing Organizations

Closely related to the concept of SPM are the notions of product software and Software Producing Organizations (SPOs). Xu and Brinkkemper (2007, p. 534) define product software as “a packaged configuration of software components or a software-based service, with auxiliary materials, which is released for and traded in a specific market”. It is different from tailor-made software, microprograms and embedded software because it has many copies and is not complementing an appliance that is being sold (Table 3-3). Other terms for software products are packaged, commercial, shrink-wrapped and commercial-off-the-shelf (COTS) software (Sawyer & Guinan, 1998).

Table 3-3: the classification of different types of software (Xu & Brinkkemper, 2007)

Number of copies	One	Many
What is sold?		
Appliance	Micro-program	Embedded software
Software	Tailor-made software	Product software

A RE-related difference between the off-the-shelf product software and tailor-made software for a specific client, is that the former is offered through releases, which requires release planning and requirements prioritization (Regnell, Höst, Natt och Dag, Beremark, & Hjelm, 2001). As mentioned earlier, the list of requirements for software products is relatively long (Vlaanderen et al., 2011).

Since the difference between traditional producer-consumer RE and product software RE requires an alternative approach, market-driven RE (MDRE) has been introduced (Gorschek, Gomes, Pettersson, & Torkar, 2012; Regnell & Brinkkemper, 2005). An MDRE context contains a continuous flow of requirements, which is not only limited to the initial development phase. Because of this new market perspective, user involvement is becoming an important topic in RE literature and is discussed in Section 3.5. The gathering of requirements potentially leads to an enormous amount of information. Regnell and Brinkkemper (2005) improve efficient and effective decision-making by providing a process quality model, a requirements state model and an example of an MDRE repository. The states that are identified are Candidate, Approved, Specified, Discarded, Planned, Developed, Verified and Released.

The emergence of MDRE is consistent with the evolution of agile RE (Cao & Ramesh, 2008), which is user-centric and deals with the constant change of requirements. A more elaborate description of agile RE is given in Section 3.4.2.

3.3. Combining RE and SPM

Table 3-4 describes a few pros and cons of RE and SPM. Requirements Engineering is a mature field of research and each of its activities have been studied relatively extensively. However, some of this research is not pragmatic; studies on formal requirements specification describe complex schemas that will hardly be used in practice (e.g. Spivey, 1989). The field of SPM is less mature (Maglyas et al., 2011), but embeds processes that are relevant for RE in the whole product management life cycle. Additionally, a maturity matrix that makes it possible to determine proficiency and areas to improve for a SPO has been developed. A

negative effect of the immaturity is the brief or ambiguous definition of concepts and relations in the field. The two descriptive frameworks that have been developed (Bekkers, van de Weerd, et al., 2010; I. van de Weerd et al., 2006) are lacking clear explanations and examples of the focus areas and their relations.

Table 3-4: The pros and cons of RE and SPM

Requirements Engineering	Software Product Management
+ Mature field	+ Embeds requirements in the SPO
+ Detailed explanation of activities	+ Describes maturity levels for focus areas
- Not always pragmatic	- Ambiguous concepts

Since we want to take advantage of both the RE and SPM perspective in this study, the models of the two research fields are combined. Figure 3-2 presents this model. The combination allows us to analyze well-explained activities from the perspective of a SPO.

Table 3-5 explains the definitions of the model's elements. With this model, we strive to combine the best of the two research fields, without adding external concepts and ideas. A number of design decisions were made in the development of this model and are rationalized in this section.

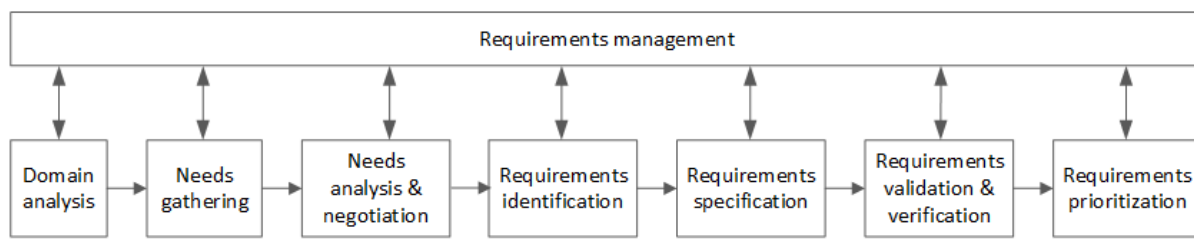


Figure 3-2: Visualization of RE and SPM combined, based on current literature

Table 3-5: Explanation of RE and SPM combined, based on current literature.

Activity	Definition	Origin
Req. management	Continuous management (planning, controlling and organizing) of needs and requirements.	RE+SPM
Domain analysis	Evaluating existing systems, opportunities, objectives and feasibility.	RE
Needs gathering	Acquisition of needs from all stakeholders.	SPM
Needs analysis & negotiation	Reviewing and understanding user needs and constraints. Articulating the needs in Market Requirements.	RE
Req. identification	Rewriting the Market Requirements to Product Requirements.	SPM
Req. specification	Documenting the detailed system requirements.	RE
Req. validation & verification	Ensuring that the Product Requirement covers the need and is consistent with standards.	RE
Req. prioritization	Prioritize verified and organized requirements	SPM

- In RE literature, requirements management is one of the linear processes. In SPM literature, it is the business function that contains several focus areas. We approach Requirements Engineering as the business function, in which requirements management is a planning, structuring and organizing management function. It therefore interacts with all of the other activities, but is not operational itself. Managing change and traceability should be an integral part of this activity;
- We decided to distinguish *needs* and *requirements*. A need can be defined as a raw and vague request from the perspective of an individual stakeholder, a requirement is nuanced, re-articulated and a more

specific thing that needs to be accomplished for the market. This distinction is consistent with Kujala, Kauppinen and Rekola (2001). A requirement can be either from the perspective of the market, i.e. the set of stakeholders (Market Requirement) or the system (Product Requirement). A clarification of this distinction is given in Table 3-6;

Table 3-6: The difference between needs, market requirements and product requirements

Term	Perspective	Example
Need	Individual stakeholder	"I've spent minutes looking for the cancel button"
Market requirement	Market	"The cancel button should be easier to find"
Product requirement	System	"The cancel button should be positioned to the right of the submit button"

- Needs gathering is not just elicitation for needs, but includes the collection of needs from other sources;
- Needs analysis and negotiation are combined in one activity, consistent with Cheng and Atlee (2007). Analysis has the goal to review and understand the customers' needs. In this, negotiation (and eventually agreement) is an important means;
- Requirements prioritization is part of the RE process and not of a subsequent release planning function. We consider the activity essential in determining a system's goals, functions and constraints, especially regarding the evolution of requirements.

In the following sections, the literature that explains each of these activities is outlined.

3.3.1. Requirements management

Paulk (1993) acknowledges requirements management as one of the *Key Process Areas* in the Capability Maturity Model for Software. According to the author, requirements management has the purpose to establish a common understanding between a customer and a software project. Comparing this purpose with more recent definitions, the description seems outdated. In SPM, requirements management is defined as a business function that comprises "the continuous management of requirements outside of releases" (Bekkers, van de Weerd et al., 2010, p. 4). The business function consists of three focus areas: requirements gathering, requirements identification and requirements organizing. It thereby takes a significantly different perspective from RE literature.

Nuseibeh and Easterbrook (2000) identify requirements management as a crucial ability to make requirements readable and traceable and thereby manage their evolution over time. This description is strongly related to requirements organization. Sommerville (2011, p. 112) defines requirements management as "the process of understanding and controlling changes to system requirements". Inherent phases that are identified include planning and change management. The SWEBoK (Bourque & Fairley, 2014) does not list requirements management as an activity, but includes requirements change management and requirements tracing as "practical considerations", together with the iterative nature and the measurement of requirements.

While there thus exists some disagreement on the exact definition, the goal of requirements management is to plan and control the evolution of requirements. An example of this activity is depicted in Figure 3-3, where the evolution is represented by the status and an person is assigned as a control measure.

Name	ID	Status	Development Status	Assigned	JIRA URL
Steve Test JIRA	BR-6	New		Bert Simpson	
Steve test business req	BR-7	Draft		Homer Simpson	
Initiate a race	BR-2	Approved for Development		Bert Simpson	
Day of the Race	BR-4	Approved for Development		Homer Simpson	
Ability to set goals	BR-3	Approved for Development		Bert Simpson	

Figure 3-3: An example of requirements management in Jama software¹

3.3.2. Domain analysis

While domain analysis has not been mentioned in all the literature on RE activities, it is a useful activity. Lamsweerde (2000) lists the elements of domain analysis:

- A study of the existing system in which the software is to be built;
- Identifying and interviewing relevant stakeholders;
- Identifying problems and opportunities in the existing system;
- Goals of the new system are determined.

A similar activity is described by Sommerville (2011) as a *feasibility study*, which should determine whether 1) the new system contributes to the organization's objectives, 2) it can be implemented within schedule and budget and 3) it can be integrated with other systems. Judging from these factors, the feasibility study is most effective when the scope and requirements of the new system are at least partly known. Therefore, a domain analysis is more relevant as the first phase of RE.

3.3.3. Needs gathering

In traditional RE literature, this activity is often referred to as requirements elicitation. According to Cheng and Atlee (2007), it comprises "activities that enable the understanding of the goals, objectives, and motives for building a proposed software system" and also involves the identification of requirements that must be satisfied in order to meet those goals (p. 3). Many techniques for eliciting requirements exist. Nuseibeh and Easterbrook (2000) identify six: traditional elicitation, group elicitation, prototyping, model-driven, cognitive and contextual techniques. Each of the techniques has its own advantages, which seem complementary. Integration of different techniques is therefore advised. The authors additionally describe the identification of stakeholders and tasks as several important aspects of eliciting requirements. Pacheco and Garcia (2012) review literature on stakeholder identification methods and give a number of best practices: 1) assign roles to stakeholders, 2) establish constructive interactions between the stakeholders (and the system) and 3) classify requirements in relation to the project goal.

In SPM, this activity is named requirements gathering, which has a similar definition. The SPM Maturity Matrix (Bekkers, van de Weerd et al., 2010) ranks the corresponding capabilities from basic registration to advanced customer and partner involvement, where requirements are systematically gathered from internal and external stakeholders and status updates are provided.

¹ <http://www.jamasoftware.com/>, retrieved on January 28th, 2015



Figure 3-4: Offline (paper prototyping) and online (UserVoice) needs gathering

3.3.4. Needs analysis & negotiation

At the start of an RE process, knowledge about a system's functionality is vague, originating from personal views and represented in a raw and informal format (Pohl, 2013). Analysis is needed to make the data more meaningful. Chemuturi (2013) describes analysis as reducing complexity by breaking something down into separate parts and thereby creating an improved insight. For requirements, analysis means verifying completeness, evaluating feasibility, grouping and identifying gaps. Kabbedijk, Brinkkemper, Jansen and van der Veldt (2009) distinguish the same four types of analyses, but add complexity analysis. Cheng and Atlee (2007) add analyses for unknown requirements interactions, obstacles to requirements satisfaction and missing assumptions.

Nuseibeh and Easterbrook (2000) relate requirements analysis to requirements modelling, which they define as "the construction of abstract descriptions that are amendable to interpretation" (p. 4). Hofmann and Lehner (2001) state that models can give stakeholders feedback on the interpretation of their requirements, in order to create an understanding. The outcome of the needs analysis should be well-articulated market requirements and therefore includes modeling.

Requirements models can be used to gather further information and are therefore useful input for negotiation. As Nuseibeh and Easterbrook (2000) note, requirements documentation is essential in effective communication. According to Lamsweerde (2000), the step of negotiation is used to compare alternative requirements and evaluate risks.

3.3.5. Requirements identification

During requirements identification, the market requirements that resulted from the needs analysis are converted to product requirements (Bekkers, van de Weerd et al., 2010). The activity originates from SPM, where it also involves connecting requirements with similar functionality. The naming of the activity is unfortunate, since other studies relate it to requirements gathering (e.g. Lim and Finkelstein (2012)). Further research of the activity in the realm of SPM is non-existent. Hull, Jackson and Dick (2011) describe a process to convert a set of needs into specified system requirements. First, a vague set of user needs can be transformed into a set of stakeholder requirements. Subsequently, system requirements are developed by determining the system's characteristics irrespective of the exact final design. Finally, the system's architecture can be used to develop component-specific requirements.

3.3.6. Requirements specification

Zave's (1997) definition of RE, which mentions precise specifications of software behavior, highlights the importance of this activity. Three other definitions (Loucopoulos & Karakostas, 1995; Nuseibeh & Easterbrook, 2000; Thayer & Dorfman, 1997) that were presented in Section 3.1.1, explicitly name 'documenting' as an element of RE. Nuseibeh and Easterbrook (2000) state that RE "represents a series of engineering decisions that lead from recognition of a problem to be solved to a detailed specification of that problem" (p. 2). Pohl (2013) explains that a complete system specification is the basic result of the RE process. The specification should be agreed upon and should be written in a formal language. In MDRE, a specification is given in the form of a repository. An outline of such a repository is presented in Figure 3-5 (Regnell & Brinkkemper, 2005).

<i>Attribute</i>	<i>Value</i>	<i>Assigned in State</i>
State	C / A / S / Di / P / De / V / R	-
ID	Unique identity	Candidate
Submitter	Who issued it?	Candidate
Company	Submitter's company	Candidate
Domain	Functional domain	Candidate
Label	Good descriptive name	Candidate
Description	Short textual description	Candidate
Contract	Link to sales contract enforcing requirement	Candidate
Priority	Importance category (1,2,3)	Approved
Motivation	Rationale: Why is it important?	Approved
Line of Business	Market segment for which requirement is important	Approved
Specification	Links to Use Case, Textual Specification	Specified
Decomposition	Parent-of / Child-of – links to other req's	Specified
Estimation	Effort estimation in hours	Specified
Schedule	Release for which it is planned for	Planned
Design	Links to design documents	Developed
Test	Links to test documents	Verified
Release version	Official release name	Released

Figure 3-5: The outline of a MDRE repository (Regnell & Brinkkemper, 2005)

From these studies, we can derive that requirements specification is a valuable activity in RE. The activity overlaps with requirements identification, since they both aim to (re)formulate requirements to a systems perspective. Formal languages to write requirements specifications include Z (Spivey, 1989) and Two-Level Grammar (Bryant & Lee, 2002).

3.3.7. Requirements validation & verification

When requirements are being developed, two things should be determined. First, the resulting requirements should actually solve the user’s problem, which can be checked with validation. Second, the specification should correctly describe the requirement, which can be checked with verification. According to Cheng and Atlee (2007), validation is a subjective evaluation of the specification involving informally described requirements. Instead of focusing on the description in a specification, validation is about checking whether the initial need is met.

SPM seems to ignore requirements verification, but instead talks about ‘release definition validation’. This is concerned with *internally* approving a release definition, before the software is built. Users or customers are thus not involved in this process. Chemuturi (2013) writes about similar ‘approval’ of requirements documents by internal sources, but also involves the client organization. Hull et al. (2011) go into more detail and describe the V-Model (Figure 3-6) that visualizes four tests for the validation of four types of requirements. The acceptance test is validation the results for stakeholders, the system test is validating the functionality of the system.

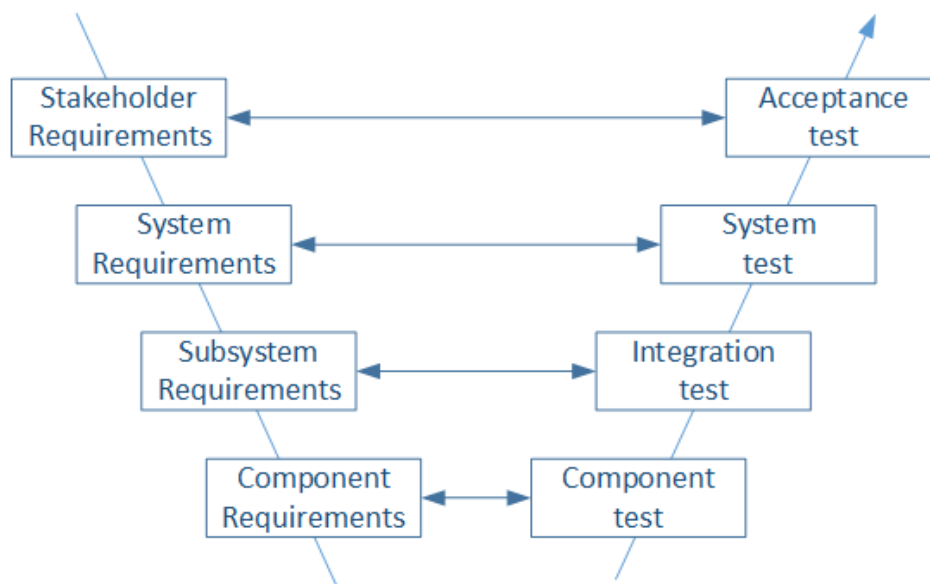


Figure 3-6: Requirements in the V-Model (Hull et al., 2011)

3.3.8. Requirements prioritization

Requirements prioritization is about assigning priority to and thereby ranking product requirements. Prioritizing is essential since projects have resource limitations and might generate conflicting requirements (Wieggers, 1999). According to Carlshamre (2002), the value and costs of a requirement are determinants for its priority. It is also important to take interdependencies into account, since a high percentage of requirements is interdependent (Carlshamre, Sandahl, Lindvall, Regnell, & Natt och Dag, 2001).

Various ways to prioritize requirements exist. Wieggers (1999) presents a simple scheme that uses the requirements’ value, cost and risks to determine relative priorities. Another famous method is the MoSCoW method (Stapleton, 1997), which categorizes requirements into four groups: ‘must have’, ‘should have’, ‘could have’ and ‘want to have, but will not have this time round’.

Berander and Andrews (2005) argue that the simplest appropriate technique should be chosen and only use more sophisticated techniques when disagreements need to be solved or critical decisions need to be made.

The simplest technique that is identified by the authors is a *top ten*, in which stakeholders pick their ten favorite requirements and do not assign an internal order.

3.4. The context of RE

Knowing the current theoretical situation of RE and SPM, we should take notice of the factors that can influence the processes and how an organization can respond to these factors. Section 3.4.1 describes the literature on situational factors (SFs) to which RE methods have to be adjusted. Subsequently, Section 3.4.2 describes the emergence of agile RE, as an approach to cope with changing requirements.

3.4.1. Situational Factors

Clarke and Connor (2012) acknowledge that in software development, one size does not fit all and argue that situational contexts should be taken into account when constructing a software development process. The authors therefore present a set of 44 situational factors that influence software processes. RE can also be surrounded by a myriad of contexts. Nuseibeh and Easterbrook (2000) state that requirements modelling and analysis should not be isolated from the organizational and social context of the system. The specific activities of RE and the way in which these activities are executed depend on situational factors.

Souer, van de Weerd and Versendaal (2007) have constructed a situational RE method for the development of content management systems, but only implement situationality by offering a decision of one of the three different project types. Gorschek et al. (2012) present a market-driven RE process model that offers assessment capabilities in order to fit it to a company's specific situation. The authors describe that requirements are selected based on criteria as market size, risks and politics, but also the product roadmap, dependencies and initial priorities.

In the situational assessment method for SPM, situational factors are described as factors that “describe the situational context in which [...] the product manager has to operate and to which the SPM processes thus have to [be] fine-tuned” (Bekkers, Spruit, van de Weerd, van Vliet & Mahieu, 2010, p.4). While our scope is broader than just SPM, situational factors have the same relevance. All situational factors identified by Bekkers (2012) affect RE. Therefore, Table 3-7 lists these situational factors. Additionally, the *degree of customization* (generic, customized or customer-specific) could also be considered as a situational factor, since it determines the applicability of SPM and MDRE (Regnell & Brinkkemper, 2005).

Table 3-7: Situational factors in SPM (Bekkers et al., 2010)

Development philosophy	New requirements rate
Size of organization	Number of products
Customer loyalty	Product age
Customer satisfaction	Product lifetime
Customer variability	Product size
Number of customers	Product tolerance
Type of customers	Company policy
Market growth	Customer involvement
Market size	Legislation
Release frequency	Partner involvement
Sector	
Standard dominance	<i>Degree of customization</i>
Variability of feature requests	<i>System complexity</i>
Number of localizations	<i>Geographical distribution of customers</i>

Defects per year

Regnell, Svensson and Wnuk (2008) state that an increasing size and complexity of software systems causes large and complex sets of requirements. The authors identify four orders of magnitude, from small-scale RE (10 requirements) to very large-scale RE (10,000 requirements). This statement supports the situational factors *new requirements rate* and *product size*, but also adds the factor *system complexity*. A final situational factor is derived from the paper of Holmström (2004), who identifies that *geographic distribution of customers* might enlarge the problems of eliciting changing user needs. The additional situational factors are presented in italics in Table 3-7.

3.4.2. Agile Requirements Engineering

Agile software development was introduced through the Manifesto for Agile Software Development (Agile Alliance, 2001), which highlighted the importance of individuals & interactions, working software, customer collaboration and responding to change in software development. A wide variety of agile methods have evolved, including Extreme Programming (XP) (Beck, 2000) and Scrum (Schwaber, 1997).

According to Cao and Ramesh (2008), agile methods proceed to the development of code without waiting for formal RE processes. XP explicitly discourages conducting “complete up-front analysis and design” (Beck, 2000, p. 1). Cao and Ramesh (2008) researched the RE practices that agile developers follow and found that the development of complete requirements specifications is impossible or inappropriate in agile environments. Instead, the practices of face-to-face communication, iterative RE and constant customer-centric (extreme) prioritization were considered most important. However, these practices each have some challenges. Table 3-8 lists the benefits and challenges of the three best practices.

Table 3-8: Benefits and challenges of Agile RE best practices (Cao & Ramesh, 2008)

	Benefits	Challenges
Face-to-face communication	<ul style="list-style-type: none"> • Unanticipated directions • Obviates time-consuming documentation 	<ul style="list-style-type: none"> • Ineffective communication leads to inadequate requirements • Depends on trust and consensus
Iterative RE	<ul style="list-style-type: none"> • Better customer relationship • Clearer requirements 	<ul style="list-style-type: none"> • Cost and schedule estimation • Minimal documentation • Ignored non-functional requirements
Extreme prioritization	<ul style="list-style-type: none"> • Business reasons for each requirement at any cycle • Opportunity for reprioritization 	<ul style="list-style-type: none"> • Business value as only criterion potentially creates problems • Continuous reprioritization lead to instability

3.4.2.1. Overcoming the challenges

While challenges in agile RE thus have been identified, most are not insurmountable. Grünbacher and Hofer (2002) research requirements negotiation as a technique to complement XP. The authors describe a requirements negotiation methodology leading to emphasis of shared vision, more complete stakeholder identification, a full perspective for on-site customer and extensive stakeholder involvement in decision-making. This negotiation could thereby address some of the challenges from Table 3-8, especially by improving face-to-face communication. Nawrocki, Jasinski, Walter and Wojciechowski (2002) argue for having testers/analysts document requirements and linking them to test cases, which would improve iterative RE practices. In addition, the authors propose to include an unusual RE phase in the XP lifecycle in order to determine goals, constraints and feasibility.

For requirements prioritization, *The Planning Game* is proposed by Beck (2000). In this game, both business people and developers categorize requirements into groups, respectively based on value and risk. In later phases, the requirements within the groups are further prioritized by customers (Newkirk & Martin, 2000).

3.4.2.2. Agility in SPM

The applicability of agile methods to SPM has also been studied. Vlaanderen, Jansen, Brinkkemper and Jaspers (2011) focused on Scrum principles and shows that requirements can be refined from visions to themes, concepts and eventually requirements definitions on a product backlog (Figure 3-7). Items on the Product Management Sprint Backlog come from the Product Backlog and need more specification before they are included on the Development Sprint Backlog. The Product Management Sprints alternate with the Software Development Sprints (Figure 3-8), to ensure that every sprint can start with a complete and fully specified sprint backlog. While this is already a better integration of RE and agile then RE literature provides, Maglyas et al. (2011) criticize the paper for only covering development activities, while leaving other activities such as marketing, customers support, and strategy formation out of its scope.

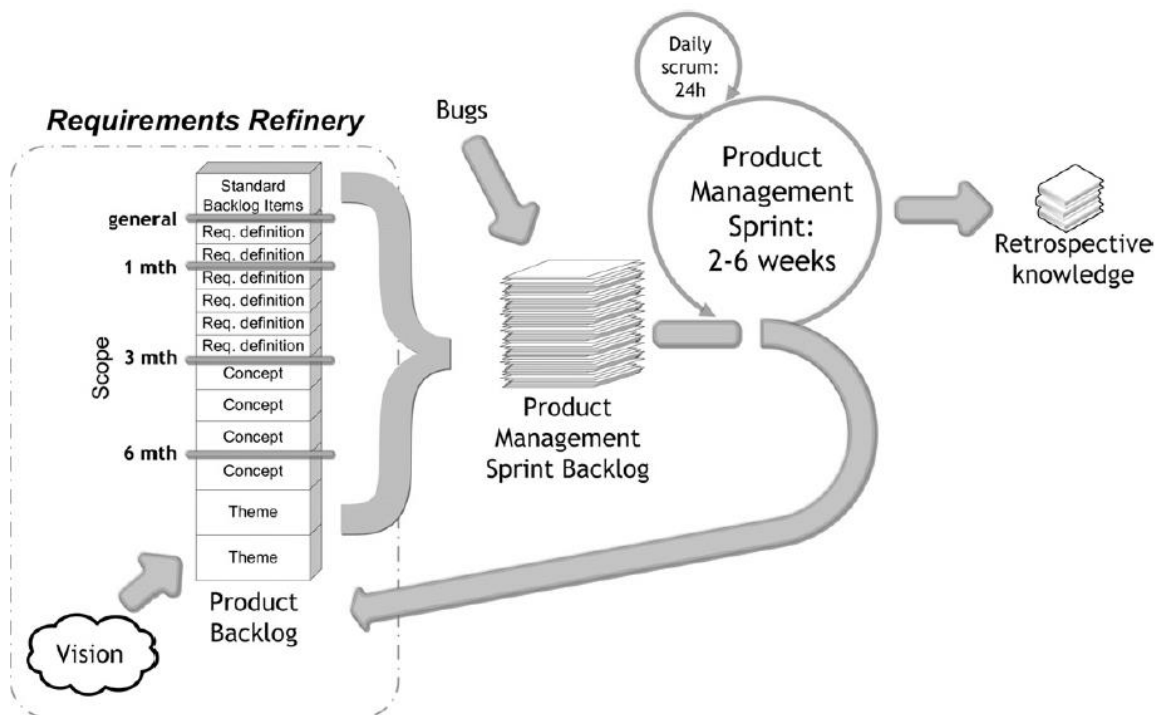


Figure 3-7: The refinery from a vision towards a standard backlog item (Vlaanderen et al., 2011)

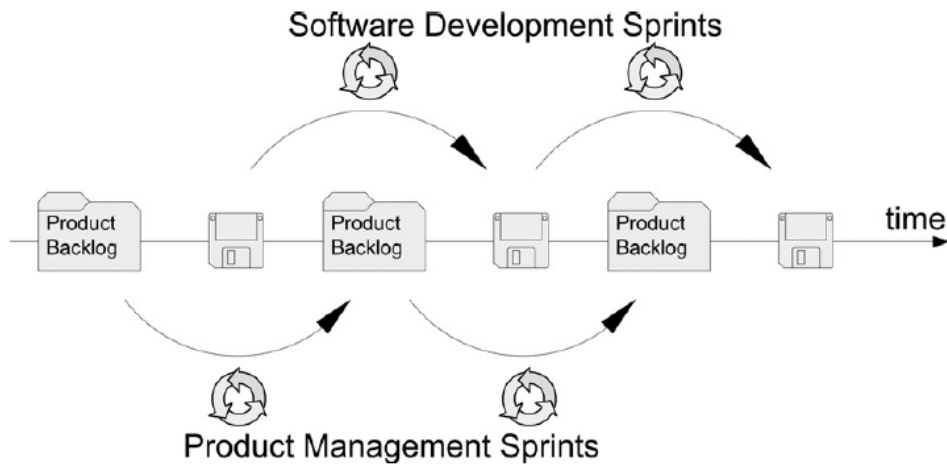


Figure 3-8: Alternating sprints in combining Scrum and SPM (Vlaanderen et al., 2011)

3.5. Customer and user involvement in RE

Involvement of customers and users in product development is not new. Already in 1998, Kaulio (1998) reviewed seven different methods for this involvement, including Quality Function Deployment (in which customer needs drive product design and production) and beta testing. However, Potts (1993) surveyed a number of software-development organizations and found that requirements were usually invented instead of elicited. A gap between developers and users was observed. Keil and Carmel (1995) focused on this gap and studied direct and indirect links between customers and developers. A link is a way in which customers and developers exchange information during the development process. The authors suggest using more direct links, but less indirect links (e.g. communication via intermediaries or customer surrogates).

Different forms of user involvement exist. Damodaran (1996) distinguishes informative, consultative and participative user involvement, which ranges from users providing information to users influencing decisions. Kabbedijk et al. (2009) study customer involvement in requirements management and identify the levels of *incident reports* (design for customers), *idea feedback* (design with customers) and *suggestions* (design by customers) are distinguished, based on Kaulio (1998).

3.5.1. Potential

The potential impact of user involvement on software quality is high. According to Kabbedijk et al. (2009), integrating product development requests that arise from customer involvement improves customer loyalty and might broaden the market. Kujala (2003) found that early user involvement is expected to lead to more accurate user requirements, avoidance of expensive and unnecessary features and improved acceptance of the system. In a later study, Kujala, Kauppinen, Lehtola and Kojo (2005) even shows that user involvement – while being rare – is correlated with better requirements quality. When requirements come from customers or users, the chance of project success is higher and the costs of the RE process are lower.

3.5.2. Obstacles

Kujala et al. (2005) also emphasize that, while customers are often seen as the most important stakeholders of a software system, end-users should be considered the most important. It might however be hard to interact with all your customers and end-users. Cao and Ramesh (2008) have studied companies that employ

agile RE and found that these companies had difficulties in gaining access to the customer and obtaining consensus among various customer groups. Holmström (2004) recognizes the problem of changing user needs when these users are distributed all over the world. As a potential solution, her research explores how virtual communities can support software maintenance. The empirical study shows that the researched game community is mainly useful for software fault repair and software adaptation; software addition is mainly inspired by in-house ideas and external requirements.

3.5.3. Innovative techniques

Because of the positive influence of user involvement on software quality and new ways to access users, innovative ideas are popping up in recent literature. Seyff, Graf and Maiden (2010) chose to research the help of mobile tools in eliciting requirements, in order to gather individual needs in a user-led manner. The authors designed the iRequire approach and application, with which users documented their needs in the form of contextual pictures and text- or voice-based need and task descriptions (Figure 3-9). A small-scale evaluation shows that users were able to document their needs and analysts were able to transcribe these needs. While the approach might become complex in large-scale RE and is focused on the early stage, it provides an interesting way to gather individual and contextual needs.

Storey, Deursen and Cheng (2010) identify community and user involvement as one of the opportunities of using social media in software development practices. The authors question how effective social media can be for the support of testing new features and requirements gathering. Social media would also be an appropriate channel to implement two lessons of Kabbedijk et al. (2009): 'give customers feedback afterwards about their suggested requirements' and to 'focus on interactivity'. An interesting industry example of user involvement in RE – that also integrates with social media – is Get Satisfaction² (Figure 3-10). The community platform gives users the opportunity to ask questions, submit ideas, give praise to the product and be a part of conversations about other ideas.

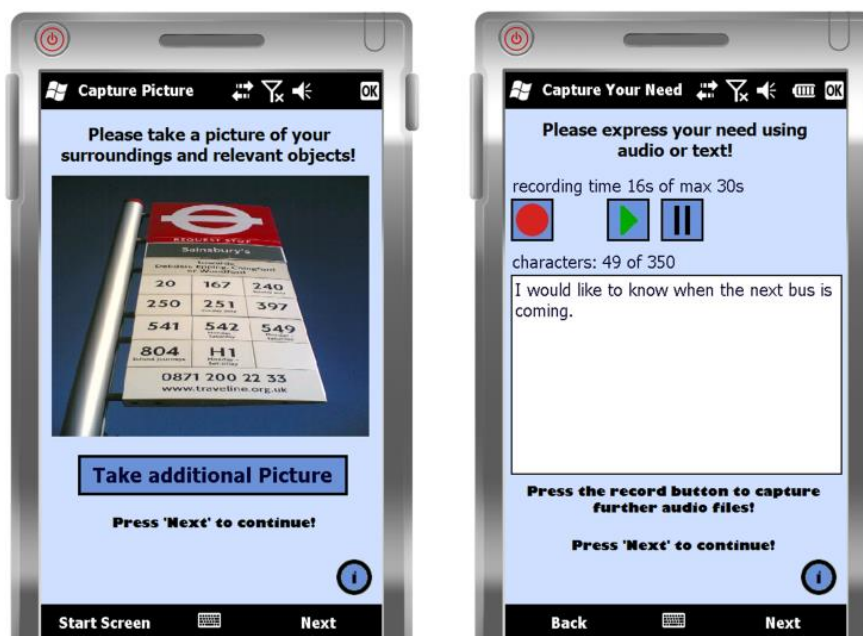


Figure 3-9: An example of the application of iRequire (Seyff et al., 2010)

² <http://www.getsatisfaction.com>, retrieved on September 12th, 2014.

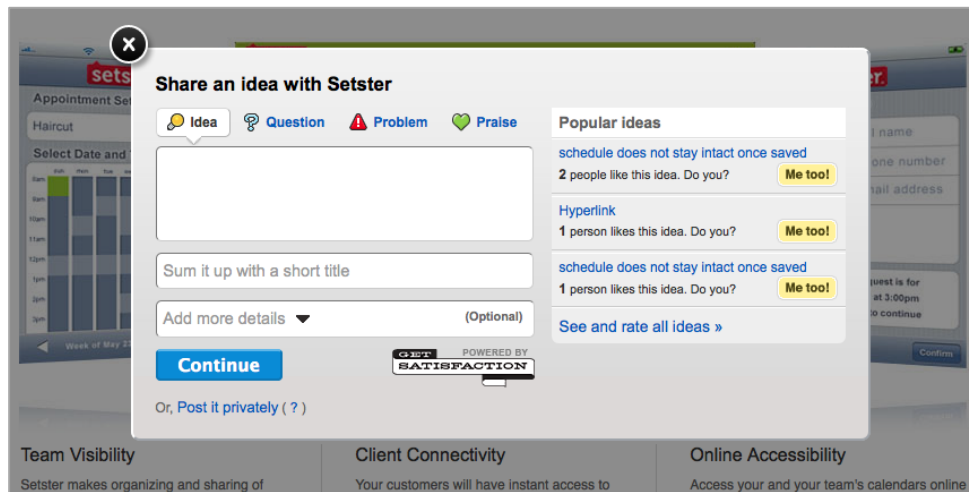


Figure 3-10: A screenshot of Get Satisfaction

3.6. Crowdsourcing

Crowdsourcing has the potential to be a suitable technique for large-scale user involvement. Howe (2006) defines crowdsourcing as “the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call”. According to Brabham (2008, p. 97), crowdsourcing is “a strategic model to attract an interested, motivated crowd of individuals capable of providing solutions superior in quality and quantity to those that even traditional forms of business can”. The author hereby refers to the high quality of wisdom of crowds, in which aggregated average answers could lead to excellent decision making (Surowiecki, 2005). Diversity and aggregation are two factors that distinguish crowdsourcing from outsourcing; crowdsourcing generates a large number of responses (e.g. solutions, ideas) which need to be aggregated by the crowdsourcing organization.

Brabham (2011) later restates his definition as “an online, distributed problem solving and production model that leverages the collective intelligence of online communities for specific management goals”. In explaining this definition, he highlights the differences with Howe's (2006) definition, which include the management goals and the online environment. One of the effects of this new definition is that Wikipedia³ is not considered a crowdsourcing platform due to the lack of ‘specific management goals’. Triggered by the disagreement between various scholars, Estellés-rolas and González-Ladrón-De-Guevara (2012) set the objective to form an exhaustive and global definition of crowdsourcing. A study of 209 documents containing 40 definitions of the concept was input for a very elaborate definition, that covers the majority of existing crowdsourcing processes:

“Crowdsourcing is a type of participative online activity in which an individual, an institution, a non-profit organization, or company proposes to a group of individuals of varying knowledge, heterogeneity, and number, via a flexible open call, the voluntary undertaking of a task. The undertaking of the task, of variable complexity and modularity, and in which the crowd should participate bringing their work, money, knowledge and/or experience, always entails mutual benefit. The user will receive the satisfaction of a given type of need, be it economic, social recognition, self-esteem, or the development of individual skills, while

³ <http://www.wikipedia.org>, retrieved on September 11th, 2014

the crowdsourcer will obtain and utilize to their advantage that what the user has brought to the venture, whose form will depend on the type of activity undertaken.” (p. 9)

More interesting than the exhaustive definition of crowdsourcing, is the actual relation with existing concepts. Schenk and Guittard (2009) describe this by comparing crowdsourcing to *Open Innovation*, *User Innovation* and *Outsourcing*. The differences that the authors highlight are described in Table 3-9. Additionally, the application of *Open Source* is shown to share elements with all compared concepts.

Table 3-9: The differences between existing concepts and crowdsourcing (Schenk & Guittard, 2009)

Open Innovation	User Innovation	Outsourcing
Restricted to innovation issues	User-driven instead of firm-driven	Crowdsourcing is an instantiation of outsourcing, not vice-versa
Describes interaction between firms, instead of a firm and the crowd	Restricted to innovation issues	
A two-way process involving selling and buying knowledge	Always involves end-users instead of any individual	



Figure 3-11: Facebook crowdsources translation.

A well-known and often studied application of crowdsourcing is Amazon Mechanical Turk (AMT)⁴, in which small tasks are performed by leveraging the human intelligence of a crowd of users. Over the years, a wide variety of other crowdsourcing examples have popped up. Threadless.com⁵ crowdsources t-shirt design, Facebook⁶ has crowdsourced the translation of its website (Figure 3-11) and Starbucks⁷ collects ideas for anything from products to music through community involvement. All these famous examples are Business-To-Consumer (B2C) applications of crowdsourcing. Both research and practice in Business-To-Business (B2B) crowdsourcing is limited. Kärkkäinen, Jussila and Multasuo (2012) studied crowdsourcing in B2B innovation processes with a systematic literature review of nine B2B cases. B2B crowdsourcing turned out to be somewhat different from its B2C variant, involving a different crowd, motivators and tasks.

Hosseini, Phalp, Taylor and Ali (2014a) have conducted a more in-depth crowdsourcing study and created a taxonomy that represents crowdsourcing in four pillars: the crowdsourcer, the crowd, the crowdsourced task

⁴ <https://www.mturk.com/mturk/welcome>, retrieved on September 12th, 2014

⁵ <https://www.threadless.com/infoabout/>, retrieved on September 11th, 2014

⁶ <http://techcrunch.com/2008/01/21/facebook-taps-users-to-create-translated-versions-of-site/>, retrieved on September 11th, 2014

⁷ <http://www.mystarbucksidea.force.com>, retrieved on September 11th, 2014

and the crowdsourcing platform. Each of the pillars has its own features which are connected by dependencies, allowing practitioners and researchers to configure crowdsourcing examples.

3.6.1. Crowd Creation

The definition of Howe (2006) implies that the crowd should be an “undefined (and generally large) network of people”. Besides the vagueness of this implication, other studies contradict the undefined aspect by outlining certain essential characteristics of a crowd. Surowiecki (2005) states that a crowd should be diverse, independent and decentralized. Hosseini et al. (2014a) base their features of the crowd on a literature review. In addition to diversity, the features include unknown-ness, largeness, undefined-ness and suitability. The practicality of creating such a crowd can be questioned, how can we be sure to have a competent and motivated individuals in a crowd that is not known or even not defined?

Geiger et al. (2011) takes a more practical point of view and identifies pre-selection of contributors as a potential characteristic of crowdsourcing processes. Pre-selection could be based on qualification or context-specific, but was often not used in identified processes. Brabham (2008b) studied the actual composition of the crowd at iStockphoto and found that the majority of the crowd consisted of ‘elite internet users’, being relatively wealthy and high-educated. This study emphasizes the difficulty of creating a diverse crowd.

3.6.2. Crowdsourcing motivation

As Brabham (2008) notes, the crowd is often paid for the solutions they provide, but this is a significantly lower amount of money than professionals would be paid, resulting in something that resembles a “slave economy”. This might indicate that motivation comes from somewhere else than the extrinsic monetary motivator.

A number of recent studies have focused on the motivation of the crowd. Brabham (2010) found four primary motivators at Threadless: the opportunity to make money, the opportunity to develop creative skills, the potential of freelance working and the love of community. Furthermore, the author identifies a form of addiction among crowd members. Findings at iStockphoto were similar, but included the creative outlet and fun (Brabham, 2008b). Kaufmann and Veit (2011) have researched the same topic, but surveyed a large number of AMT workers. The participants ranked payment, task autonomy and skill variety as the most important motivators.

Chandler and Kapelner (2013) are the first to conduct a field experiment in the journey to find crowdsourcing motivators. Image-labeling subjects on AMT were given three categories of meaningfulness. The ‘meaningful group’ was told that they were assisting medical researchers, the control group was given no context and the ‘shredded group’ was told that their work would be discarded. The three groups were paid in an identical way, but showed significant differences. In more meaningful tasks, workers were more likely to participate and produced a higher quantity of output. Low-meaning tasks decreased the quality of output. The study of Chandler and Kapelner hereby gives interesting insights in the importance of meaning in crowdsourcing tasks.

Money seems to be an important motivator to do crowdsourcing work, but should ideally be complemented by the ability to (autonomously) build skills and an engaging community. The addition of meaning to the crowdsourcing task can increase the quality and quantity of the output.

3.6.3. Crowdsourcing quality

Outsourcing a business function to an undefined group of people might of course raise quality issues. A number of studies evaluate the quality of the results of crowdsourced tasks, mainly using AMT. Snow et al. (2008) state that AMT-supplied data is often plentiful, but noisier than expert data. Their results of using AMT for natural language annotation however shows that a small number of non-experts can provide the quality

of an expert annotator. Kittur, Chi and Suh (2008) have used AMT for the collection of user input on the quality of articles. In a first experiment, the researchers gained little useful results, while in a second experiment, they included verifiable questions and gained higher quality responses. The study thereby shows the effect of task design on the quality of crowdsourcing results. Ambati, Vogel and Carbonell (2010) research translation with AMT and show that multiple translations are needed to suppress noise and multiple translators often disagree.

On an individual level, the quality of crowdsourcing results seems poor. However, the presence of a crowd allows for combining multiple results and thereby increasing quality. It also has to be noted that language tasks are of a different nature than tasks related to RE, where quality might be reviewed in very different ways. While translation tasks have 'right' or 'wrong' solutions, elicitation for requirements is mostly dependent on the creativity, relevance and articulation of needs of the user.

3.6.4. Crowdsourcing in Requirements Engineering

Little research can be found on the use of crowdsourcing in RE. Lim, Damian and Finkelstein (2011) have created StakeSource 2.0, a tool that uses "a crowdsourcing approach to identify and prioritize stakeholders and their requirements" (p. 1022). The tool asks stakeholders to suggest new requirements and rate existing requirements. Requirements are recommended using collaborative filtering. Finally, stakeholders in conflict are identified. The StakeRare method (Lim & Finkelstein, 2012) uses StakeSource 2.0 in a method that supports requirement elicitation and prioritization. The evaluation shows that StakeRare is able to elicit a complete set of requirements, which are prioritized accurately. Stakeholders were motivated to use the method and requirements conflicts were detected, potentially increasing stakeholder buy-in.

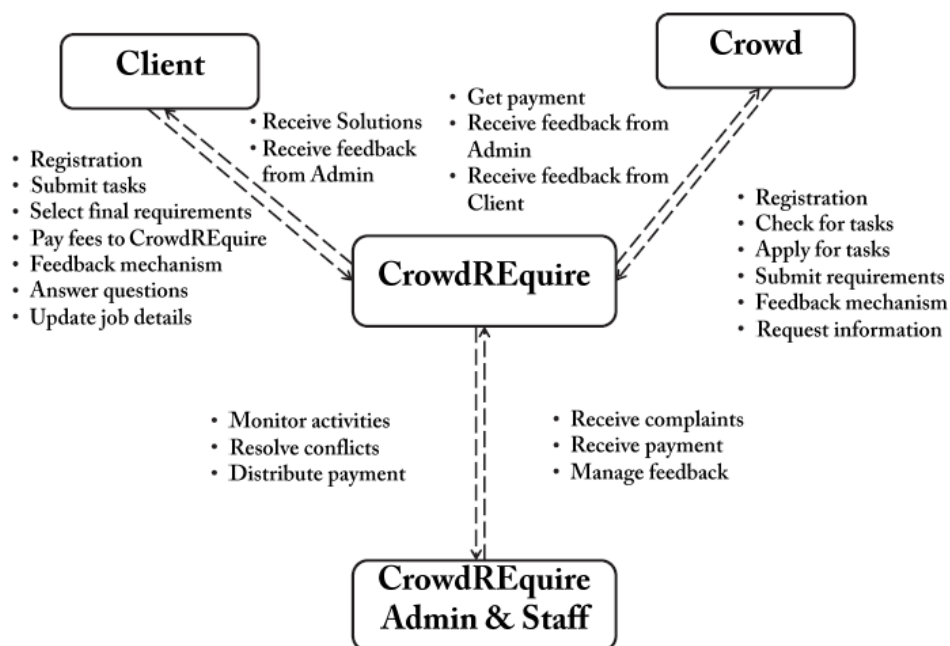


Figure 3-12: The use case diagram of CrowdREquire

Hosseini, Phalp, Taylor and Ali (2014b) are in the early stage of researching the use of crowdsourcing in RE and more specifically in requirements elicitation. Their study is a work in progress, but has already identified a list of quality attributes that accompany crowdsourcing features. Adepetu, Ahmed, Abd, Zaabi and Svetinovic (2012) describe the crowdsourcing platform CrowdREquire, which supports requirements

engineering. While the business model, context model, use case diagram (Figure 3-12) and business event list are outlined, the solution is lacking relations to the specific phases of RE. Important challenges as incentivizing the crowd and improving specification quality are hardly addressed.

3.7. Gamification

Deterding, Dixon, Khaled and Nacke (2011, p.10) define gamification as “the use of game design elements in non-game contexts”. These game design elements should be characteristic to games and have a significant role in gameplay. The variety of applications is wide, from an immersive running game where the player is virtually chased by zombies⁸ (Figure 3-14), to reputation and expert badges on the Q&A website Stack Overflow⁹ (Figure 3-13). According to Gartner (2011), “by 2015, more than 50 percent of organizations that manage innovation processes will gamify those processes”. The research firm states that gamification has the goals to achieve higher levels of engagement, change behaviors and stimulate innovation. Engagement is driven through four means:

- Accelerated feedback cycles;
- Clear goals and rules of play;
- A compelling narrative;
- Tasks that are challenging but achievable.

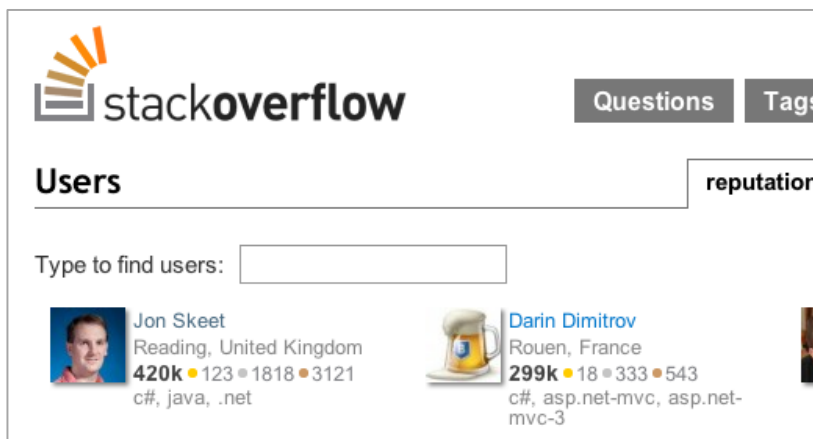


Figure 3-13: Gamification in Stack Overflow through reputation and badges

The term ‘gamification’ is criticized, for example by Robertson (2010), who argues that the application of the least essential things of games (points, badges) should not be called gamification, but rather ‘pointsification’. Nicholson (2012) explains that external rewards even decrease internal motivation. The author addresses the importance of situational relevance (the background of the user), situated motivational affordance (the organizational context), universal design for learning (what, how and why of gamification) and player-generated content (individual goals and methods). These factors should lead to meaningful gamification, resulting in deeper engagement between the users, activities and organizations.

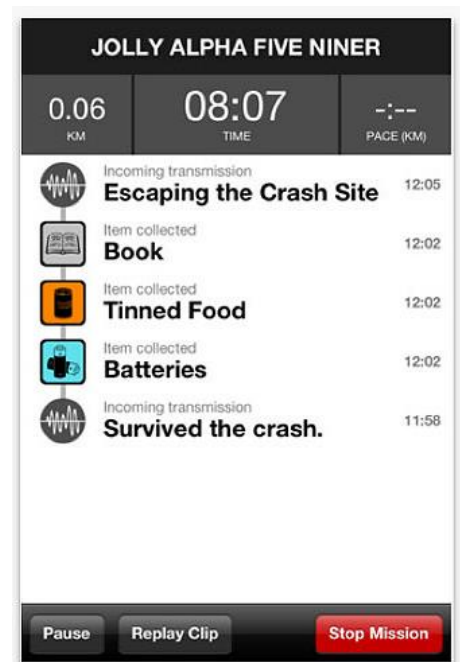


Figure 3-14: Gamification in the Zombies Run Game

⁸ <https://www.zombiesrungame.com/>, retrieved on September 11th, 2014

⁹ <http://www.stackoverflow.com>, retrieved on September 11th, 2014

Since many studies suggest gamification, but the effects are largely unknown, Hamari, Koivisto and Sarsa (2014) study the motivational affordances (game elements), psychological outcomes and behavioral outcomes of gamification studies. The most important elements that are derived from their study are points, leaderboards and achievements. Most studies show (partly) positive effects of gamification, though not always on the long term.

Hamari and Koivisto (2013) have studied the effects of social factors on the attitude towards and use of gamified services. The authors found that the exposure to a network positively influences the social influence, recognition and reciprocal benefit in the service. These three elements improve people's attitude towards the service and thereby increase the chance of continued use and recommendation. Figure 3-15 depicts these significant ($p < 0.001$) positive effects of social factors on attitude and the intention to use or recommend the service through Word Of Mouth (WOM). The model is in line with the Technology Acceptance Model (TAM), which explains and predicts user acceptance of information technology. The TAM identifies the perceived ease of use and perceived usefulness of a computer system as determinants for the attitude towards using and the behavioral intention to use (Davis, Bagozzi, & Warshaw, 1989).

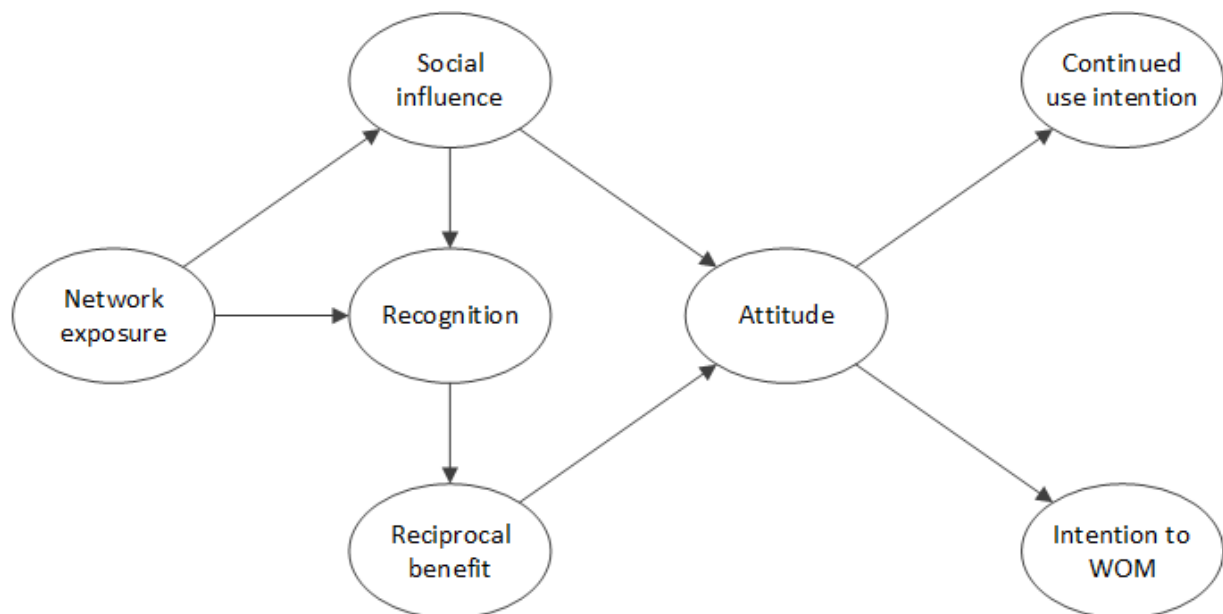


Figure 3-15: The effects of social factors on the attitude towards and use of gamified services. An arrow indicates $p < 0.001$ (Hamari & Koivisto, 2013).

3.7.1. Gamification and crowdsourcing

Research on motivation in crowdsourcing was described in Section 3.6.2 and emphasized the importance of a motivated crowd. The potential of gamification to motivate users makes it particularly interesting for crowdsourcing. Most of the papers reviewed by Hamari et al. (2014) describe gamification in crowdsourcing systems. A number of industry examples of gamification is strongly related to crowdsourcing. The innovation game *Idea Street*, for example, stimulates 120,000 people to generate ideas for the U.K.'s Department for Work and Pensions, using points and leader boards. Threadless lets users score t-shirts in order to vote them in or out of the shop. Additionally, users can win royalties or a gift card for the shop by submitting a design.

The ability to trigger motivation is not the only benefit of gamification. Eickhoff et al. (2012) leverage gamification to attract and retain participants for relevance assessments. Players of the game get points when previous players agree with their choices. In addition, bonus points are given after a set of judgements and the task becomes more challenging over time. This all leads to a certain ranking on the leaderboard. Compared to the traditional crowdsourcing paradigms, the gamified approach leads to quicker and higher quality responses with less ‘cheaters’ (users that provide useless responses).

Witt, Scheiner and Robra-Bissantz (2011) research the effect of game mechanics within online idea competitions. Participants of these competitions could get game points by contributing or commenting on ideas. Social points were rewarded when other participants vote on ideas. A leaderboard was used to keep track of the relative rankings of the participants. Results show that the participants were mainly driven by the usage of knowledge, curiosity and rewards. However, the points and rankings did not strongly affect the participants’ happiness.

3.7.2. Gamification in Requirements Engineering

Not much literature could be found regarding the application of gamification in RE. Fernandes et al. (2012) have made an attempt to apply gamification to requirements elicitation by developing the game-based collaborative tool ‘iThink’. The tool aids in collecting new requirements and gaining feedback on existing requirements (Figure 3-16). The elicitation is performed using the Six Thinking Hats method, to increase participation, engagement and collaboration. Two case studies indicate enhanced user involvement in requirements elicitation. Ribeiro, Farinha, Pereira and Mira da Silva (2014) further evaluate the effectiveness and acceptance of iThink. While the used surveys were not very rigorous, the results show that both participants and project managers are satisfied with the method.

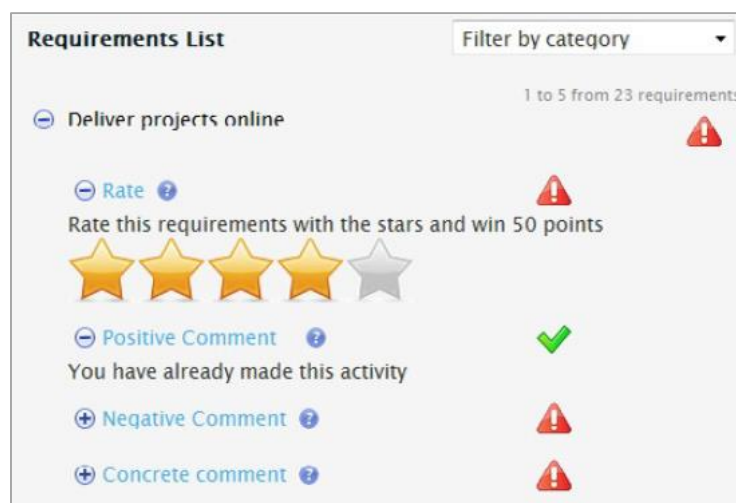


Figure 3-16: A screenshot of iThink

Gamification potentially has a very positive impact on crowdsourcing and requirements engineering, since it could lead to engagement, user satisfaction, innovation, less crowdsourcing ‘cheaters’ and quick and high quality response. However, the game elements have to be selected and designed with great care in order to generate this impact.

3.8. Conceptual model

The concepts of the class diagram in Figure 3-17 are resulting from the literature review that is described in the current chapter. The notion of a separate *User* and *Customer* – both being *Stakeholders* – is different

from traditional frameworks (Bekkers, van de Weerd, et al., 2010) and emphasizes the relevance of this study. Kujala et al. (2005) stated that end-users are more important than customers in giving feedback, since they are the ones that experience the quality of the software. In our approach, it is essential to represent all perspectives in the feedback, which rationalizes the separation of stakeholders.

The blue classes *Crowd*, *Platform* and *Task* come from the work of Hosseini et al. (2014a) as three of the four pillars of crowdsourcing. The fourth pillar is the crowdsourcer, the *Software Producer*. The activities of the *Requirements Engineering process* come from our AS-IS model in Section 3.3. The green class, *Game element*, shows the influence of gamification.

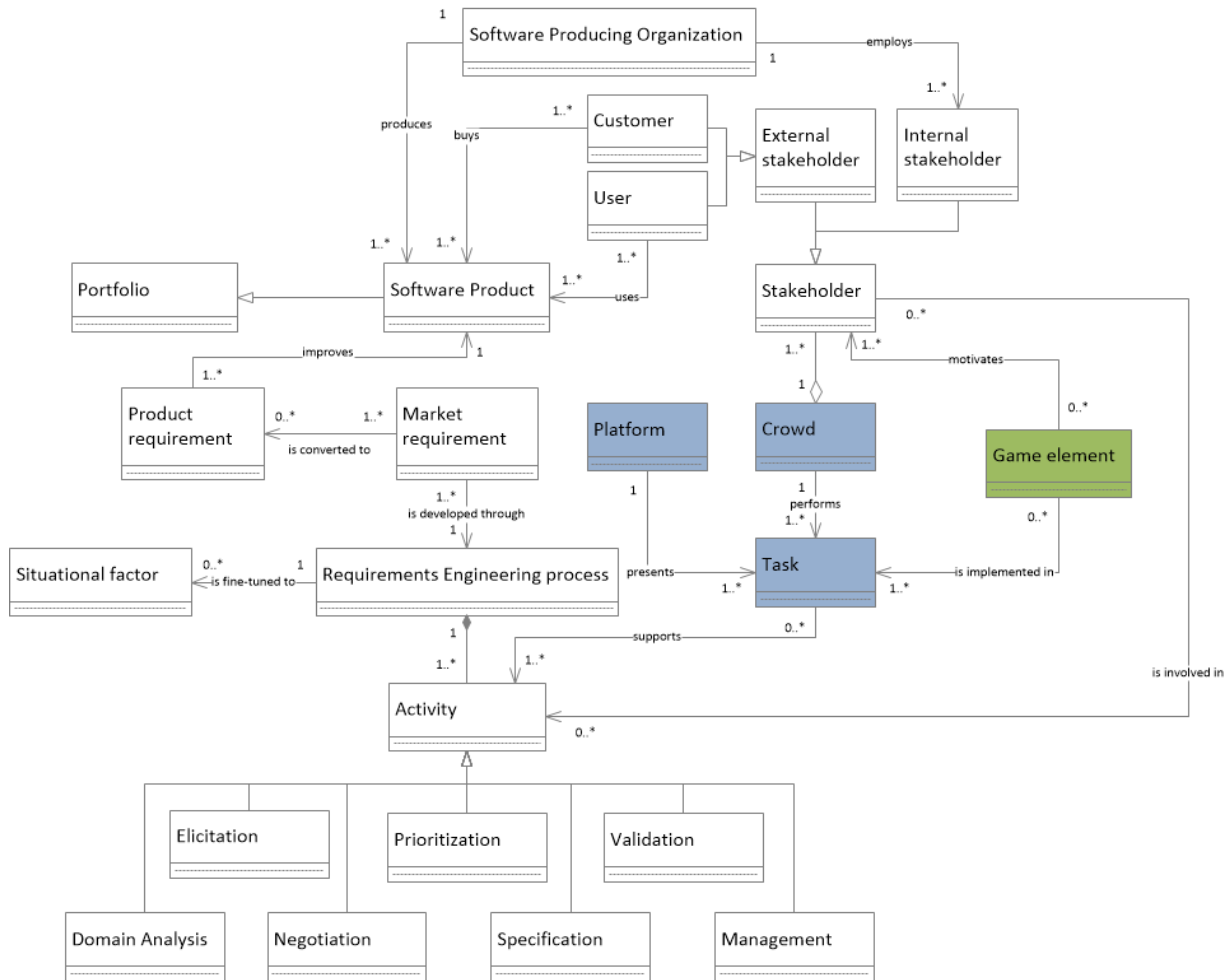


Figure 3-17: Conceptual model of CCRE-related concepts

3.9. Method Engineering

Method Engineering (ME) is defined as “the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems” (Brinkkemper, 1996) and used in this thesis for the development of the CCRE method. An important aspect of the discipline is the distinction between methods, techniques, tools and methodologies. The definitions of these terms are given in Table 3-10.

Table 3-10: the definitions of ME-related terms according to Brinkkemper (1996)

Term	Definition
Method	“An approach to perform a systems development project, based on a specific way of thinking, consisting of directions and rules, structured in a systematic way in development activities with corresponding development products”
Technique	“A procedure, possibly with a prescribed notation, to perform a development activity”
Tool	“A possibly automated means to support a part of a development process”
Methodology of information systems development	“The systematic description, explanation and evaluation of all aspects of methodical information systems development”

A method consists of method fragments, which include product fragments and process fragments. The common technique for modelling a method is a process-deliverable diagram (PDD) (van de Weerd & Brinkkemper, 2008). An example of such a diagram is given in Figure 3-18. The activities on the left are modelled based on a UML activity diagram, the concepts on the right based on a UML class diagram.

Situational Method Engineering is concerned with the construction of project specific methods (Harmsen et al., 1994). Essential is the method base, in which standardized method fragments are stored and from which these can be retrieved. Through a process of project characterization, fragment selection, fragment assembly, performance measurement and administration, the method base is constantly updated and queried. The coloring that was introduced in the previous section (blue = crowdsourcing, green = gamification, orange = both) will be used throughout the diagrams – including PDDs – in this thesis. It supports situational method engineering by showing the origin of fragments.

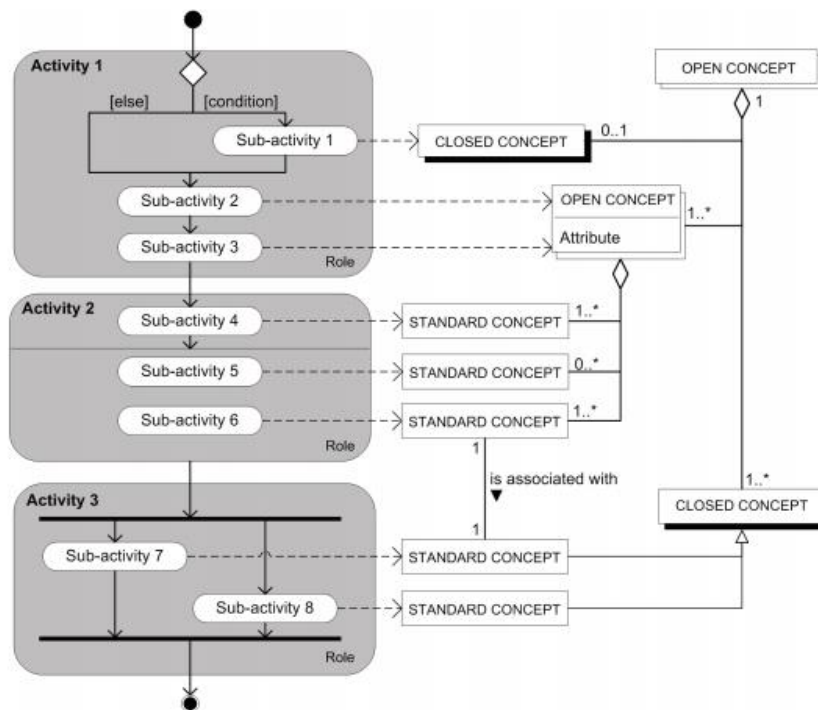


Figure 3-18: An example of a PDD (van de Weerd & Brinkkemper, 2008)

4. Expert Interviews

4.1. Introduction

In order to get an overview of necessary method fragments for the CCRE method and to identify constraints and opportunities for configurations of those fragments, RE experts from research and industry were interviewed. A total of ten experts, which are listed in Table 4-1, participated in the semi-structured interview.

The questions are grouped, based on three topics: describing the current RE process, identifying room for improvement and successfully implementing crowdsourcing and gamification. Appendix A contains the interview questions and the introduction that was shared with the interviewees. It has to be noted that the questions were slightly tailored to the expert's interest in each interview, in order to get useful answers. This tailoring can be exemplified with the first question: product software professionals were asked which RE activities they conducted, advisors were asked which activities they observed in advising the industry and researchers were asked which activities they observed in researching the industry.

The questions were developed with the aim of validating the phases that were found in the literature review, finding strengths and weaknesses of the real-world process and assessing the potential value and applicability of crowdsourcing and gamification. The main artefact of this study, the CCRE method, is described in Chapter 5.

Table 4-1: Interviewed experts and their backgrounds

#	Expertise	Type of organization	Organization location	Role
1	Software Quality, Application Development	Advisory	The Netherlands	Partner
2	Innovation, Business Process Improvement	Advisory	The Netherlands	Partner
3	Functional Design, Software Architecture	Advisory	The Netherlands	Senior manager
4	Functional Design, Interaction Design	Advisory	The Netherlands	Advisor
5	Project/Product Management, Software Development	Product software (S)	The Netherlands	Technical director
6	Product Management, Software Operations	Product software (L)	The Netherlands	Product manager
7	Software Engineering, Crowdsourcing, Gamification	Research institute	United Kingdom	Senior lecturer
8	Systems Engineering and Human Factors	Research institute	United Kingdom	Lecturer
9	Collaborative Creativity in Requirements Engineering	Research institute	Belgium	Researcher
10	Software Product Management, Process Quality Improvement	Research institute	The Netherlands	PhD Candidate

4.2. Results

Each interview was recorded and summarized, after which the advices of the experts were extracted. Some advices were literally stated, while others were interpreted and rephrased. “Do not request users to perform difficult work” [UR1] was literally stated by interviewee 10. An example of a rephrased advice is “Ask clarifying questions” [NE1], which came from the notion that “You always need extra questions and interpretation” (interviewee 3). This extraction led to a total of 112 advices. Interviewees put focus on varying areas, which was reflected in their advices. For example, interviewee 9 focused very much on offline representation of users, while interviewee 8 was mainly interested in crowdsourcing and gamification aspects. The interview summaries can be found in Appendix D.

The advices were mapped on the concepts of the conceptual diagram that was depicted in Section 3.8. After an initial mapping, in which the concepts served as categories, several categories contained many advices (e.g. *User Involvement* and *Gamification Elements*) and a number of advices were overlapping (e.g. “involve users, to understand what the real problems are” and “involve real users, not representatives”). These results led to a reiteration of assigning categories and assembling advices, while keeping track of the number of times an advice was given. An example of an advice that resulted from two aggregated advices is “Identify or create the right interactive feedback channel” [CA4], which was the convergence of “Identify the right input channel” and “Create a dedicated platform to annotate software with your feedback”. The 68 final advices are listed in the following section, complemented by a concise context of the advice. The interviewee that gave the advice is referred to in this context, with or without quotes. A number behind the advice shows that it was given by multiple experts.

4.2.1. Vision and Openness

VO1. Do not discuss strategic decisions – These are decisions that are explicitly made by a company and cannot be overseen by the customers or users. In addition, these decisions are part of the vision of a company (#6).

VO2. Most customers don’t know what they need (2) – “They say they want A, while they mean to say B, but they actually need C.” (#6) Phrased another way, “customers often come up with something, but don’t actually use it.” (#2)

VO3. Do not say that you know what is good for your users – “That is very IT 1.0.” As a software developer, you might not have a vision about a specific market. “It is not a weakness to let users guide you.” (both #1)

VO4. Make software that the user doesn’t know (s)he wants – “That’s the art of requirements engineering. The most famous example is the iPad, people didn’t know they want one, but it became popular.” (#6)

4.2.2. Situational Factors

SF1. Base your RE techniques on the context of your product, organization and specific customer (2) – The Requirements Engineering process is dependent on many factors. The size of the organization is very influential, as well as the type of product and the diversity and the number of stakeholders. Involving users is also very dependent on the type of product, “an ERP package is more standardized than logistic software for truckers” (#10).

SF2. Situational factors include strategic goals, the type of software, frequency of use, current usability and the number of users – Considering strategic goals as a factor is in line with [VO1]. The type of software might determine the degree of standardization, but it also determines – combined with frequency of use and current usability – the willingness of users to be involved in improving the software. “For a project in which

we successfully involved users, people worked with an insufficient system every day. That's why our project worked. They wanted to be involved." (#3) The amount of users determines which involvement technique is successful, crowdsourcing requires a relatively large-scale system.

4.2.3. Contextual Analysis

CA1. Determine your degree of vision & openness – Due to the large support for this factor, this advice was further specified in advice [VO1-VO4]. According to one expert, "the degree of vision determines the openness for customer suggestions" (#6). However, the interplay between vision and openness can be more dynamic: "Users are important in the evaluation of your vision" (#10).

CA2. Determine an intended outcome – The intended outcome is highly influential on the stakeholders you want to involve. "Our client demanded that the usability of an application would be improved. We told them that we really needed to involve the users. And not only the wiz kids, but also the older users that have difficulties in using IT." (#3)

CA3. Structure the existing feedback channels – The overload with ideas, questions, suggestions, requirements etc. makes it complex to manage and structure input, but "if you don't properly structure this, you misjudge it" (#10). You need to control all the channels that are used.

CA4. Identify or create the right interactive feedback channel (2) – "It helps a lot to identify the right channel." (#10) When you have one channel dedicated to feedback and communicate this to your customers, they will start to use it. One expert steered customers to the right channel by ignoring other channels: "There are only a few customers that bypass the channels, I keep that door locked" (#6).

4.2.4. Stakeholder Involvement

SI1. Provide an incentive to be part of the community (2) – "A community needs to emerge from free will. (...) Why would they put effort in your product?" (#10) Within your company, you can give people time and money to further develop ideas, but externally it is more difficult. "When people are emotionally involved in a product, you can motivate them without money." (#2)

SI2. Deliver options ready-made to participants (2) – "Refining ideas is easier than suggesting ideas. (...) When you deliver things ready-made, in small packages, it is easier for stakeholders to interact with it." (#10) "Our first workshop starts with best guesses (...). This is easier than starting with an empty sheet." (#3)

SI3. Make RE a participative co-design activity – It often happens that businesses use a fake participative approach, by involving people but finally making decisions themselves. This has to change towards real participative co-design. "Requirements Engineers are not translators anymore, but can be seen as the facilitators of the co-design activity. (...) The problem has to be defined together and finally a solution has to be created together." (#9)

SI4. Involve stakeholders structurally instead of sporadically – A large scale is not the only problem of user involvement, a lack of hierarchy is also problematic. "When we would send someone from our company, they will give feedback, because they get paid or get rated. Users miss the hierarchy to structurally make time at the right moment." (#1) This is not only due to motivation, it is also difficult to organize. However, if you really want effective involvement, a structured approach is necessary.

4.2.5. Crowdsourcing

CS1. Crowdsourcing has the goals to let stakeholders 1) learn from other stakeholder groups and 2) reach consensus – "Crowdsourcing is used to enhance consensus and decisions making." (#8) This is in line with the Wisdom of the Crowds concept (Surowiecki, 2005). By showing the progress of the RE process and of specific

requirements, the stakeholders' learning about the context of the system and about the context of other stakeholder groups can be increased.

CS2. Continuously and dynamically involve users, without trying to prevent bias – “Crowdsourcing is not only about how many you involve, but also about how long you involve them.” (#7) The crowdsourcer should be prepared to see members of the crowd coming and leaving. It should also be acknowledged that bias is the nature of life, there are people who are influential and others who aren't. Based on the vision, openness and intended outcome, bias needs to be prevented, in order to reach certain goals. In such a case, a blended approach might be better than a crowdsourcing approach.

CS3. Consider minorities' opinions – “Number is very deceiving in crowdsourcing, you should not listen to the majority but rather look at groups. (...) Don't treat [the crowd] as uniform, then you might as well take one of them.” (#7) The reason we a minority might object something could be very important.

CS4. Spread ideas that are created by others – “There are users who come up with ideas and users who refine ideas of others.” (#10) It is essential to connect those groups and let people build on each other's ideas.

4.2.6. User Responsibilities

UR1. Do not request users to perform difficult work – This will discourage them from being involved (#10).

UR2. Do not have a large group suggest, discuss and rate requirements, leading to a big mess – “If it is an open source thing, which the whole world can access, it will cause chaos.” (#4) There might be people who make it less pleasant for other people.

UR3. Allow users to rate each other's stories – In this way, you can determine the quality of the requirements (#4).

4.2.7. Crowd Formation

CF1. Select communication instruments and invent an initial marketing campaign – This campaign is essential to get people involved. “[With internal idea generation] we sometimes show a video of the big boss to emphasize the importance of participation” (#2).

CF2. Communicate openness and what you do with the feedback (2) – “You have to be open for suggestions. If that's clear, people will find you.” (#10) In that way, people know what they can expect. “If people have a feeling that you don't use the feedback, it will die, even if you have given rewards. Like when you write an article in the newspaper and the newspaper is gone afterwards. (...) Transparency is important.” (#7)

CF3. Include all relevant stakeholders, including users (3) and experts (2) – “It is always best to involve users, to understand what the real problem is.” (#4) Often, a set of user representatives is chosen. However, for most software, “the set of users is very large and always changing, you don't know who is going to use it” (#7). Domain experts can be useful to develop your roadmap and model elicited requirements (#5) or to have constant contact during agile development (#1).

CF4. Check for variety (2), independence and decentralization – “When a crowd is heterogeneous, you won't have bias.” (#1) The four pillars of crowdsourcing (Hosseini et al., 2014a) need to be taken into account, although you can interfere when you have a strategic objective (#7).

CF5. Use sub-communities to focus on requirements (2) – You can “isolate” a group of users into sessions, which would lead to a sub-community (#10). These representative smaller groups could be selected by the product owner and customer (#4).

4.2.8. Crowd Control

CC1. Introduce guidelines on the platform – It might be a good idea to introduce guidelines, such as templates or rules of the community, this would prevent misuse and chaos (#4).

CC2. Moderate discussions, don't let the crowd do that (2) – “You obviously have to moderate it, especially when it is on the internet.” (#1) If you leave moderation to the crowd, it might turn into an unexpected direction (#3).

CC3. Give each customer group one vote, to make the voting fair – Otherwise, one customer group might have a larger representation than another group and only vote for their own requirements (#6).

CC4. Do not steer involved users – Moderators should not steer users. “People should be free” (#10), although that has the risk of leading to a lot of data.

CC5. Put unwanted comments in draft mode with an explanation – In this way, chaos can be prevented and the user is provided with a rationale why the comment does not fit the purpose of the platform (#2).

4.2.9. Feedback to the Crowd

FC1. Interact about crowd feedback (2) – “It is essential to interact with the users about their feedback, else they will quickly lose interest in contributing.” (#4) One expert stated this even more clearly: “You need to do something with your platform, else it is worse than having no platform at all” (#10).

FC2. Show timely results (2) – “Users should be motivated by knowing that their feedback is used.” (#3) It might be good to give a group of users access to the acceptance environment, “users are generally motivated when they see timely results of their suggestions” (#4).

FC3. Visualize Involvement – Awareness is essential and visualization can help in creating this. “People would like to see trends for their community, colleagues or peers.” (#7)

4.2.10. RE Process

RP1. RE should be a mix of non-isolated, intertwined activities (2) – “RE does not really exist of phases” (#9) and the different steps should not be isolated (#10).

RP2. The process should be agile and outcome-oriented, without too much simplification (2) – By making the RE process iterative, you create the ability to manage change (#4). In addition to this flexibility, the approach is outcome-oriented and pragmatic, which is sometimes missing in research (#7). The risk of being agile is simplifying too much, so awareness is needed to prevent that (#1).

RP3. Have short development and feedback cycles – This allows for continuous interaction and improvement (#1).

RP4. Have a sequence of ideation and idea development competitions – This supports different phases in developing requirements (#2).

RP5. Think many steps further than customers do – Product Management creates innovation by thinking ahead, overseeing implications and assessing feasibility (#6).

RP6. Give all stakeholders the same workflow – Everybody is able to share needs equally and all needs arrive at the same place (#6).

RP7. Save and organize the needs and requirements very well – The assurance of development requests creates a clear overview. “Although there is sometimes chaos in my head, the wishes are organized in the digital system”. (#6)

RP8. Possibly use part of your Sprint for your users' ideas – Hereby you can maintain your own vision, but effectively involve users (#4).

4.2.10.1. Elicitation

EL1. Do not suggest your own ideas – “Maybe you should do this as a pseudonym.” (#10) When a company asks a lot of questions about what they do, it doesn't show confidence.

EL2. Use data mining on existing feedback channels – There is very little research on this topic, but it has a lot of potential. Currently, social networks are only used in a backward way, but it could also be used in a forward way: “this is why we engineered it” (#7).

EL3. Ask a specific question – This is a good way to prevent chaos, to be transparent about your openness and to make the feedback process effective (#2).

EL4. Do not only use prototyping, since it limits the users' scope – “When you elicit feedback on a prototype, you only get feedback on the prototype. Your own design then limits what users can tell you.” (#5) To encourage users to think outside the box, a more open question is needed.

EL5. Determine who you want to hear – “All stakeholders have different needs, so we first determine who we want to hear”. (#2) End-users, customers, employees and product owners all have different perspectives. The target group is therefore very dependent on the intended outcome. This advice is in line with [CA2].

EL6. Prevent users from hijacking ideas – “[Users] often take someone else's idea and turn it around, to have their own idea.” (#2) This is obviously not constructive and should be prevented in the CCRE method.

4.2.10.2. Negotiation

NE1. Ask clarifying questions – “You will always need extra questions and interpretation” to further understand user needs (#3).

NE2. Allow peer reviewing (2) – To make analysis less intensive for the crowdsourcer, you could have a peer review by the crowd (#7). This should however be well-designed to keep it organized. This is very much in line with advice UR3, but takes the peer-reviews beyond ratings.

NE3. Use ranking and rationale to structure discussions – Crowdsourcing might of course lead to a chaos. To keep the discussions structured, a ranking or rating and a rationale behind that ranking should be used (#8).

4.2.10.3. Prioritization

PR1. Involve the crowd in prioritization – “Opening it up to the crowd opens possibilities to see other stakeholders and other reasons.” Involving the crowd in prioritization creates transparency and fairness (#7).

PR2. Use a lean but organized prioritization technique – “The leaner the process, the better.” A very lean process can however cause that you lose the overview. Good organization should help then. (#10)

PR3. Order requirements, to prevent everything from being a 'must have' – “In the past, MoSCoW was used for prioritization, but this led to every requirement being a 'must'.” (#3) A better way of prioritization is an actual order of relative importance.

PR4. Give weight to stakeholders (2) – “One customer might have a very different opinion, but might be very important to the business.” In another case, “a customer that always makes the top 3 of requirements, is a good initiator” (both #5). These factors can determine the weight of stakeholders. Rewards that you receive can also give you weight. “The more points you get, the more trustworthy you are”. (#7)

PR5. Have an insight in the number of customers that have a specific need – “We strive to have one description per wish and we also have a counter for these wishes.” (#6) This insight makes it easier to determine the priority of the need.

PR6. Don’t let users determine decisive priorities – The crowdsourcer has an overview of all users and can therefore make better decisions (#2). Users can be involved in prioritization, but should not be determining.

4.2.10.4. Validation

VA1. Evaluate your vision, detailed requirements and early product versions (2) with users – As already identified in CA1, “users are important in the evaluation of your vision” (#10). Detailed requirements should also be evaluated by users, in a validation (#4). This could be done by a sub-community. Finally, users can evaluate early versions of the software. “This information can be used to determine the next step.” (#4) Another expert invites customers to a ‘Quality Center’ after testing, to see “whether the product complies with their expectations” (#6).

VA2. Prepare users for unfinished features – While it is good to give users access to an early version of your software, these users should be aware that the software is unfinished. Otherwise, their expectations may be very different from the actual release (#4).

VA3. Keep using rigorous validation for quality dimensions – “Quality has many dimensions, which are not all crowdsourcable. Like security, you still need design time with a very rigorous validation and verification and hard milestones. Otherwise, the whole system will fail.” (#7)

VA4. Demo for a big audience – There are still difficulties in agile development. The product owner has personal preferences, with his own situation as a reference framework. You cannot really solve that, it is what it is. You should have your demo for a bigger audience: the development team, the product owner and his manager and maybe a bigger audience of users. For a company with a million customers, it is of course very hard to involve all users. (#1)

4.2.11. Gamification Elements

GE1. Gamification has the goal to motivate users and enhance problem understanding – The motivational character of gamification is very much in line with the goals that Gartner (Gartner, 2011) has identified. In addition, gamification might help stakeholders to better understand a certain scenario, by showing the different steps and characteristics of a scenario (#8).

GE2. Don’t reward a high quantity of input – The danger of providing incentives is that you get a lot of rubbish. The incentive should therefore be put on the quality of the input. (#3)

GE3. Reward help – “Ubuntu uses statuses for people that help other people, that works. The status that you get by helping people can be used in other activities. Helping should be rewarded.” (#10) Rewarding help can also be beneficial when stakeholders find it hard to come up with ideas, but can contribute to the ideas of others.

GE4. Use exploration, group forming, roles and rewards as gamification techniques – “Users would like to see how their feedback is used, which is exploration, a form of gamification. You can also allow them to form groups. Even if you are a minority with special needs, you can have a group.” (#7) On a platform, you might have different roles: suggesters, criticizers, refiners. “This is something you can gamify.” (#10) A more basic form of gamification is giving points or rewards for feedback (#7).

GE5. Weigh points over time – “We don’t use badges since that makes it harder for new entrants to keep up with people that already have badges.” (#2) It is necessary to make points valuable for new entrants too. This can be achieved by weighing them over time.

GE6. Give users a (gamified) real-world scenario, consisting of different steps – Turning a scenario into a serious game might increase the system thinking of stakeholders. Different steps can be created and based on the decisions in the game, the design solution and exploitation can be optimized. (#8)

GE7. Give the crowd a good feeling (at least) and a sustainable benefit – If there is no monetary reward, participants should at least get a good feeling. “The crowd determines whether they get a good feeling.” Besides the good feeling, the benefit of being involved should be sustainable. “Gamification works in a group in which I want to show off, but after a while I might lose interest.” An example in which this works is open source, where a variety of kinds of help is relevant and appreciated. (#1)

4.3. Summary

Ten interviews with experts in the field of Requirements Engineering, Software Product Management and crowdsourcing have provided us with interesting insights. Some interviewees showed skepticism towards the application of crowdsourcing and gamification in RE, while others acknowledged an interesting potential. Clearly, large differences were found between experts from the traditional field of software engineering and experts from the field of crowdsourcing and gamification. This uncertainty of the potential of these innovative mechanisms confirms the infancy of the field, which was already identified during the literature review. Other conflicting opinions were found in the degree of vision and openness that a SPO should have. This vision could vary from a completely defined roadmap to crowd-managed software. We hypothesize that the optimal solution lies somewhere in between both extremes.

Comparing the interview findings with the literature review, we found several similarities and differences. Infancy of the field was identified in both research phases, but the potential of user involvement on the quality of requirements (Kujala et al., 2005) was hardly mentioned in the interviews. While research acknowledges Wisdom of the Crowds as a high-value mechanism in decision making (Surowiecki, 2005), most experts emphasized the non-decisive role that users should have. The presence of situational factors was endorsed by the experts and several examples of relevant factors have been given.

A difference that could be expected was the isolation of separate phases and activities in RE. While research has put significant effort in defining phases, activities and steps (Section **Error! Reference source not found.**), the industry has an increasing focus on agile, outcome-oriented development in which all phases and activities are intertwined.

The next challenge is to utilize these advices and integrate them into the Crowd-Centric Requirements Engineering method. The results of this integration are shown and rationalized in the next chapter.

5. The CCRE Method

5.1. Development

Both the findings from the literature review and the advices derived from the expert interviews were used as *method suggestions* for the CCRE Method. A method suggestion could either identify the need for a specific method fragment or the organization of the method fragments, which we both call *method results*. Table 5-1 gives examples of various method suggestions and their effect on the CCRE method. As the examples show, one suggestion can lead to multiple results, one suggestion can lead to one result or multiple suggestions can lead to one result.

Table 5-1: Examples of method suggestions and results

Suggestion type	Method suggestion	Result type	Method result
Expert advice	Deliver options ready-made to participants [SI2]	2 activities	“Develop multiple design options” “Present design options”
Expert advice	Continuously involve users [CS2]	Activity flow	Crowd involvement as a continuous phase
Literature finding	Crowdsourcing can deliver accurate prioritization (Lim & Finkelstein, 2012)		
Expert advice	Involve the crowd in prioritization [PR1]	Activity	“Vote for needs”

5.2. Method structure

From the advices, some high level method requirements regarding the process were derived. The central phase would be the *crowdsourcing* phase in which needs are elicited and refined. Before that, the scope, vision and intended outcome have to be determined (*feasibility*). Additionally, stakeholders and a feedback channel have to be selected (*context*) and the stakeholders have to be invited, informed and engaged (*preparation*). After the crowdsourcing phase, high-potential requirements have to be extracted from the resulting needs (*identification*). *Focus groups* should further discuss those requirements, which of course have to be *prepared*. When the requirements are fully specified, they can be implemented as functionalities (*development*). This simplified version of the CCRE method is depicted in Figure 5-1.

In Section 3.3, a combined model of RE and SPM processes was shown. The CCRE method aims to improve these processes, which makes a mapping of the two models very relevant. Table 5-2 presents this mapping.

Two observations are important in reading this mapping. First, *Requirements management* overlaps with all phases of the CCRE method. This choice has been made since Requirements management consists of the planning, control and organization of requirements, which has to be done during each of the phases. Second, *Crowdsourcing preparation* has no other overlapping activities than requirements management. This does not make the activity unimportant. Crowdsourcing preparation is essential in assuring effective *Crowd involvement*.

The resulting method can be found in Appendix E. It contains eight phases, 42 activities and 29 concepts. The coloring of fragments indicates whether they incorporate crowdsourcing and/or gamification. All these fragments will be described in the next sections. The code between brackets behind the fragment refers to

the advice that inspired the fragment. For each activity, a small example of this activity is given in *italic*. The examples are based on an imaginary application of the method to Microsoft Word.

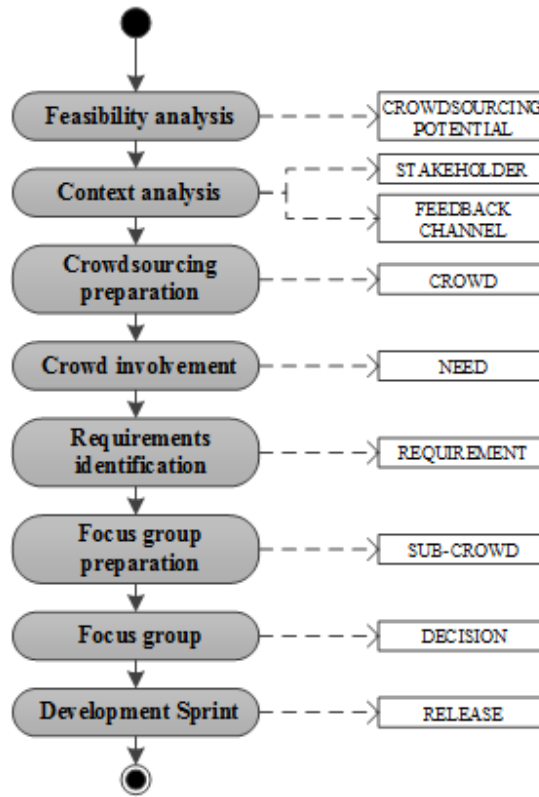


Figure 5-1: The simplified version of the CCRE method

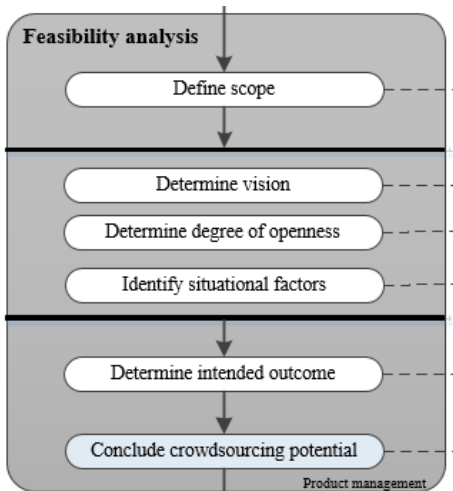
Table 5-2: Mapping of the CCRE phases on the processes of RE/SPM

	Requirements management	Domain analysis	Needs gathering	Needs analysis & negotiation	Requirements identification	Requirements specification	Requirements valid. & verif.	Requirements prioritization
1. Feasibility analysis	X	X						
2. Context analysis	X	X						
3. Crowdsourcing preparation	X							
4. Crowd involvement	X		X	X				X
5. Requirements evaluation	X				X	X		X
6. Focus group preparation	X					X		
7. Focus group	X		X	X		X	X	X
8. Development Sprint	X						X	

5.3. Phases and activities

5.3.1. Feasibility analysis

This phase has the goal to determine the applicability of CCRE for the specific situation. It is therefore positioned as the very first step in executing the method, to prevent the waste of resources in pursuing something irrelevant.



5.3.1.1. Define scope [V01]

The scope of the method is defined in this activity. It is important to develop boundaries of the application of CCRE. For instance, in one situation the method might be useful for the long-term product roadmap (the scope is defined with respect to the time horizon), while in another situation CCRE might only be applied to the product’s user interface (the scope is bounded by a set of features to be considered). A well-defined scope benefits both the internal stakeholders, by giving a certain explicit focus, and the external stakeholders, by being transparent about the area in which feedback is expected.

A non-extensive list of possible scopes includes:

- General improvements for the upcoming version (early-stage RE);
- Feedback on a beta product;
- Analysis of one specific (set of) feature(s);
- Definition of the long-term roadmap.

Example: our method will be scoped to Microsoft Word 2013. Since it is a very extensive product, we will focus on improvements to a specific set of features: the design and layout functionality of the product. This includes factors like themes, fonts and page margins (Figure 5-2).

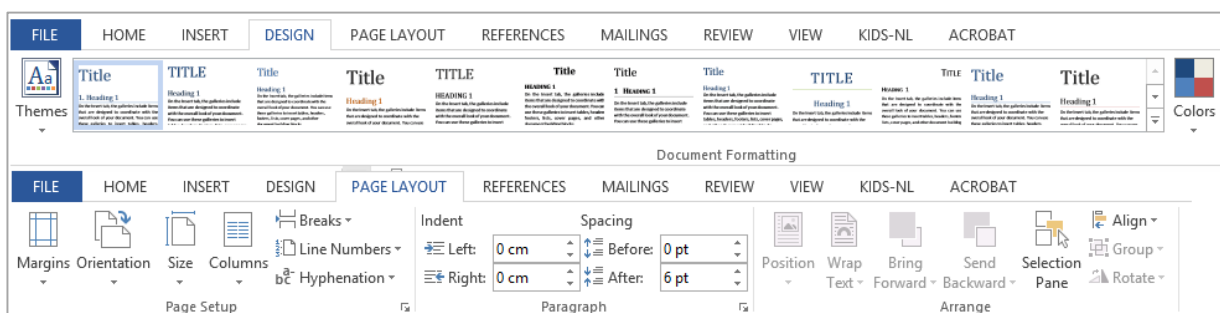


Figure 5-2: The scope of the Microsoft Word example, Design & Page Layout features

5.3.1.2. Determine vision [CA1, RP5, VO4]

After the scope is determined, the vision for that specific scope has to be determined too. Often, not all the input for the requirements comes from customers and users. Several decisions for the future might have been made already by the SPO and possibly have been included in a product roadmap. In this activity, it is also essential to take the influence of top management into account.

Example: the vision of Microsoft in creating this functionality was to make it simple for all users to create standard layouts and formats. They expect to keep that vision in future versions of Word. To stay competitive, novice users should also be able to use the design options that are available.

5.3.1.3. Determine degree of openness [CA1, VO3]

While it is important to have an own vision, it is a strength to have a certain degree of openness towards input from customers and users. In this activity, that degree of openness has to be determined. The vision might influence this degree, but vision and openness can also complement one another.

Example: while Microsoft has a very clear vision for this scope, they are open towards opinions of their users. The company will not take every idea into account, but when a suggestion has significant support and is inventive, they will research the potential for implementation.

5.3.1.4. Identify situational factors [SF1, SF2]

The situational factors influence the operations of a software company and should therefore also influence the techniques that are used for RE. A list of situational factors from the SPM literature was already presented in Section 3.4.1. The interviews added the factors of *strategic goals*, *type of software*, *product usability*, *frequency of use* and *number of users*. *Strategic goals* is largely overlapping with *company policy*, the *type of software* is described by many factors from the literature list (e.g. *product size*, *type of customers*). The final (but non-exhaustive) list of situational factors for the CCRE method is presented in Table 5-3.

Table 5-3: Situational factors for the CCRE method

Situational Factor	Value	Situational Factor	Value
Development philosophy	<i>Agile/Not agile</i>	Number of localizations	#
Size of organization	#FTE	Bug rate	# Bugs / time unit
Customer loyalty	1-10	New requirements rate	# Requirements / time unit
Customer satisfaction	1-10	Number of products	#
Customer variability	<i>Low - High</i>	Product age	# years
Number of customers	#	Product lifetime	# years
Type of customers	<i>Indiv/Enterprise (S/L)</i>	Product size	<i>Small - Large</i>
Number of users	#	Product tolerance	<i>Low – High</i>
Type of users	<i>Role in organization</i>	Product usability	<i>Low – High</i>
Frequency of use	<i>Daily/weekly/...</i>	Effect of company policy	<i>Low – High</i>
Market growth	<i>Decr./Constant/Incr.</i>	Customer involvement	1-10
Market size	# Potential customers	Effect of legislation	<i>Low – High</i>
Release frequency	# Releases / time unit	Partner involvement	1-10
Sector		Degree of customization	<i>Standardized/Tailored /Customized</i>
Standard dominance	<i>Low - High</i>	System complexity	<i>Low – High</i>
Variability of feature requests	<i>Low - High</i>	Geographical distribution of customers	<i>Specific/National /International</i>

For each of the factors, the value has to be determined, ideally with some rationalization. Some situational factors have a clear effect on the crowdsourcing potential. High customer involvement will make the use of crowdsourcing easier and more effective, since the customers are willing to participate. A very low number of releases makes crowdsourcing less effective, since users will not quickly see the results from their input. Other situational factors have mixed effects on the crowdsourcing potential, having both pros and cons. For

instance, a large number of customers increases the potential amount of input, but makes it hard to structure that input. Also, an old product might need innovation, but the motivation or resources to change a legacy product might be lacking. Determining the effects of this type of situational factors requires a thorough analysis.

Example: we will not give a complete list of the situational factors, but highlight some. The number of users is several hundreds of millions. The frequency of use is often daily to weekly. Effect of company policy on the product is probably pretty high. There's a high variability of customers and users, which include families, students, consultants, large industries, education institutions, etc. We give customer involvement a 5 and customer satisfaction a 7. These values could result from product forums and a customer survey.

5.3.1.5. Determine intended outcome [CA2]

Our interviews have shown that the intended outcome influences the stakeholders and RE techniques. To put it more generally, the intended outcome is a determinant for the whole crowdsourcing process and its success. Without looking too much at the aspects that were determined in the previous phases, the organization has to determine what they want to achieve by utilizing the CCRE method.

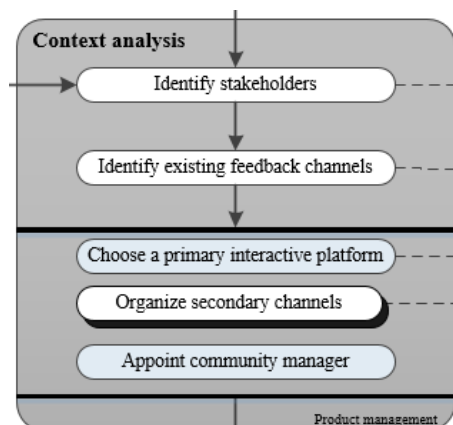
Example: with the input, Microsoft will presumably not alter the current version of Word. Great requirements that improve the usability of the design functionality will be integrated in Office 365. Additionally, Microsoft wants to create a stronger bond with its user base.

5.3.1.6. Conclude crowdsourcing potential

This is the concluding activity of the feasibility analysis. The organization's vision, openness and situational factors are mapped on the intended outcome, to see whether the outcome is achievable. More importantly, the potential of crowdsourcing and the CCRE method in achieving this outcome is determined. There is not one right way to conduct this activity, but one structured technique would be to list the opportunities and threats of each situational factor and the vision of the company. When the opportunities outweigh the threats, the company has a significant degree of openness and the intended outcome is achievable, the potential for crowdsourcing is high. This activity is a go/no-go moment in the method. Even a no-go decision is informative for the company, giving them insights on the combination of their context, vision and openness.

There is a high number of users, but they are not really involved. There is room for improvement in the satisfaction value; a 7 is not enough in this highly competitive field. The high variability of customers and users might make it difficult to come to a consensus but provide an interesting combination of perspectives.

Professional moderation and guidance is needed, but there is definitely potential for crowdsourcing.



5.3.2. Context analysis

This phase builds on the feasibility analysis by further analyzing the context around the defined scope and intended outcome. The stakeholders that can be involved in crowdsourcing are identified as well as the channels via which they currently provide feedback.

One interactive platform has to be selected as the main channel and the other channels have to be organized.

5.3.2.1. Identify stakeholders [CF3, EL5]

All stakeholders that are relevant in reaching the intended outcome have to be identified. Several situational factors

(number/type of users/customers, size of organization) provide a starting point for this identification. Important and often overlooked stakeholders are end-users, who experience the real problems, and domain experts, who understand the context. Other relevant stakeholders could be customers, development team members and project managers.

For more advanced stakeholder identification, a tool like StakeSource (Lim, Quercia, & Finkelstein, 2010) could be used. StakeSource crowdsources the activity of stakeholder identification and builds a social network by having the crowd identify and prioritize stakeholders. An interesting feature is the identification of potential problems, which can be found with the social network characteristics. For example, StakeSource identifies communication problems when a stakeholder is very central, while having a large distance from other reachable stakeholders.

Example: Microsoft Word has many different stakeholders. We can identify the users, the development teams (including designers, programmers and managers), the clients of Microsoft that buy licenses for their software, plug-in developers and even experts relevant to our scope (e.g. User Experience experts).

5.3.2.2. Identify existing feedback channels [CA3]

In practice, product managers are overloaded with ideas, questions, suggestions and complaints. During this activity, the channels through which these forms of input arrives are identified. All channels via which the stakeholders – who were identified in the previous activity – provide feedback have to be considered. The identification is essential to be able to structure the existing channels, choose the primary channel and reach the stakeholders.

Example: feedback on Microsoft Word is given via official channels as email, feedback forms and in phone calls, but also via unofficial channels such as Facebook, Twitter and a number of online forums.

5.3.2.3. Choose a primary interactive platform [CA4, FC1]

In this activity, the optimal channel is either selected from the existing channels or added as a new channel. The notion of platform implies that users can build on the channel by providing input. An essential characteristic of this platform is *interactivity*. Interactivity allows the organization and the crowd to interact about the provided input. *Optimal* means that the platform has to be effective and fit the situational factors of the organization; software with 1 million users probably needs a different platform than software with ten users. The degree of openness is also influential; when the crowdsourcer only wants to receive votes for design options, the platform should not allow content generation. Additionally, the platform has to be reachable for all stakeholders and provide the output that is necessary for later CCRE activities (i.e. allow for elicitation, negotiation, prioritization and identifying active contributors).

Existing platforms that could be used are GetSatisfaction¹⁰, UserVoice¹¹, Innovation Factory¹², Be-novative¹³, InnoCentive Idea Management¹⁴ or Wazoku¹⁵. Table 5-4 lists these platforms and shows their differences. An SPO might decide to develop a custom-made platform, which suits the needs of the organization.

¹⁰ www.getsatisfaction.com, accessed on December 2nd, 2014

¹¹ www.uservoice.com, accessed on December 2nd, 2014

¹² www.innovationfactory.eu, accessed on December 2nd, 2014

¹³ www.be-novative.com, accessed on December 2nd, 2014

¹⁴ <http://www.innocentive.com/innovation-solutions/innocentive-idea-management>, accessed on December 2nd, 2014

¹⁵ www.wazoku.com, accessed on December 2nd, 2014

Example: Microsoft already uses several channels, but for this method, we can select UserVoice as the primary channel. It is specifically focused on software feedback and the contextual information might be useful to analyze the needs of users.

Table 5-4: Overview of existing interactive platforms.

Name	Software-specific	Elicitation	Negotiation	Prioritization	Customer Support	Req. Evolution	Gamification	Extra
GetSatisfaction	Yes	Yes	Yes	Yes	Yes	No	Yes	"Give praise to the organization"
UserVoice	Yes	Yes	Yes	Yes	Yes	No	Internally	Sends contextual information
Innovation Factory	No	Yes	Yes	Yes	No	No	No	Idea Challenges
Be-novative	No	Yes	Yes	Yes	No	?	Yes	Groups, vision board
InnoCentive Idea Management	No	Yes	Yes	Yes	No	Yes	No	Workshops for screening & maturation
Wazoku	No	Yes	Yes	Yes	No	?	Yes	Opinion is weighted via algorithms

5.3.2.4. Organize secondary channels [CA3, EL2]

Even when a primary channel is chosen, this channel will be bypassed by some users. Therefore, other channels that were identified have to be organized. Organization means:

- Identifying the type of feedback that is provided through the channel;
- Creating a workflow to process that feedback;
- Controlling the workflow.

A workflow for channels with a large amount of feedback might be data mining. This idea is further elaborated in the 'Mine feedback from secondary channels' activity.

Example: Microsoft needs to come up with ways to organize the emails, phone calls and social media updates that provide needs for Office 2013. Let's say these will all be analyzed with a data mining tool and integrated into the UserVoice platform.

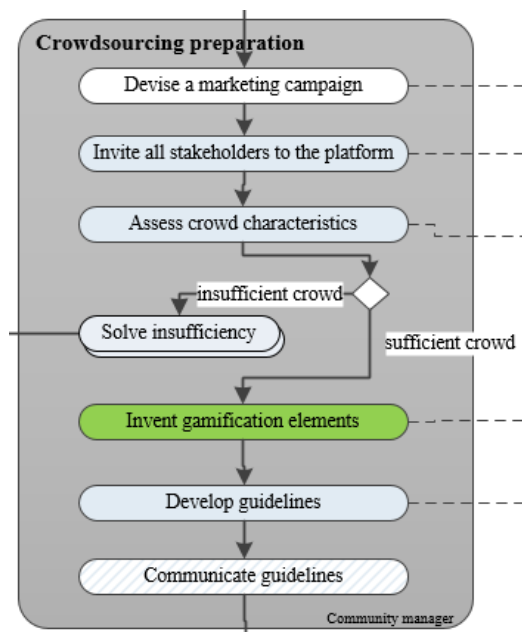
5.3.2.5. Appoint community manager [CC2]

This activity is a logical consequence of the presence of a crowd and a platform. Someone needs to be appointed to execute the Crowdsourcing preparation phase and moderate all the incoming input. Ideally, this person has the goal of making the platform as constructive and effective as possible. A bias towards needs or opinions of specific users or the SPO is unwanted. The community manager needs to know how the CCRE method works in order to provide the right guidance to the crowd.

Example: since we're talking about a very large project in a very large organization, the community manager will be a full-time job and a specialized person needs to be appointed to fulfill this role.

5.3.3. Crowdsourcing preparation

Before the actual involvement of the crowd can start, the crowd has to be formed, evaluated and prepared for the process.



5.3.3.1. Devise a marketing campaign [SI1, CF1]

Crowd-Centric Requirements Engineering will only be a success when stakeholders are involved. To mobilize stakeholders, the existence of the interactive platform has to be communicated and an incentive to be part of the community has to be provided. This marketing campaign is invented and developed in this activity of the method.

Example: Microsoft starts a marketing campaign on its technically oriented social media pages and blogs. Additionally, developers, important clients and experts will be invited personally. The campaign is called 'Invent your Word' and the best contributor will be invited to the global headquarters to spend a day with the development team.

5.3.3.2. Invite all stakeholders to the platform [EL3, SI4, CF2, CF3, RP6, UR2]

Once the marketing campaign is devised, it has to be executed. While the details might be different for every instantiation of the method, the relevant stakeholders that were identified in the previous phase have to be invited.

During the feasibility analysis, the scope of the method and openness within that scope was determined. Now this should be communicated to the invited stakeholders in a very concise but effective manner. A question with a short complementary description is an effective way to give stakeholders the right direction without steering their input too much.

Example: this is the actual execution of the campaign. The blogs will be written and the personal invitations are sent. The specific question that is asked is "How can the design and layout functionality of Word 2013 be simplified, while remaining effective?"

5.3.3.3. Assess crowd characteristics [CS2, CF4]

As several scholars have noted, a crowd should be undefined or even unknown (Hosseini et al., 2014a; Howe, 2006). However, the expert interviews and theory on Wisdom of Crowds (Surowiecki, 2005) have taught us that the crowd should be diverse, independent, decentralized, large and suitable for the task. Each of these characteristics has to be evaluated in this activity. After this evaluation, it should be clear whether the crowd is sufficient.

Example: Microsoft sent a sample of the reached crowd a questionnaire about their background. This shows that most stakeholders are experienced IT users and thus a problem in diversity is observed. The crowd is however large enough, independent and decentralized.

5.3.3.4. Solve insufficiency

When the crowd is considered insufficient in the previous activity, the insufficiency has to be solved. The concrete solution and implementation of this solution is dependent on the characteristics that were not met. When a crowd is not diverse, minority groups need to be identified and invited. When a crowd is not independent, communication with the members of the crowd should take place to develop independency. Most of these solutions lead to the invitation of additional stakeholders and therefore a flow to this activity is depicted.

Example: to solve the observed problem in crowd diversity, Microsoft promotes the feedback platform on more places where novice users can be found, e.g. their general Facebook and Twitter page. In addition, Microsoft includes a step-by-step guide to show how simple it is to provide input.

5.3.3.5. Choose gamification elements [GE1-7]

As one of the shortcomings of current RE processes is the lack of incentives, the CCRE method should provide adequate incentives. In order to do this, gamification elements are devised, chosen and implemented. The choice for and implementation of these elements is very dependent on the situational factors that were identified in the first phase, which is called situational relevance by Nicholson (2012). The experts suggested exploration, group forming, roles and rewards as useful techniques. Our literature study has identified accelerated feedback cycles, clear goals and rules of play, a compelling narrative, challenging tasks (Gartner, 2011), situated motivational affordance, universal design for learning and player-generated content (Nicholson, 2012) as important elements. Pointsification in the form of points, leaderboards and achievements can provide a short-term incentive (Hamari et al., 2014).

Example: UserVoice itself doesn't include gamification features for the users, only for its administrators. Microsoft will therefore keep a ranking of the most popular needs and corresponding contributors on its own social media, which stakeholders can share with their connections. Additionally, stakeholders weekly receive a certain amount of points, based on their activity, that they can allocate to needs of others.

5.3.3.6. Develop guidelines [CC1, CC3]

To prevent misuse and chaos, to promote constructive usage of the interactive platform and to communicate the CCRE process, guidelines are developed in this activity. A distinction can be made between platform guidelines and process guidelines. Platform guidelines explain appropriate usage of the platform, e.g. "stakeholder input has to be relevant to the specific question". A process guideline describes the process beyond the platform, e.g. "active stakeholders will be invited to participate in focus groups".

As with many factors in this method, the guidelines are situation-dependent. When customer involvement is low, inviting stakeholders to the focus group should be done extra carefully. When the variety in the size of customers is high, advice CC3 might need to be taken into account, to prevent that one customer group dominates the voting process.

Example: all input on the platform has to be relevant to our scope, that's a platform guideline. The process guidelines describe that the focus groups are small online forums on which requirements are further refined and design options can be evaluated.

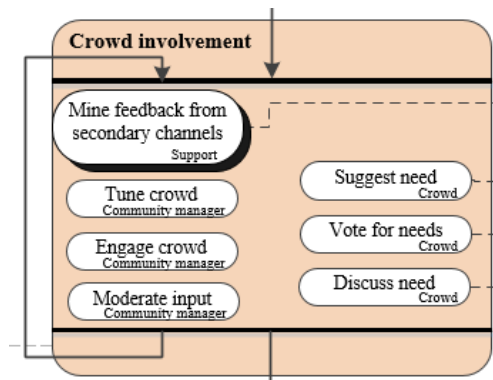
5.3.3.7. Communicate guidelines [VA2]

This activity is rather simple; the guidelines that were developed in the previous activity have to be communicated to all stakeholders. This can happen via different channels (e.g. in the invitation, on the platform) and at different points in time (before, during or after crowd involvement). Guidelines after crowd involvement should only be focused on follow-up phases such as the focus groups or information about (unfinished) pre-releases.

Example: to sign up for the platform and participate in the crowdsourcing task, stakeholders have to read and accept the guidelines.

5.3.4. Crowd involvement [UR1, SI4, CS1, CS2, RP6]

This is the phase in which the crowdsourcing is conducted. Invited stakeholders provide their input on the interactive platform and feedback on the other channels is collected. It is essential in this phase to keep the barriers to participate very low.



5.3.4.1. Mine feedback from secondary channels [CA3, EL2]

As mentioned earlier in this chapter, stakeholders will presumably partially bypass the chosen channel. That feedback should not be ignored, but mined in order to gather valuable needs. A very recent phenomenon is the use of social media for giving (unsolicited) feedback. (Horrigan, 2008) has conducted a study on 2400 American adults and found that 30% of those adults have posted an online comment or review regarding a product, 32% has provided a rating.

Two techniques to effectively analyze this feedback might be opinion mining and sentiment analysis (Pang & Lee, 2006). Opinion mining strives to generate a list of product attributes and aggregate opinions about each of them (Dave, Way, Lawrence, & Pennock, 2003). Sentiment analysis is the computational treatment of opinion, sentiment, and subjectivity in text and therefore can be used as an interchangeable term (Pang & Lee, 2006).

Since this thesis is focused on the application of crowdsourcing in RE and not on the application of data analytics and business intelligence, this concept will not be further described. However, it is a very promising area for future research.

Example: the other channels (email, social media, etc.) were already structured. Opinion mining can be used to measure whether the general opinion towards the crowdsourcing task is positive. A more sophisticated method is going to be used to mine suggestions from these channels and integrate them in the interactive platform. Somebody might tweet on Twitter "Why can I not create my own theme with one click in Word?" This is posted as a need on the platform and the author is notified and invited to further elaborate on it.

5.3.4.2. Suggest need [CS4, EL4]

The crowd should be able to share their needs with other stakeholders. Within the scope of the question that was communicated to the crowd, all kinds of needs should be expected and allowed.

This activity is largely similar to *needs elicitation* as it was depicted in the model that combined RE and SPM in Figure 3-2. Mapping the need on the requirement state model, the first state of *Candidate* is reached (Regnell & Brinkkemper, 2005).

Example: one of the UserVoice users suggests to "Remove the margin options from the main toolbar, this is too advanced".

5.3.4.3. Discuss need [CS1, NE1-3]

Since crowdsourcing has the goals to let stakeholders learn from other stakeholder groups and reach a consensus, the crowd should be enabled to discuss the suggested needs. As a discussion technique, needs could be branched. This prevents users from hijacking existing needs [EL6] and instead motivates them to improve those needs. Discussion and branching are good mechanisms to turn a vision, theme or concept into a requirement definition (Vlaanderen et al., 2011).

This activity can be seen as the *requirements negotiation* step, that is often lacking in the traditional RE process. In Section 3.3.4, needs analysis and negotiation was described and factors that can be analyzed were listed. Verifying completeness, evaluating feasibility, identifying gaps, obstacles to requirements satisfaction and missing assumptions can be conducted in this activity. More difficult are the grouping, complexity

analysis and identification of unknown requirements interactions, which should be performed at a later stage.

Example: a user comments on the previous need and states “I don’t agree with you, I think many novice users find this an easy way to create their layout”.

5.3.4.4. Vote for need [UR3, NE2,3, PR1-3,5]

This activity is the first part of *requirements prioritization*. The crowd should be able to communicate whether they agree or disagree with the suggested needs. This is in line with the work of Berander and Andrews (2005), who state that the simplest appropriate technique should be chosen for prioritization. At this point, no disagreement needs to be solved and no decision needs to be taken, which makes a simple voting mechanism appropriate.

Since experts have told that it is not desirable to have users determine the ultimate priorities and literature shows that prioritization techniques can complement one another (Berander & Andrews, 2005), prioritization will be continued in activities 5.3 and 7.4.

Example: users can allocate their points to the needs of others, so someone might choose to give 3 points to the previously described need.

5.3.4.5. Moderate input [CC2, CC5]

When needs or comments are irrelevant, moderation should be performed. It is advisable to do this in a more constructive way than just deleting needs. The crowdsourcer should give feedback to the stakeholder that provided the irrelevant content, explaining how he or she can provide more useful feedback.

Example: someone posts “the commenting feature of Word looks outdated” as a need. The moderator deletes the need and sends the person a private message stating that the commenting feature is not relevant to the design and layout functionality.

5.3.4.6. Tune crowd [CS2, CF4]

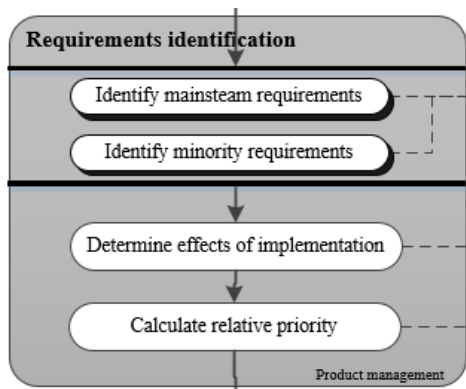
This activity is a combination of ‘Assess crowd characteristics’, ‘Solve insufficiency’ and potentially ‘Identify stakeholders’ and ‘Invite all stakeholders to the platform’. Since the crowd is dynamic, its characteristics might change during the ‘Crowd involvement’ phase. These characteristics should therefore constantly be assessed and potential insufficiencies should be solved. A good way to do this is to monitor the size of the crowd and the variance in type of stakeholders that form the crowd.

Example: after a week, it becomes apparent that the UX experts have not yet signed up for the UserVoice platform. The product management team of Word therefore sends a reminder to these stakeholders.

5.3.4.7. Engage crowd

The presence of gamification elements on the platform might not provide enough motivation to the crowd to stay active. The stakeholders have to be updated on their leaderboard position, discussions and votes on their needs or comments and the selection of requirements for focus groups. These updates can lead to improved engagement.

Example: all stakeholders get email notifications when a person comments or votes on their need or comment. Additionally, blog posts are written after focus groups and shared in email updates to the stakeholders.



5.3.5. Requirements identification [RP5]

In this phase, the needs that were suggested, discussed and voted upon will be analyzed as requirements. This means that the conversion from a need to a requirements has to be made. The expertise of the product management professionals is essential in this phase, since they have to make assumptions that cannot be made by the users. Related to Section 3.3.4, grouping, complexity and unknown requirements interactions should also be analyzed at this phase.

In one case, a need might be very well-formulated and refined by the contributing stakeholders (e.g. “the system needs to do x”), which makes direct conversion to a requirements possible. However, it might also be the case that a need is phrased as a vague ‘ask’ (e.g. “I want to be able to do x more efficiently”). In that case, product management should get involved in the discussion and stimulate the generation of concrete solutions, working towards a product requirement.

5.3.5.1. Identify mainstream requirements [PR1]

Mainstream requirements are the requirements with much support. Support can be measured by counting votes and comments, where the distinction between agreement (more support) and disagreement (less support) needs to be made.

Example: Microsoft selects the top 5 of needs, based on total points and comments. Using the comments, the actual requirements are clarified. One mainstream requirements states that “the margin option should get a less prominent position”. Another argues that “there should be more professional-looking themes to choose from”.

5.3.5.2. Identify minority requirements [CS3, PR1]

Minority requirements are requirements that did not necessarily get many positive comments and agreeing votes, but do have a high value. Small user groups might require a specific functionality, which has to be identified by product management. Deciding whether requirements are valuable should be done in consistency with the vision that was defined in the first phase, but also needs to be determined case-by-case.

Example: a few users from the publishing industry described that they spend many clicks on creating columns. Therefore, the minority requirement “a shortcut should enable the creation of columns” is developed.

5.3.5.3. Determine effects of implementation [VO2, PR2, RP5]

For each of the requirements identified in the two previous activities, the effects of implementation have to be determined. Those effects are the business value that the implementation will generate, the penalty that the business would suffer if the feature is not included, the costs of implementing the requirement and the risk that it might pose to the business. The effects are based on the semi-quantitative analytical approach of Wiegers (1999) and provide a second step in *requirements prioritization*.

Example: the ‘shortcut-requirement’ is only supported by a few users from small clients and therefore has a low business value: 2. The penalty that would be suffered due to exclusion of the functionality is minimal, since these users do not have many alternative products: 3. The cost of implementing the requirement is substantial, since the shortcut cannot be used for other functionalities: 7. The risk that it might pose to the business is very low: 2.

The ‘themes-requirement’ had much more support and the comments show that it might have much value for important user groups. The business value therefore is 7. The penalty that would be suffered is a little lower,

since the substitution product for professional themes is also Microsoft-owned (Microsoft Publisher): a 5. The costs of implementing are average, since some external designers need to be hired: 5. The risk that is might pose to the business is almost non-existent: 1.

5.3.5.4. Calculate relative priority [PR2, PR3]

To further follow the approach of (Wiegers, 1999), the relative priority of the requirement has to be calculated. This can be done by giving a separate weight to the business value, penalty, cost and risk. For the benefit, costs and risk, the percentage of the accumulated benefit, cost or risks of all requirements has to be calculated. Subsequently, the following formulas can be used to calculate the relative priority of each requirement:

- 1 Benefit = Business value * Business value weight + Penalty * Penalty weight
- 2 Benefit% = Benefit / Accumulated benefit
- 3 Relative priority = Benefit% / (Cost * Cost weight * Cost% + Risk * Risk weight * Risk%)

Example: the weight of each factor is 1. The accumulated benefit, cost and risk are all 100.

Short-cut requirement:

$$\text{Benefit} = 2 * 1 + 3 * 1 = 5$$

$$\text{Benefit\%} = 5/100 = 5\%$$

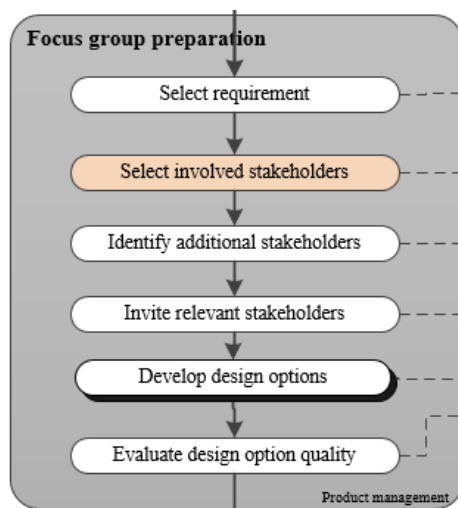
$$\text{Relative priority} = 5\% / (7 * 1 * 7\% + 2 * 1 * 2\%) = 0,094$$

Theme requirement:

$$\text{Benefit} = 7 * 1 + 5 * 1 = 12$$

$$\text{Benefit\%} = 12/100 = 12\%$$

$$\text{Relative priority} = 12\% / (5 * 1 * 5\% + 1 * 1 * 1\%) = 0,462$$



5.3.6. Focus group preparation [VO2, CF5]

For specific (sets of) requirements that were identified in the previous phases and have positive effects for the business, focus groups are organized. These focus groups have the goal to further develop the requirements by looking at design options. In this phase, those focus groups are prepared.

5.3.6.1. Select requirement [PR6, CA2]

Activity 5.3 was used to determine the positive and negative effects of implementing requirements, which led to a relative priority in activity 5.4. Based on this analysis and all the input that was acquired during the *crowd involvement* phase, a selection of requirements that are desirable to implement can be made. Product management has the responsibility to select important requirements, but needs to take the input of the

whole crowd of stakeholders into account. Mapping the need on the requirement state model, the second state of *Approved* is reached (Regnell & Brinkkemper, 2005).

The following activities have to be performed for each selected requirement or set of consistent requirements.

Example: the 'shortcut-requirement' has a lower relative priority than other requirements and is therefore not selected. Similar to the 'theme-requirement', the 'margin-requirement' is high on the list. It is therefore selected in this activity.

5.3.6.2. Select involved stakeholders [SI4, GE4]

All the stakeholders that had an important role in the contributions to the requirement that is selected, have to be selected in this activity. These stakeholders at least include the initiator and most active commenters (both positive and negative).

Example: Many users and several developers commented on and voted for the 'margin-requirement'. The stakeholders with the most votes and comments as well as the initiator are selected.

5.3.6.3. Identify additional stakeholders

The list that was developed in the previous activity might be incomplete, since some relevant stakeholders might have missed the corresponding need or might even not be active on the platform. Since these stakeholders are important in the development of a requirement, they still need to be involved in the focus group. During this activity, these stakeholders are identified.

Example: several designers and a manager are involved in deciding the position of features in Word toolbars. They are identified as additional stakeholders.

5.3.6.4. Invite relevant stakeholders

The stakeholders that were identified in the two previous activities are invited in this activity. For the invitation, the most suitable communication channel has to be selected and might have been part of the feedback channels that were identified in the second phase.

Example: the identified stakeholders receive a private message on UserVoice, which states that they get the chance to further describe their opinion. The chance to win a trip to the headquarters is repeated.

5.3.6.5. Develop design options [SI2]

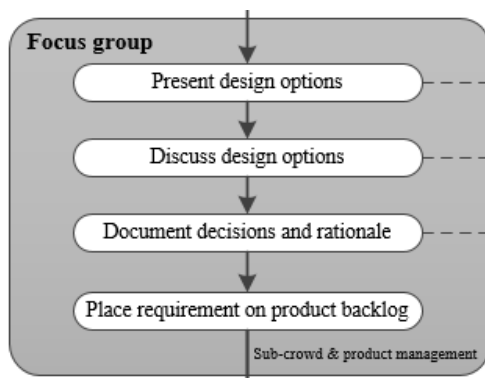
Based on the requirement as it is described until now, design options that fulfill this requirement can be designed. These design options make it easier for the focus group to give feedback, since they have something visual (and maybe even functional) to comment on. Possible ways to create the design options are use case development, paper prototyping, mockup development or functional prototyping. Suggestions that are made in comments or in the suggested need are an ideal starting point for a design option.

Example: the designers that were selected create several design options with the use of paper prototyping. In one, the margin options are placed under an 'advanced' button. In another, all page layout options are a sub-functionality in design.

5.3.6.6. Evaluate design option quality [VA3]

While many stakeholders might give useful suggestions and comments in the *crowd involvement* phase, not all quality aspects are crowdsourcable. Examples are security, maintainability and compliance with regulations. These aspects therefore need to be evaluated by the product management team or quality specialists. Design options that lack necessary quality need to be eliminated in this activity.

Example: all design options adhere to security, maintainability and design standards. Therefore, none of the design options is eliminated; all will be presented in the focus group.



5.3.7. Focus group [CF5, SI3, RP4]

After preparation, the actual focus group can take place. This focus group might be either online or offline, depending on the availability and geographical distribution of stakeholders. When the need is still in the vision, theme or concept stage, this is the time that it should be turned into a requirement definition. While the original agile requirements refinery lets the SPM team perform the first definition step (Vlaanderen et al., 2011), the participatory nature is essential in this CCRE phase.

5.3.7.1. Present design options [SI2, CS4]

The design options that were developed in the previous phase and passed the quality check, are presented to the invited stakeholders in this activity. The form of presentation is of course dependent on the type of design option; a prototype might be tested, while a mockup can lead to a discussion.

Example: the paper prototypes are shown and explained on the online forum.

5.3.7.2. Discuss design options [CS1, VA1]

Since multiple design options were developed and presented, these options can be compared and discussed. Arguments for or against a specific option might be given, options can be improved and options might even be chosen or eliminated.

Example: the selected stakeholders comment on the design option, someone might say that “it is hard to find the lay-out options when they are positioned under design”.

5.3.7.3. Document decisions and rationale [RP7]

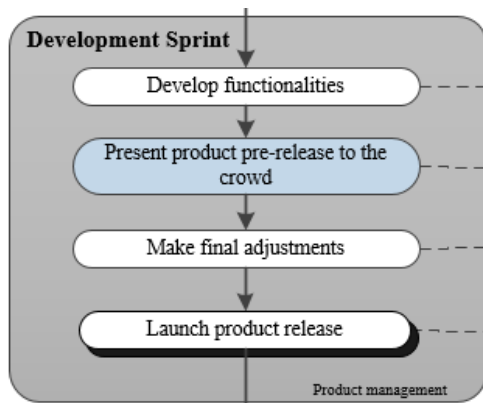
The discussion of the previous activity is documented in this activity. (Lamsweerde, 2000) lists documentation as one of the main activities of RE and describes it as “the various decisions made during the process are documented together with their underlying rationale and assumptions”. The documentation in this activity therefore includes decisions on eliminated and accepted options, but also contains the rationale behind these decisions, to optimize the traceability. By documenting the elimination of options, duplicate work is prevented and future decision making gains improved efficiency. Mapping the need on the requirement state model, the third state of *Specified* is reached (Regnell & Brinkkemper, 2005).

Example: the product management decides to create an ‘advanced’ button and documents this decision in a requirements database. The option in which ‘page layout’ is a sub-functionality of design is eliminated because it strongly decreases the findability of some features. By documenting this rationale, it can be taken into account the next time that a sub-functionality is considered.

5.3.7.4. Place requirement on product backlog [RP8, PR3, PR6]

By now, a clear description of the requirement should be available and linked to all corresponding input. The requirement can be placed on the product backlog, in which the final step of prioritization is implied. Mapping the need on the requirement state model, the fourth state of *Planned* is reached (Regnell & Brinkkemper, 2005).

Example: the change will be implemented in the next Sprint and gets a specific position.



5.3.8. Development Sprint [RP2, RP3]

This phase is generic and has to be instantiated in a way that is favorable to the company. Every software company has its own development methodologies and processes. The CCRE method should be tailored to the situation at hand. The Development Sprint is typically not within the scope of RE, but included in the CCRE method to show the result of the crowdsourced input.

5.3.8.1. Develop functionalities

In this activity, the actual functionalities that are described in the requirement on the Sprint Backlog are developed. The design options and documented discussions serve as an input for the design and development decisions. The traceability of requirements and the corresponding needs should make it easier to make these decisions.

Mapping the need on the requirement state model, the fifth state of *Developed* is reached (Regnell & Brinkkemper, 2005).

Example: the 'advanced' button is developed and integrated in the beta version of Office 365.

5.3.8.2. Present product pre-release to the crowd [VA1,4, GE4]

When all functionalities from the Sprint Backlog are developed, the improved product can be pre-released. This pre-release means that the crowd is able to try a beta version of the release and provide feedback on it. This activity can be seen as an acceptance test. Since the 'Crowd involvement' phase is continuous and thus runs parallel to this activity, the stakeholders can suggest, discuss and vote for needs in regard to the pre-release. In addition to being useful for the developing company, early access should be promoted as a reward for being involved.

Example: everyone that was active in developing the requirements is invited to test the first version of office 365, which includes some of the suggested requirements, including the repositioned 'margins-option'.

5.3.8.3. Make final adjustments [RP3]

Based on the feedback that is gathered in the previous activity, the final adjustments to the product can be made. Completely new needs will not be taken into account; these should be considered in new 'Requirement identifications'. Assuming that unit and integration tests are performed during the development activity, the sixth state of *Verified* is reached in the requirement state model (Regnell & Brinkkemper, 2005).

Example: the crowd likes the new version, but thinks that the button might get a slightly different styling. This is quickly done in the reserved time in the subsequent Sprint.

5.3.8.4. Launch product release [FC2-3]

After all the final adjustments have been developed and implemented, the final version of the release is ready and can be deployed to the complete crowd. Mapping the need on the requirement state model, the seventh and final state of *Released* is reached (Regnell & Brinkkemper, 2005). When the product is released, feedback should be provided on the platform to let the stakeholders know which requirements were implemented. Ideally, the feedback is visualized and users can explore the origin and outcome of features and requirements. Contributors to the requirements should at least be rewarded on the platform.

Example: the public version of Office 365 contains the new functionality, including the final adjustment. One of the contributors to the developed functionality is invited to the headquarters.

5.4. Concepts

Table 5-5 lists the concepts that are included in the CCRE method, together with their definitions.

Table 5-5: The concepts and their definitions in the CCRE method

Concept	Definition
SCOPE	The area on which the CCRE method will be focused. This might be a specific product, feature or type of users in a specific timeframe.
VISION	The ideas that the organization has regarding the future of the specific SCOPE. This might be a product roadmap or certain decisions that have already been made by the product or upper management.
DEGREE OF OPENNESS	The willingness of the organization to listen to and act upon the opinion of other stakeholders, within the defined VISION and SCOPE.
SITUATIONAL FACTORS	Factors that describe the situational context in which the organization has to operate and to which the RE and SPM processes thus have to be fine-tuned (based on Bekkers, Spruit et al., 2010).
INTENDED OUTCOME	The result that the organization wants to achieve by conducting the CCRE method.
CROWDSOURCING POTENTIAL	The potential of continuing the CCRE method, based on the previously determined concepts.
STAKEHOLDER	Anyone who is relevant to the defined SCOPE and INTENDED OUTCOME of the method. Stakeholders can include developers, project managers, (prospective) clients, (prospective) users, experts and partners.
FEEDBACK CHANNEL	An online or offline channel through which STAKEHOLDERS currently give feedback on the software.
PLATFORM	An open and interactive platform that is chosen as the primary channel to receive feedback.
POLICY	A description of the workflow for other FEEDBACK CHANNELS.
MARKETING CAMPAIGN	A communication mechanism to invite stakeholders to become involved in the CROWD.
CROWD	The network of STAKEHOLDERS.
GAMIFICATION ELEMENT	A game design element that aims to increase the level of engagement, change behavior or stimulate innovation (based on Deterding et al., 2011; Gartner, 2011).
GUIDELINE	A prescription for the use of the platform or description of the progress of the involvement process. Platform guidelines describe appropriate usage of the platform, process guidelines describe the CCRE process.
FEEDBACK	Relevant input from stakeholders that was shared on other channels than the chosen PLATFORM.
NEED	A raw need regarding the SCOPE that is shared on the PLATFORM by a STAKEHOLDER. Needs have varying forms, including questions, complaints and ideas. Votes on the need might be in favor of or against the need.
COMMENT	A response to a NEED, shared by a STAKEHOLDER on the PLATFORM.
REQUIREMENT	A well-defined 'thing' that mandates that something must be accomplished, transformed, produced or provided (based on Harwell et al., 1993). It is a further developed NEED. Each requirement has a certain potential business value, penalty for not implementing, cost of implementation and risk of

	implementation.
SELECTED REQUIREMENT	A REQUIREMENT that is selected to further develop in a FOCUS GROUP.
SELECTED STAKEHOLDER	A STAKEHOLDER that has contributed to the development of a SELECTED REQUIREMENT.
SUB-CROWD	A network of SELECTED STAKEHOLDERS for a specific SELECTED REQUIREMENT.
DESIGN OPTION	A design of a potential implementation of a SELECTED REQUIREMENT. This design could be a low fidelity prototype, but might also have some working functionality.
FOCUS GROUP	An online or offline setting in which DESIGN OPTIONS are presented to a SUB-CROWD.
DECISION	A choice – made in a FOCUS GROUP – to accept, reject or change a DESIGN OPTION.
DOCUMENTATION	A written transcript of the DECISIONS and the supporting rationale, discussed in a FOCUS GROUP.
FUNCTIONALITY	An implemented DESIGN OPTION.
PRE-RELEASE	An initial version of the product RELEASE, in which FUNCTIONALITIES are implemented.
ADJUSTMENT	A change of certain elements of the PRE-RELEASE.
RELEASE	The finished product version, developed in the current sprint.

5.5. CCRE drivers

This section contains the method results other than activities and concepts that are listed in the previous two sections. We call these results the six *CCRE drivers*.

5.5.1. Continuous phases [CS2, RP1]

Instead of the linear sequential phases of traditional RE, CCRE consists of three preparatory phases and five continuous phases. When the ‘Focus group’ for a specific set of requirements is being prepared, the ‘Crowd involvement’ and ‘Requirements identification’ phases can still be ongoing. When an organization is comfortable with CCRE, it might even be possible to run multiple instantiations of the method simultaneously, in which each instantiation has its own scope.

5.5.2. Staged participation [RP4]

The stakeholders are involved in different phases; during crowd involvement, focus groups and the development sprint. Each of those phases contains requirements from a certain level of detail and maturity. By separating the phases, stakeholders experience the results of their input and are encouraged to bring requirements to ‘the next level’.

5.5.3. An open, interactive platform [CS1, CC4, FC1, NE1]

Participants should be guided to be relevant to the defined scope, but should not be steered into a specific direction. This would decrease the value of crowdsourcing. In addition, stakeholders from all available backgrounds should interact about the given input in order to learn from each other and potentially reach a consensus.

5.5.4. Result-driven involvement [FC2, FC3, RP2]

While some literature has been focusing on the development of requirement models and specifications, the outcome of CCRE should be functionality that has been developed by the crowd. Therefore, the whole

method should be focused on this outcome. Involvement has to be visualized and stakeholders should be able to see timely results of their input.

5.5.5. Optimal traceability [RP7]

The different levels of requirements detail and maturity in the different phases make it possible to see where requirements came from. A specific need might have been branched into a new need, identified as a mainstream requirements and implemented as a functionality. The CCRE method makes it possible to see which stakeholders were involved and what reasons were promoting or opposing the need.

5.5.6. Stakeholder transparency [PR4]

During the method, the perspective and the constructiveness of stakeholders should be transparent. Knowing whether someone is a user or developer and seeing the historical usefulness of someone's needs should give a company insight on the importance of input.

5.6. Excluded and modified advices

Only one of the expert advices has been excluded and one advice has been modified. [EL1] stated that companies should not suggest their own ideas, to maintain a confident image. However, triggering a discussion might be very effective in motivating users. To prevent users from thinking that the company is self-doubting, the company can complement their suggestions with a clearly stated vision.

[PR4] – Give weight to stakeholders – has been transformed into the sixth driver. Giving the stakeholder actual weights would undermine the democracy that should be created during crowd involvement. However, the role and competence of users should be transparent and can be used in the decision-making process. A need that has little support, but has been raised by an important client, might be identified as a minority requirement with a high business value.

6. The prototype: Refine

6.1. Introduction

Instead of selecting an existing interactive platform to collect crowd input, we have chosen to develop our own platform: *Refine*. The name refers to **R**equirements Engineering and the main process that is supported by the platform: *refining* stakeholders' needs.

Refine allows users to suggest needs, comment on needs and other comments, branch needs and vote for needs and comments. This structure is depicted in the Class Diagram (CD) in Figure 6-1. The coloring of the diagram is similar to the PDD, which means that **blue** classes and variables implement crowdsourcing, **green** implements gamification and **orange** implements both. The crowdsourcing aspects include content, prioritization and roles; the gamification aspects are points and roles. The diagram does not contain any surprising design decisions. Users create needs, comments and votes. Comments can be either connected to other comments or to needs. Votes can be on needs or comments. Since users cannot vote against comments, only the voteCount is stored in that class. For needs, the votesUp and votesDown are stored; users can agree or disagree with needs.

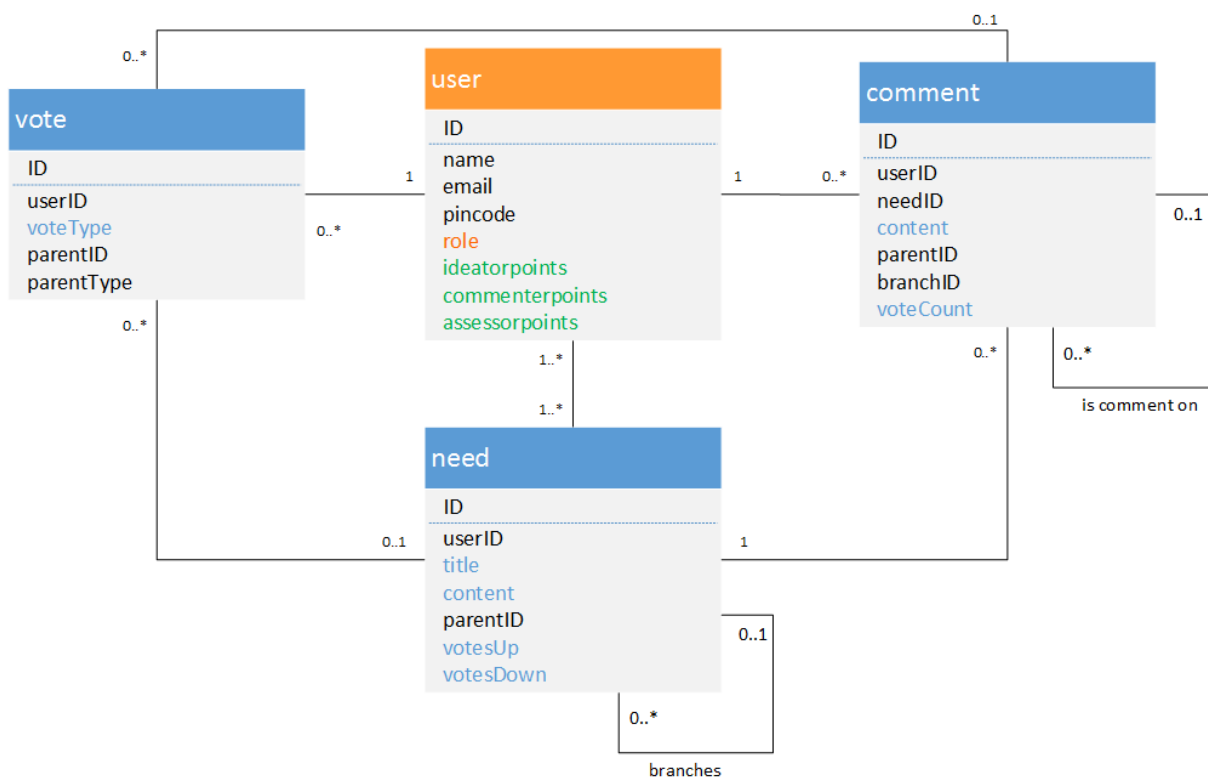


Figure 6-1: The CD of Refine

A Functional Architecture (FA) is displayed in Figure 6-2. This FA is focused on the ‘Crowd Involvement’ phase of CCRE. The coloring of the diagram is similar to the PDD and CD. The first picture shows the positioning of ‘Crowd Involvement’ in relation to other functions; the second picture gives the details of the ‘Crowd Involvement’ module. ‘Crowdsourcing preparation’ and ‘Requirements identification’ are therefore only

shown on a high level. While these functions are not the core crowdsourcing functions, they are essential in preparing the crowdsourcing function and processing its results and therefore have a blue color.

Relating the FA to the PDD, Crowd management includes the activities 'Tune crowd' and 'Engage crowd'. Moderation is the controlling function, while the suggestion of needs, commenting and branching are operational functions. Gamification is supportive and uses values provided by operational functions to create leaderboards and motivate users for constructive performance of the actions. All striped elements are implemented in *Refine*.

Relating the platform to the CCRE method, it supports the 'Crowd involvement' phase. Ideally, it also supports the start of the 'Requirements identification' phase, by ordering the resulting needs and suggesting concrete solutions in the form of useful discussions.

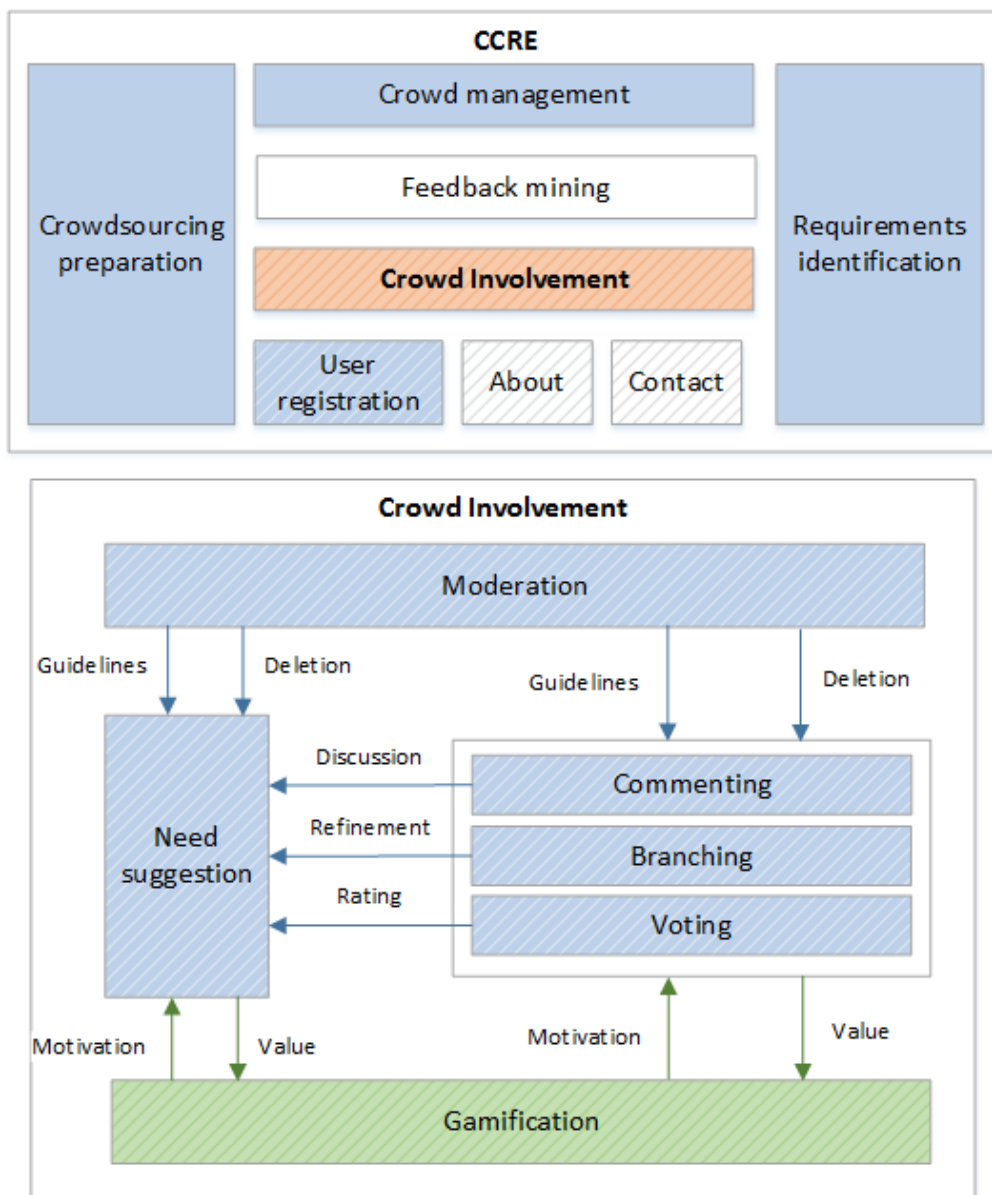


Figure 6-2: The FA of the crowdsourcing phase of CCRE and the Crowd Involvement module

6.2. Technical specifications

The platform has been built using JavaScript (including jQuery), HTML, CSS, PHP and MySQL. The interface is built on the Bootstrap¹⁶ framework. The choice for these technologies was made because of personal experience and suitability to quickly build a web prototype. For the gamification elements, the Application Programming Interface of PlayBasis¹⁷ was used. This means that the values for the points (that are variables of the user class in Figure 6-1) are stored in Playbasis' databases.

6.3. Gamification elements

Based on the social factors that improve continued use of a gamified service and the intention to WOM (Hamari & Koivisto, 2013), several game elements were implemented in the prototype. The selection was based on advices from the experts; [GE4] recommended roles, group forming, exploration and rewards. Leaderboards provide a good way to visualize the rewards (points) and are an important game element according to the study of Hamari et al. (2014). Endorsements are automatically included with the users' ability to prioritize.

Group forming, roles and exploration should improve the effects of network exposure, by making a network more accessible. The roles, resources & points and endorsements create a reciprocal benefit by offering something in return for contributions. Points and leaderboards should lead to recognition. Group forming leads to social influence, since users want to comply or even identify with a group. Endorsements lead to recognition as well as social influence. Table 6-1 gives a clear overview of the mapping of game elements on social factors.

Table 6-1: Mapping of gamification elements on social factors that should be triggered.

Social factors / Game elements	Network exposure	Social Influence	Recognition	Reciprocal benefit
Roles	X			X
Resources & points			X	X
Leaderboards			X	
Group forming	X	X		
Exploration	X			
Endorsements		X	X	X

Deterding et al. (2011) differentiate five levels of game design elements:

- Game interface design patterns – “Common, successful interaction design components and design solutions for a known problem in a context, including prototypical implementations”;
- Game design patterns and mechanisms – “Commonly recurring parts of the design of a game that concern gameplay”;
- Game design principles and heuristics – “Evaluative guidelines to approach a design problem or analyze a given design solution”;
- Game models – “Conceptual models of the components of games or game experience”;
- Game design methods – “Game design-specific practices and processes”.

¹⁶ www.getbootstrap.com, accessed on December 2nd, 2014

¹⁷ www.playbasis.com, accessed on December 2nd, 2014

The elements that are implemented in *Refine* cover the first four categories and are identified for each specific gamification element.

6.3.1. Roles

Category: game design principles and heuristics

Users of *Refine* can fulfill three roles: ideator, commenter and assessor. No choice has to be made; one user could fulfill one, two or all three roles. The proficiency in a role is represented by ideator points, commenter points and assessor points.

6.3.2. Resources and points

Category: game design patterns and mechanisms

Points can be earned by all actions that were listed in the introduction of this chapter. In addition, getting a vote on a need or comment generates points. Coins (resources) have to be spent to perform certain actions. This decision was made to let users think twice before they share a need or comment and promote constructiveness. Table 6-2 lists the resources and points that are spent and earned with the various actions. The values were determined in an exploratory manner and are yet to be validated. However, the value of the earned points emphasizes the importance of sharing and branching needs. The relatively high numbers are chosen to stimulate idea generation and building on existing ideas. The available number of coins allows users to share two or three needs and several comments if the user has not received votes.

Only the effects of actions were shared with the users, the passive effects were not published.

Table 6-2: Resources and points that are spent and earned with various actions

Action	Coins	Points
Register	+10	-
Sharing a need	-3	+5 ideator
Commenting	-1	+1 commenter
Voting	-	+1 assessor
Branching	-3	+5 ideator +1 commenter
Passive		
Branched need	+1	+3 ideator
Vote on comment	+1	+1 commenter
'Agree' vote on need	+1	+1 ideator
'Disagree' vote on need	-	-1 ideator point

6.3.3. Leaderboards

Category: game interface design patterns

Three types of leaderboards were used:

- An overall leaderboard, which shows the ranking based on accumulated points;
- Role-specific leaderboards, which show the ranking for ideator, commenter or assessor points;
- Need-specific leaderboards, which show the ranking for the accumulated points for one specific need. The ideator is marked by a badge.

The overall and role-specific leaderboards are shown on the leaderboards page, the need-specific leaderboards are shown on the need details pages.

6.3.4. Group forming

Category: game models

Group forming is stimulated by two mechanisms: the transparency of the stakeholders' background and the separation of leaderboards per need. On the registration form, users select their background (e.g. developer, end-user), which is publicly shown on their profile. This transparency should clarify their perspective and increase the learning between the different stakeholder groups.

The separation of stakeholders per need creates groups that are refining a specific need.

6.3.5. Exploration

Category: game models

Due to the short demonstration, exploration has been implemented in a minimal manner in this prototype. Stakeholders can branch needs that were suggested by others and follow those traces. The origin of a well-developed need (the parent) can be found as well as the outcome of a raw need (the branch).

In case of a longer period of validation, exploration can allow stakeholders to see how needs have evolved in product functionalities and releases.

6.3.6. Endorsements

Category: game design patterns

While votes and comments are not typical game elements, they function as endorsements. These endorsements can confirm or oppose a need, and/or provide a building block to improve a need.

6.4. Pages

Apart from the home-, about-, leaderboards- and contact pages, refine contains three important pages: the needs overview, need details and user profile. On top of each page, the menu bar can be found. Besides the hyperlinks to other pages, the user status bar on the right shows the coins and points of the user.

6.4.1. Needs overview

This page highlights the specific question, allows the users to share and vote on needs and gives an overview of all the needs that have been shared. The question is stated on top of the page and is complemented by a short description.

For every need, the title, content, ideator, time submitted, votes for and against the need and number of comments are shown. Users have the ability to sort the needs by recentness and popularity, which is measured in the votes for the need. Additionally, users can search for needs by providing keywords. This reduces the chance that a duplicate need is submitted.

When the user arrives on the page for the first time, a tutorial can be started to learn about the functionality of all elements. This tutorial highlights the user status bar, the box to share needs and the needs that have been shared.

The screenshot shows the 'Needs' section of the Refine website. At the top, there is a navigation bar with 'Home', 'About', 'Needs', 'Leaderboards', and 'Contact'. The user is logged in as 'Test user' with a profile picture and a 'Sign out' button. The main heading is 'How can Qubus 7 help you in being more efficient?'. Below this, a paragraph explains that Qubus 7 is brand new and the site wants to hear from users. There is a search bar for needs and a 'Sort by: most recent | most popular' option. The needs are listed in a grid:

- Style-less qubus7**: A need for a more browseable interface. By Remy Alidarlo on 30 Dec 2014 | 2 comments. 6 Agree, 0 Disagree.
- Adding "Loading screens"**: A need to see a loading indicator. By Thomas Beekman on 27 Dec 2014 | 4 comments. 12 Agree, 0 Disagree.
- Minimize number mouse clicks needed.**: A need to improve the 'Continue!' button. By Thomas Beekman on 27 Dec 2014 | 5 comments. 8 Agree, 0 Disagree.
- Unclear how to start questionnaire**: A need for clearer instructions. It is unclear how to start a questionnaire after clicking it. This appears to be an... Loading indicator.
- Visualization of question tree**: A need for better hierarchy representation. By Pieter Buijenhuis on 06 Jan 2015 | 7 comments. 6 Agree, 2 Disagree.
- Adding a wizard or tutorial**: A need for a first-time user guide. By Sinned on 16 Jan 2015 | 1 comments. 6 Agree, 0 Disagree.

Figure 6-3: The needs overview of Refine

6.4.2. Need details

For every need, this page shows the related comments, parent, branches and need-specific leaderboard, in addition to the information that was already shown in the needs overview. Users have the possibility to comment on the need as well as branch the need. Similar to the overview page, users can start a tutorial to see how the various elements of the page work.

Do you see this page for the first time? Check out this tutorial! [Start tutorial](#) Close [X]

Unclear how to start questionnaire

It is unclear how to start a questionnaire after clicking it. This appears to be an arrow. This could be improved for example by a text "Start questionnaire!" next to the arrow.

By Tom Slenders, on 20 Jan 2015

Agree (7)
Disagree (0)

Comment

Comment
Cancel

Comments

Do you mean that it is unclear how to enter a questionnaire? In that case I agree.
Door Roel Smits op 21-01-2015 | [Comment](#) | [Useful](#) (1)

Branches

No branches yet.

Best contributors

Name	Points	Badge
Tom Slenders	12	
Roel Smits	3	
Joost Koedijk	1	
Rene	1	
desiree de lege	1	
Chelsea	1	

Figure 6-4: A need details page on *Refine*

6.4.3. User profile

This page shows the public profile of a user. On the top, the name and role of the user is presented. On the left, the amounts of the three types of points and the point total are shown. The three most recent needs and comments are positioned on the right of the user profile. This number was chosen to keep the list clear and organized when a user has shared many needs and comments.

Remco Snijders

User

Ideator		17
Commenter		8
Assessor		16
Total		41

Recent needs

Need title	Date
Fill in a questionnaire with keyboard buttons	30 Dec 2014
Role in Answerset	19 Dec 2014

Recent comments

Need title	Comment	Date
Back and home button missing	Good point! Similar to the 'Return to overview' need by Re...	20 Jan 2015
Navigation	Could you be a little more specific in what you mean with \...	16 Jan 2015
Independent focus on question tree	The reason that I disagree with this need is that it's effe...	16 Jan 2015

Figure 6-5: A user profile on *Refine*

7. Demonstration

7.1. Introduction

As the DSRM of Peffers, Tuunanen, Rothenberger and Chatterjee (2007) prescribes, we have demonstrated the method and prototype. Subsequently, this demonstration will be evaluated in Chapter 0. The demonstration was conducted with a case study on *Qubus 7*, the new release of a B2B Governance Risk and Compliance (GRC) tool, developed by KPMG Technology Advisory. A beta version was released to a select group of customers and users, who were subsequently invited to be involved in the improvement of the software. Figure 7-1 shows a screenshot of the tool. On the left part, a question tree is shown; on the right, individual questions are presented.

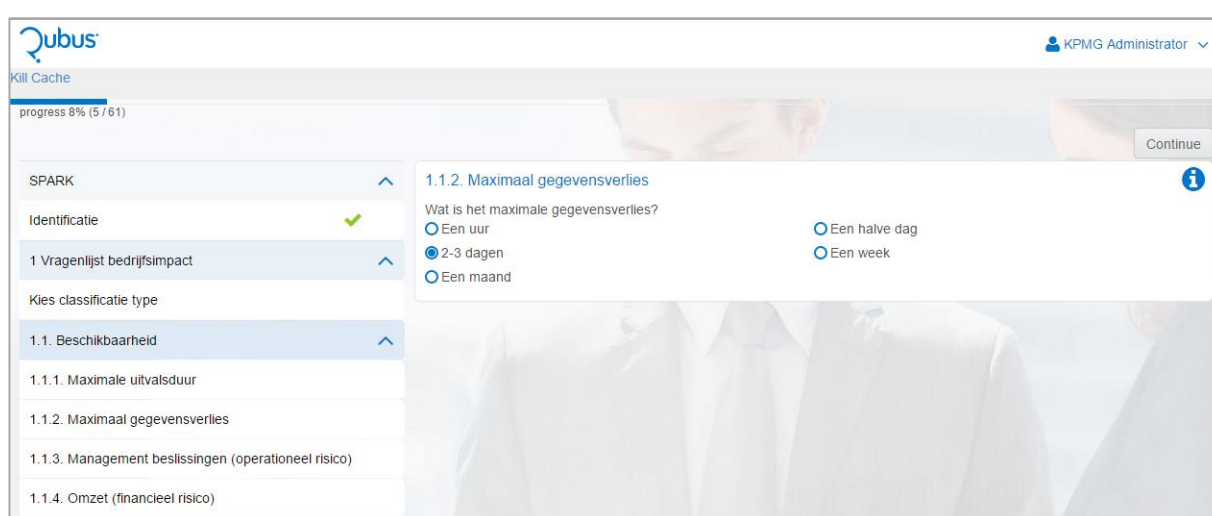


Figure 7-1: A screenshot of Qubus 7

Qubus automates the process of GRC by replacing long lists of elements, that have to be checked and signed for, by a workflow and collaborative environment for questionnaires. Examples of GRC methodologies are Audit Management, Strategic Risk Management and IT Governance, Risk and Compliance. The software is highly configurable, to make it suitable for every GRC process.

Many new features were introduced in the shift from Qubus 5 to Qubus 6. Among other features, the newer version was entirely web-based, had extensive reporting functionality and offered multiple deployment options. Qubus 7 distinguishes itself from the sixth version on three aspects. New technologies were used in the code, the application is platform independent and its responsive design allows for usage on mobile devices.

Users can have the role of *respondent* or *reviewer* in relation to an *answer set*. An answer set is an instantiation of the questionnaire. The respondent provides the answers, the reviewer checks these answers. For the demonstration, each beta user was given access to at least two answer sets. In one, he/she was the respondent, in the other the reviewer.

7.2. Instantiating the method

In this case study, we have instantiated the first seven phases of the CCRE method: the Feasibility analysis, Context analysis, Crowdsourcing preparation, Crowd involvement, Requirements identification, Focus group preparation and Focus group. The last phase of the method, which is aimed at the Development Sprint, was not instantiated due to time constraints and the looser connection with RE. The complete process took twelve weeks, of which five weeks were used for the actual crowdsourcing, the ‘Crowd involvement’ phase.

The following subsections describe the process and outcomes of conducting the activities within the instantiated phases.

7.2.1. Feasibility analysis

To conduct the first five activities of this phase, the product owner of Qubus was interviewed. For each of the activities, one or several questions were asked to determine the value of the related concept. Some concepts, such as the vision and the openness, were also discussed with the bigger SPM team, consisting of the product principal, company supervisor and product owner. The product principal is the authorizing officer for the product and controls the finances. He also determines if and when a new Sprint is started. While we already planned to apply the CCRE method in this case study, the feasibility analysis provided us with the situational factors that were relevant and helped us to tune the method to the situation at hand.

7.2.1.1. Define scope

In this case, the CCRE method focuses on a test environment of Qubus 7. This environment is a beta version of the release and is not free of bugs. Users have the ability to see an overview of answer sets, fill out those answer sets and create a copy of the standard answer set. There was no limitation on a certain feature, role or perspective, the complete front-end of the application was open for feedback.

7.2.1.2. Determine vision

With the new release, more focus has been put on the mobile usability of Qubus. It should be possible for auditors to complete an answer set from their tablet, while they’re at the customer. Vision on the market also exists, Qubus mainly wants to address auditors. The beta version should give customers a taste of Qubus 7 and will serve as a demo of the product in the near future.

At a product level however, there is no roadmap for the product and no specific decisions regarding functionality have been made for the future. We therefore conclude that there is only little vision on the product.

7.2.1.3. Determine degree of openness

The product management team was really open and did not want to jeopardize the usefulness of the method by limiting the openness. Initially, all feedback is welcome. The product owner stated: “The process will teach us which feedback is useful and which is not.”

7.2.1.4. Identify situational factors

Table 7-1 contains the values of the identified situational factors. In the size of the organization, the product owner is considered a developer as well as a SPM team member. Since Qubus 7 is currently in the beta stage and therefore has no customers or users, these amounts are given for the previous version of Qubus. These customers and users potentially make the shift to Qubus 7. Since the values were determined in an interview, most values are estimations instead of exact measurements.

A remark should be made on the size of the organization. The development team is flexible, since there is only one full-time member. The activity of others is based on the amount of necessary work and other

projects. The most striking values of the other SFs are the high customer loyalty, high product age, low bug and requirement rate and low user involvement. The customer loyalty is high because some of Qubus’ clients are also part of the same organization and will probably not choose for a substitute product. This makes the low involvement surprising, since it should be easier to be involved when you are part of the same organization.

Table 7-1: The situational factors for the demonstration

Situational Factor	Value	Situational Factor	Value
Development philosophy	Agile	Number of localizations	2
Size of organization	3FTE developers, 3 SPM team members	Bug rate	1/month
Customer loyalty (1-10)	8.5	New requirements rate	1/month
Customer satisfaction (1-10)	7	Number of products	3
Customer variability	High	Product age	25 years (All versions)
Number of customers	10-12 (Qubus 6)	Product lifetime	5 years (Qubus 7)
Type of customers	Enterprise departments	Product size	Large
Number of users	50-80 (Qubus 6)	Product tolerance	High
Type of users	High education, low IT skills	Product usability	Medium
Frequency of use	Daily – quarterly	Effect of company policy	Low
Market growth	Constant	Customer involvement	Customers – 80% Users - 30%
Market size	50 potential clients	Effect of legislation	Low
Release frequency	3-6 releases/year	Partner involvement (1-10)	3
Sector	Auditing	Degree of customization	Tailored
Standard dominance	Low	System complexity	Users – Low Developers – High
Variability of feature requests	Low	Geographical distribution of customers	Mostly in the Netherlands

7.2.1.5. Determine intended outcome

Since the product owner was open towards all feedback and had no limitations in the scope, there was no clearly defined intended outcome. However, the product owner expected to get feedback on the usability of Qubus and hoped to get input to improve the efficiency and ease of use of Qubus 7. There were already some known weaknesses of the beta version, which were on the planning to be solved, but the outcome of CCRE might provide a new perspective (from users and experts) on those weaknesses and prioritize them.

7.2.1.6. Conclude crowdsourcing potential

Several conclusions are drawn from the previous activities. The scope of the method has little limitations and there exists little vision regarding the future of the product. Additionally, the product team is very open to all sorts of feedback, while hoping for feedback regarding the usability and ease of use of the product. Looking at the situational factors, there is a medium amount of customers and users but low user involvement and average customer satisfaction; there is clearly room for improvement with the available stakeholders. Since the development philosophy is agile and the release frequency is reasonable, results can be implemented

relatively quick. The tailored nature of the product might make it difficult to get consistent feedback for the software. Every client has its own instantiation of the product and therefore has a different view.

To conclude, there is a large potential for crowdsourcing, while there are small risks due to the low customer involvement and high degree of customization.

7.2.2. Context analysis

Stakeholders and existing feedback channels were identified in the interview with the product owner. The decision for a primary platform and the community manager were specific for this case study, since we wanted to closely control and measure the execution of the method.

7.2.2.1. Identify stakeholders

The number of developers, customers and users was already identified during the situational factor identification. In addition to these stakeholders, there is a product principal, that initiated the new release of Qubus and a product owner. The stakeholders were initially grouped as SPM team members, development team members, clients and users. Since the product was developed in a department that housed experts in the field of user experience and software development, these were also identified as stakeholders.

7.2.2.2. Identify existing feedback channels

The SPO was not actively gathering feedback on the software product. However, the largest client once provided a set of requirements for the previous version of Qubus. The list of situational factors already showed that the amount of feature requests and bugs is relatively low. The little feedback that is provided, comes in through four channels:

- Directly during interactions with clients;
- Phone calls;
- Emails;
- During the testing part of the implementation, when a Qubus team member is on-site.

7.2.2.3. Choose a primary interactive platform

The *Refine* prototype has been developed as the interactive platform of this method, to stay in control of all related elements. The platform is described in detail in Chapter 6. Alternative platforms were not seriously considered, but a small assessment (Table 5-4) of the available tools shows that there is no software-specific platform that optimizes requirements traceability by including requirements evolution. Additionally, the gamification elements that were available on those platforms were minimal.

7.2.2.4. Organize secondary channels

The secondary channels have not been organized, since the bug and requirement rate that was observed there was very low. We decided to promote the primary channel as the best route to provide feedback.

7.2.2.5. Appoint community manager

I, as the researcher that conducts this study, was appointed as the community manager. This role involved the overall coordination of the method instantiation, guidance during all activities, contact with the stakeholders and moderation on the platform.

7.2.3. Crowdsourcing Preparation

Since this is the first case study with *Refine* and it is applied to a beta version, the expected motivation of stakeholders to be involved was very low. This phase was therefore essential to create a crowd that was varied, motivated and large. On the other hand, the research character of the platform and method made it easier to let stakeholders know that this was a pilot and thereby let them adhere to the guidelines.

7.2.3.1. Devise a marketing campaign

The stakeholders were informed in three steps. First, we sent the development team, product management and experts from the organization an email with an explanation of the process and their role as the first users of *Refine*. These stakeholders would be asked to share some initial needs, comments and votes on the platform, in order to make it more attractive to use. The second group consisted of existing clients of Qubus. In the email to that group, they would be asked for permission to contact the end-users from their organization. These users were contacted as the third group.

We have not added an external incentive for participation, but motivated the stakeholders by communicating the potential improvements to their work with Qubus 7 and the contribution to scientific research.

7.2.3.2. Invite all stakeholders to the platform

According to the marketing campaign that was described in the previous activity, the stakeholders were invited. All four members of the development team, the three SPM team members and two experts were initially emailed. Unfortunately, it was more difficult to reach and convince the clients than expected, which resulted in a single client, who was responsible for nineteen users.

The scoping question was determined based on the feasibility analysis. The product owner was open towards all kind of feedback, but had some focus on usability and increasing the efficiency of the users' work. We therefore came to the question "How can Qubus 7 help you in being more efficient?". This question was communicated on the platform.

7.2.3.3. Assess crowd characteristics

In total, ten internal and nineteen external stakeholders were invited. While this is not a large crowd, it is a sufficient size for a first case study and allows for easy moderation of the platform. If all stakeholders would be active, the crowd would be varied and suitable with IT experts as well as more novice users, a good distribution of external and internal stakeholders and positions varying from junior developer to product principal. Since everyone was specifically invited, the crowd was not unknown.

7.2.3.4. Solve insufficiency

At this point, no insufficiency was identified. However, the poor reachability of other clients made us skeptical towards the involvement of the invited client and users. We therefore invented a weekly update, which is further explained in the 'Engage crowd' activity.

7.2.3.5. Invent gamification elements

For this specific demonstration, the gamification elements were invented during the development of *Refine*, the prototype. Because of the short duration of this demonstration, the exploration element was not extensive. Suggested needs would not make the complete conversion to an implemented functionality, which made the origin and outcome of needs not fully traceable. A demonstration in which needs are also added to the Sprint Backlog and implemented would allow for a better traceability of needs and therefore improve exploration. The group forming element was enforced in this demonstration, by inviting the stakeholders in three stages. In signing up for *Refine*, the stakeholders could choose for a Team Member, Client, User or Other role.

All other game elements were specific for the prototype of *Refine* and were explained in Section 6.3.

7.2.3.6. Develop guidelines

In addition to the explanation about the platform functionality, gamification elements and the feedback process, several guidelines were developed for the demonstration. The first three guidelines were related to

the case being a pilot and part of this research project. The third and fourth guideline would however be useful in other cases.

- 1 Shortcomings of the platform should be communicated to the team instead of exploited;
- 2 Stakeholders are requested to fill in a questionnaire after the demonstration period;
- 3 Members of the development team should provide 'attractive' feedback, that motivates external stakeholders to participate;
- 4 Stakeholders should indicate whether they want to be involved in potential focus groups.

7.2.3.7. Communicate guidelines

The guidelines were communicated via multiple channels. General guidelines on the process could be found in the invitation to share needs. More detailed information about the platform was shared on an about-page of *Refine*. Members of the development team were additionally informed about the process and their special role (i.e. the last guideline) face-to-face.

7.2.4. Crowd involvement

In the previous phase, the assembled crowd was invited to test Qubus 7 and sign up for *Refine*. Once signed up, the stakeholders could share, comment on and vote for needs. Information about the platform was given in the invitation and on the about page. During the involvement, existing participants were constantly encouraged and new participants were invited.

7.2.4.1. Mine feedback from channels

Coherent with 7.2.2.4, feedback from other channels was not relevant in this demonstration.

7.2.4.2. Moderate input

Luckily, this activity was unnecessary except for one minor case. A participant submitted a wrong need which had to be deleted and for which the coins (see Section 6.3.2) had to be refunded.

7.2.4.3. Suggest need

On the *Refine* platform, participants have the ability to *share* needs. As Figure 6-3 shows, this feature had a prominent place on the needs overview page. To share a need, the description of the need and a title for the need was necessary. The title field is intuitively positioned after the description field. Participants first formulate their need and then summarize it in a title. A shared need could be edited at all times and is labeled as 'Your need'. In total, 21 needs were suggested by the participants. Examples of needs are given in Section 7.2.5.1. More detailed results are outlined in Section 8.1.

7.2.4.4. Vote for needs

Participants could vote for or against needs through an *Agree* and *Disagree* button. The current amount of votes was shown with a green and red bar. 113 agreeing votes and seventeen disagreeing votes were given by the participants.

7.2.4.5. Discuss need

On the need details page, participants could comment on needs and other comments. This allowed the crowd to rationalize their opinion and thereby discuss the need and learn from other stakeholders. In addition, comments could be voted upon. This mechanism allowed the participants to emphasize a useful comment. On the 21 needs, 37 comments were given. The comments varied from a simple "Agree" to a technical analysis of the effects of an auto-forwarding feature. Participants voted on sixteen comments.

7.2.4.6. Tune crowd

In the first week, a low percentage of the invited development team members was active. The largeness of the crowd was therefore insufficient. After inviting the Qubus client and users, these new invitees also remained largely inactive. Besides the weekly update that is described in the next activity, we increased the largeness of the crowd by inviting seven additional participants with an IT background. While they were working for the SPO, they had not used or developed Qubus before. By inviting these participants, the largeness (38 invitees) and suitability were improved, but the variety was decreased. This was a characteristic that we were unfortunately unable to boost.

7.2.4.7. Engage crowd

A weekly update – containing the most popular needs and a leaderboard – was invented and emailed to all invited users (Figure 7-2). This email was sent five times. In addition, the development team members were personally addressed (offline) to encourage their activity. These measures motivated some of the inactive participants to contribute more.

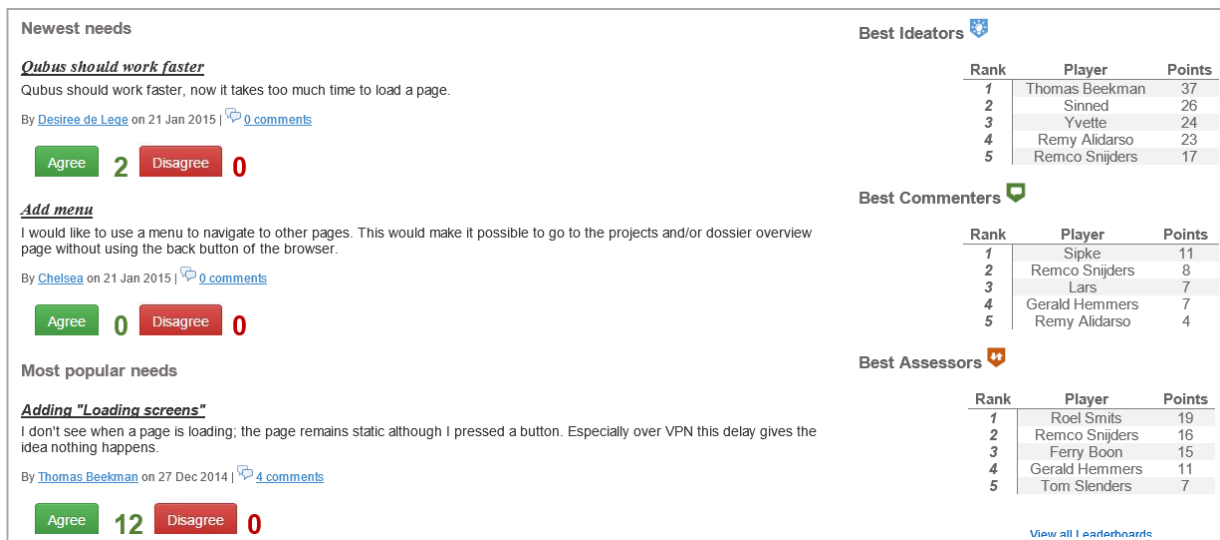


Figure 7-2: A part of a weekly Refine update

7.2.5. Requirements identification

After five weeks of crowd involvement, we sat together with the product principal and the product owner to discuss the needs until that moment. The most popular needs (based on agreeing votes) were discussed in detail, while the remaining list of needs was scanned for potentially interesting needs. In the same meeting, the effects of implementation were determined for the identified requirements and the relative priority was calculated.

7.2.5.1. Identify mainstream requirements

Two needs, displayed in Figure 7-3 and Figure 7-4 were significantly more popular than other needs. The need that suggested loading screens was additionally supported by a need that suggested a loading indicator and comments of a development team member and expert that stated the ease of implementation. The need was specific enough to convert to a concrete mainstream requirement: “a loading screen should be shown when a page is loading”.

The second most popular need, which suggests to minimize the number of mouse clicks, did not provide a direct solution. Part of the need was caused by a known problem of the beta version; the continue button did

not work. Two comments (Figure 7-5) and one branch did provide a solution. One of the development team members suggested auto-forwarding, which was dissuaded by another team member. Another team member suggested putting more comments on one page. The branch suggested control of a questionnaire with keyboard buttons, but the mobile focus of Qubus 7 as well as the complexity of the functionality made it less popular. The requirement that was stated was “users should be able to navigate through an answer set with less mouse clicks”.

Another need (“Visualization of question three”, Figure 7-6) was the fifth most popular need, but described a potential problem of the user interface of the application. In the comments, members of the development team discuss that the implementation might seem easy, but could provide problems when a question tree can have many sublevels. During the meeting, we decided to give this need a chance as a mainstream requirement, to see what its relative priority would be. “The hierarchy of the question tree should be presented more clearly.”

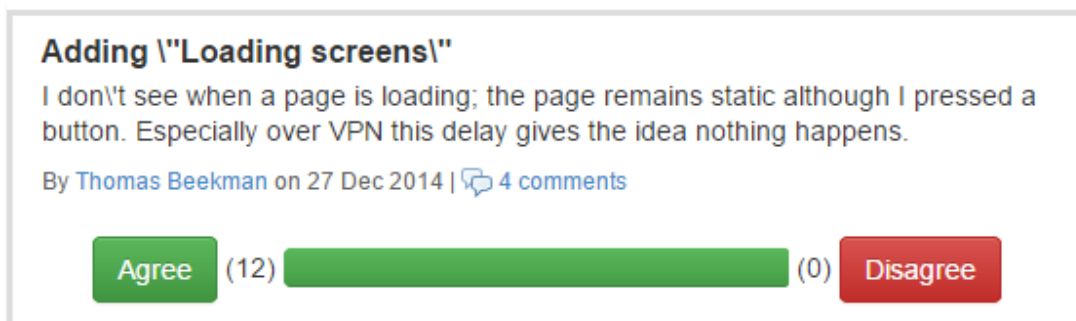


Figure 7-3: The need for loading screens, as shared on *Refine*

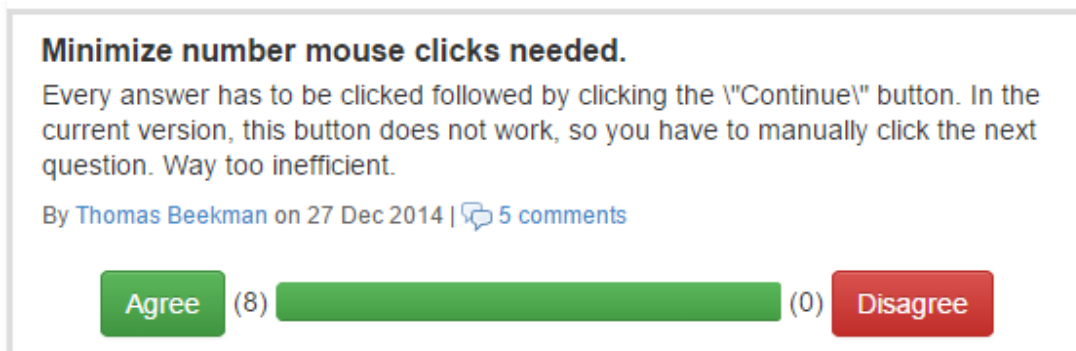


Figure 7-4: The need for a lower number of clicks needed, as shared on *Refine*

Comments

place more questions on 1 page ... this is possible with the configuration , creation of a questionnaire. So support on setting questionairs would solve this issue.
 Door [gerald hemmers](#) op 20-01-2015 | [Comment](#) | [Useful \(0\)](#)

Branched: I made a branch for a possible solution, I would like to get your feedback!
 Door [Remco Snijders](#) op 30-12-2014 | [Comment](#) | [Useful \(0\)](#) | [View branch](#)

I agree, although I think the continue button is fine. Maybe add an option to be auto-forwarded to the next (relevant) question in the answerset (questionnaire)?
 Door [Remy Alidarso](#) op 28-12-2014 | [Comment](#) | [Useful \(0\)](#)

auto-forwarding is too dangerous, on many questiontypes the values can (and will be) saved on change/blur. Having an auto forward in the case of a checkbox question or open question where you click outside of the text box would forward you to the next question even though you didn't answer the question the way you wanted to. In short, the "continue" button should be fixed.
 Door [Ferry Boon](#) op 16-01-2015 | [Comment](#) | [Useful \(1\)](#)

I think that the auto-forwarding option might be dangerous, since people often want to review their answer. Adding it as an option would maybe increase complexity? I think it would be nice to press 'Enter' to go to the next question/page, like in Typeform (www.typeform.com).
 Door [Remco Snijders](#) op 30-12-2014 | [Comment](#) | [Useful \(0\)](#)

Figure 7-5: The comments on "Minimize number mouse clicks needed" on *Refine*

Visualizaton of question tree

The visualisation of the question tree does not represented the hierarchy of the question tree good enough to understand the hierarchy quickly.

By [Pieter Buitenhuis](#) on 06 Jan 2015 | [7 comments](#)

Agree

(6)

(2)

Disagree

Figure 7-6: The fifth most popular need, as shared on *Refine*

7.2.5.2. Identify minority requirements

There were no needs that were supported by a specific small subset of the crowd. This was probably caused by the small size of the crowd. However, the product owner identified one need that was specific, easy to implement and did give a “nice touch” to the application (Figure 7-7). We decided to consider this a minority requirement: “the overview of answer sets should show the role that the current user has in relation to that answer set”.

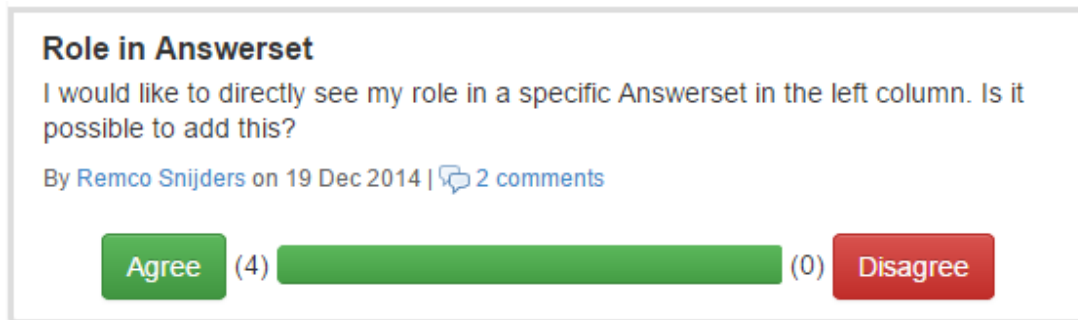


Figure 7-7: A less popular need, as shared on *Refine*

7.2.5.3. Determine effects of implementation

After the identification of four requirements, we were able to relatively scale these requirements – from one to ten – on four aspects: business value, penalty, costs and risk. The results are shown in Table 7-2. The totals, percentages and relative priorities are calculated, the other numbers are assessed. Without going into detail on the discussion for every requirement, the decisions on the requirement with the highest priority are rationalized. Adding loading screens does not add any real value to the application; it can be seen as a ‘necessary evil’. That also emphasizes the penalty that is suffered when it is not implemented. When users constantly have to wait for a loading application, they get annoyed. The costs are relatively low, the right implementation has to be determined and one of the developers has to develop and test it. Finally, there is hardly any risk to this requirement; there are no dependencies on or from other requirements.

7.2.5.4. Calculate relative priority

Based on the template of Wiegers (1999), the relative priority was calculated for each of the four identified requirements (Table 7-2). This prioritization shows that the order of the mainstream requirements remains similar to the order on the platform. However, the minority requirement got a second position, largely due to the low costs and risk of implementation.

Table 7-2: The effects of implementation and relative priority of the four identified requirements. The orange requirement is a minority requirement.

Weight		1	1			1		1		
Requirement		Rel. business value	Rel. penalty	Total benefit	Total benefit %	Rel. costs	Costs %	Rel. risk	Risk %	Rel. priority
#1	Loading screens	1	10	11	30%	4	18%	2	13%	0.304
#2	Role in answerset	2	3	5	14%	3	14%	2	13%	0.205
#3	Minimization mouse clicks	5	5	10	27%	7	32%	5	31%	0.071
#4	Improved visualization of question tree	4	7	11	30%	8	36%	7	44%	0.050
Total		12	25	37	100%	22	100%	16	100%	

7.2.6. Focus group preparation

In this case, we organized one focus group as a 45 minute meeting. We decided to discuss multiple requirements and invited five participants.

7.2.6.1. Select requirement

We decided to select the top 3 requirements from Table 7-2. The first two requirements were not expected to need much discussion. Requirement #3 was related to several suggestions for implementation and probably needed more analysis and discussion before placement on the product backlog.

7.2.6.2. Select involved stakeholders

The four best contributors of the two mainstream requirements (#1 and #3) were invited. Since the contributors from the two requirements were overlapping, five participants were invited. Requirement #2 was refined by only a few participants; additional stakeholders were therefore not selected from that list.

7.2.6.3. Identify additional stakeholders

No additional stakeholders were identified, because the development team, SPM team and experts were all represented by the selected stakeholders and we did not want to make the focus group too large to be effective. The client and users were not willing to participate.

7.2.6.4. Invite relevant stakeholders

All selected stakeholders were employees of the organization and were therefore invited by email as well as face-to-face. One of the invitees could not join, the other four confirmed their presence.

7.2.6.5. Develop design options

For requirement #1, no design options were developed since the comments suggested that a standard solution was already on the Product Backlog. For requirement #2, a screenshot of the application was edited to show a potential implementation of the functionality (Figure 7-8). Finally, a bullet list and one screenshot were developed as design solutions for requirement #3. Two items of the bullet list were based on the comments on *Refine*, one on a branch and another one on an idea of the community manager.

- Auto-forwarding – when a user filled in a question, he is forwarded to the next question;
- Reposition continue-button (screenshot in Figure 7-9) – This does not minimize the number of clicks, but minimizes mouse movement and makes proceeding through the answer set more logical;
- Keyboard control – users can answer questions, go back or proceed by pressing keyboard buttons;
- Support on settings for questionnaires – the Qubus team member that supports the implementation should tell the client to put multiple questions on one page, thereby reducing the number of clicks.

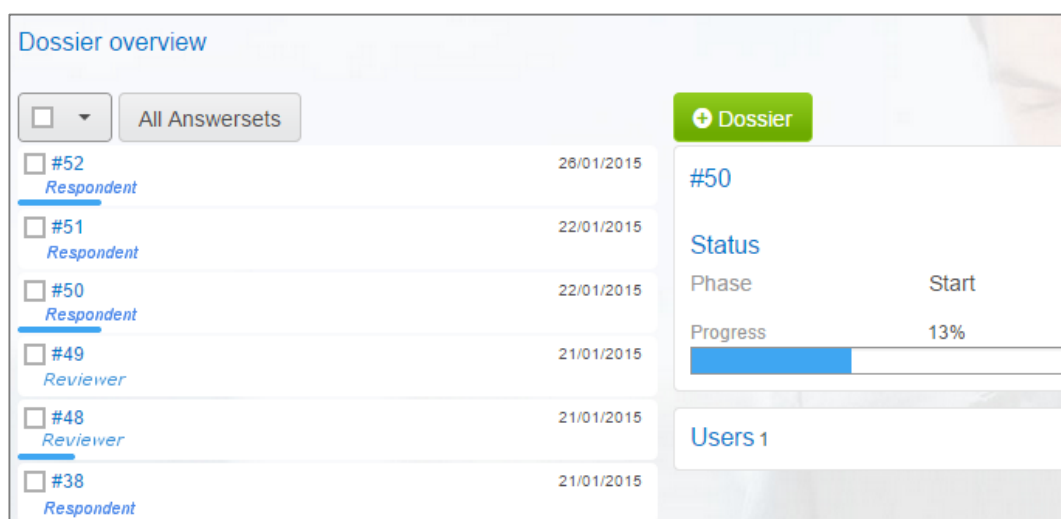
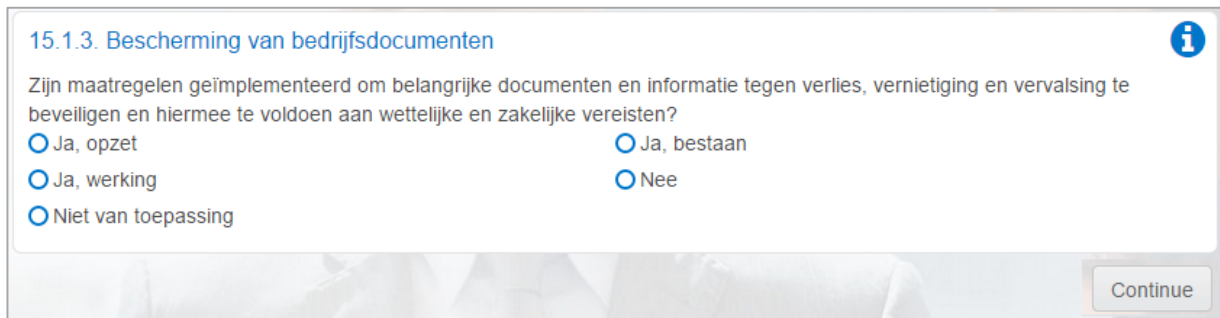


Figure 7-8: The role for every answer set, a design option for requirement #2



15.1.3. Bescherming van bedrijfsdocumenten i

Zijn maatregelen geïmplementeerd om belangrijke documenten en informatie tegen verlies, vernietiging en vervalsing te beveiligen en hiermee te voldoen aan wettelijke en zakelijke vereisten?

Ja, opzet
 Ja, bestaan
 Ja, werking
 Nee
 Niet van toepassing

Continue

Figure 7-9: The repositioned Continue-button, a design option for requirement #3

7.2.6.6. Evaluate design option quality

All requirements and design options were simple features and did not need a preventive security or maintenance evaluation. However, the auto-forwarding option for requirement #3 might collide with standards and confuse users. This assumption should be confirmed or denied by the focus group and the design option was thus not eliminated. Keyboard control would be inconsistent with the mobile focus of Qubus 7 and was therefore rephrased to keyboard shortcuts.

7.2.7. Focus group

Two development team members, the product owner and an expert joined the community manager in the focus group. In 30 minutes, the requirements and design options were discussed. The community manager ended the meeting with a summary of the decisions.

7.2.7.1. Present design options

The design options were presented and discussed in the order of priority. This was also done because requirement #1 was the most concrete, followed by requirement #2 and #3. The meeting was in an informal setting and the members of the focus group were given a handout with the design options and the need titles, descriptions and comments. In the presentation, the design options were related to these descriptions and comments.

7.2.7.2. Discuss design options

Each of the requirements was separately discussed, taking the design options into account. As the comments suggested, the relevance of requirement #1 was already known, but its presence on *Refine* emphasized its urgency.

The design option for requirement #2 was different from what the product manager had in mind and triggered the discussion whether showing the role would be relevant at all. Users of Qubus would almost never have different roles in different answer sets, which makes it unnecessary to display their roles in the menu that should provide a clear overview.

Requirement #3 triggered a more extensive discussion in which several design options were considered and compared. Auto-forwarding was already discussed on *Refine* and the same arguments were used in the focus group. For several types of questions (e.g. radio buttons), the auto-forwarding option might provide a nice solution, but disables the user from double-checking their answer. For other types of questions (e.g. multiple checkbox), the option even disables the user from finishing their answer. The suggestion of repositioning the continue-button was immediately accepted. Keyboard shortcuts were considered complex to implement and not wanted by the members of the focus group. The fourth design option was suggested on *Refine* by one of the focus group members. The other members accepted it as an appropriate solution. Explaining the clients that they should put multiple questions on one page significantly decreases the amount of clicks.

7.2.7.3. Document decisions and rationale

The description in the previous activity is the (summarized) documentation of the decisions and rationale. During the meeting, notes were taken to serve as input for this description.

7.2.7.4. Place requirement on product backlog

Requirement #1 was confirmed in the focus group and directly placed on the product backlog, requirement #2 was omitted during the focus group and requirement #3 led to two requirements for the product backlog. The outcome of the focus group was a set of three requirements:

- 1 Add a loading screen
- 2 Reposition the continue-button
- 3 Inform clients about the usability of multiple questions on one page

Due to some uncertainty when the next Development Sprint is starting, the requirements were not yet positioned on the backlog.

8. Evaluation

The case study on Qubus 7, in which the CCRE method was instantiated, led to a large set of results. While the low amount of participants does not allow us to draw statistically significant conclusions, the combination of results identifies strengths and weaknesses of the method and assesses the potential for improved requirements engineering. The case study was evaluated in four ways:

- Observation of the demonstration and its results;
- A questionnaire for participants;
- An interview with the product management of Qubus;
- Expert interviews in the product software industry.

Each of the following subsections describes the results of one of the evaluation techniques. Section 8.5 aggregates the findings into one comprehensible set of results.

8.1. Observation

The observation of the seven instantiated phases has been described in the previous chapter. This section focuses on the observation of the prototype.

The crowd, which consists of the people that signed up for *Refine*, consisted of nineteen of the 37 invited stakeholders (51,4%). During the phase of crowd involvement, 21 needs and 37 comments were shared, two branches created and 130 votes given. The participants collected 373 points; 191 ideatorpoints, 53 commenterpoints and 129 assessorpoints. As Figure 8-1 shows, the amount of points is very unevenly distributed; six of the participants (31,6%) earned more than 50% of the points. 68,4% earned 90% of the points.

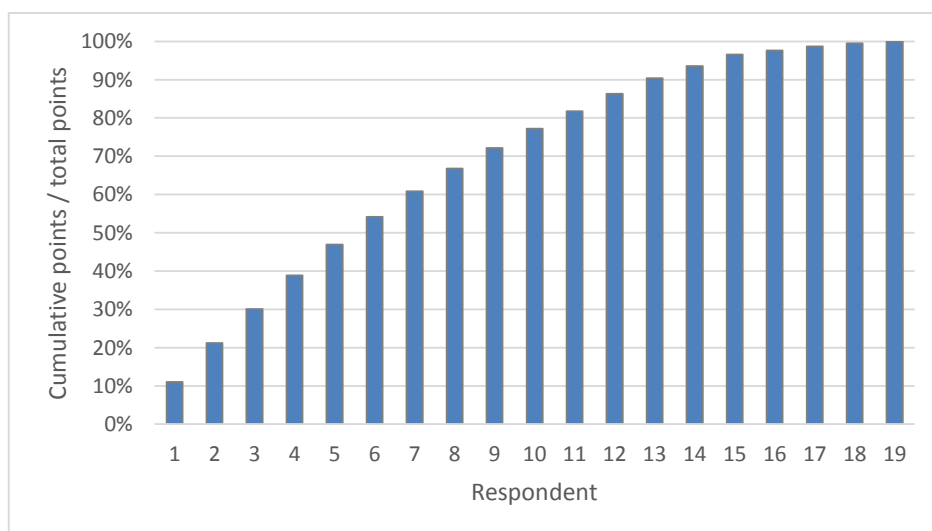


Figure 8-1: The distribution of points among the participants (ordered by earned points, descending)

Table 8-1 displays the activity of the different types of stakeholders on *Refine*. Most of the input came from the development team and 'others' (Qubus-unrelated employees of the SPO), largely because these were two of the largest groups. The community manager was the most active stakeholder, since he had the role to keep the other stakeholders motivated and trigger discussions. Looking at the other stakeholder type averages:

- ‘Others’ shared the most needs, which might be explained by their unfamiliarity with the product;
- Development team members left the most comments, which can be explained by the fact that they have the most knowledge on the technical feasibility and opportunities of requirements;
- The development team members voted the most, followed by the product management. This can be explained by their insight in the possibilities of the application;
- The most points were earned by development team members, which follows from the previous two statements, but also suggests that they gave the most valuable input.

Table 8-1: The activity of the different stakeholder groups on *Refine*. *N* is the number of active participants. *Active* is the percentage of invited stakeholders that was active. *Avg.* is the average number per individual stakeholder, *Total* is the total number for the whole group.

Stakeholder type	N	Active	Needs		Comments		Votes		Points	
			Avg.	Total	Avg.	Total	Avg.	Total	Avg.	Total
Community manager	1	-	2	2	8	8	21	21	41	41
Product management	2	67%	0	0	3	6	6	12	13.5	27
Development team	4	100%	0.8	3	3.8	15	14.8	59	26.8	107
Experts	4	100%	1	4	1.3	5	2.5	10	16	64
Client	1	100%	0	0	0	0	3	3	3	3
Users	1	6%	1	1	0	0	3	3	11	11
Other	6	86%	1.8	11	0.5	3	4.2	25	20	120
Total	19	50%	1.1	21	1.9	37	6.8	130	19.6	373

8.2. Participant questionnaire

Seventeen participants filled in the questionnaire. Two participants did not; the community manager conducted the study and the user unfortunately did not respond. The questionnaire consisted of seven topics:

1. Introduction and IT background
2. Previous experience with software feedback
3. Difficulty, perceived usefulness and motivation in regard to the CCRE process
4. Goals, difficulty and motivation on *Refine*
5. Effect of game elements on *Refine*
6. Usefulness of functionality of *Refine*
7. Further remarks

The following subsections shortly outline the results for each of these topics. Since the amount of participants does not allow for the scientific explanation of differences between groups, we will limit the results to means and qualitative observations.

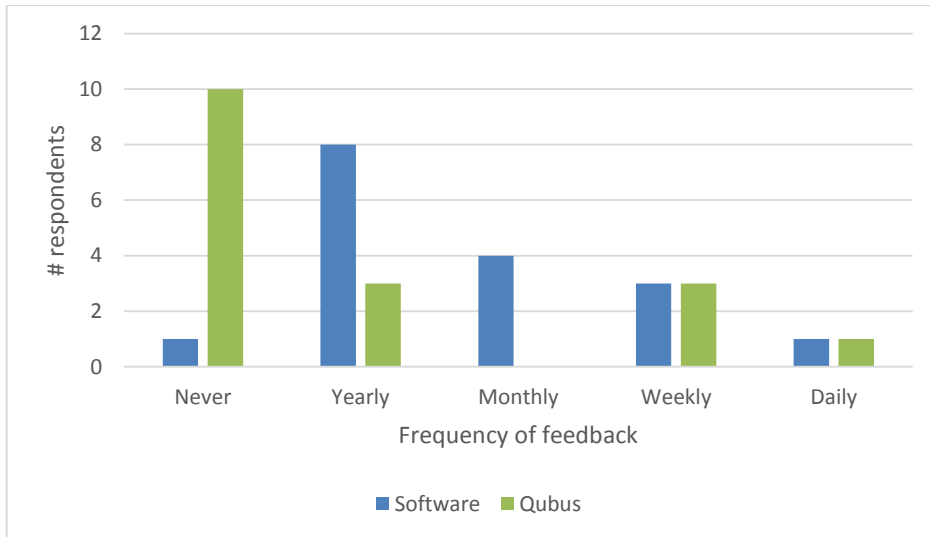
8.2.1. Introduction and background

All respondents submitted their email address to allow a connection with the results of the platform. Eight respondents had used a previous version of Qubus before, while nine had never used the product. All respondents had a study or job related to information technology.

8.2.2. Previous experience with software feedback

Table 8-2 shows that the respondents had little experience with giving feedback to software, and to Qubus in specific. Only seven respondents had ever given feedback to Qubus, while sixteen had ever done this to software in general.

Table 8-2: frequency that the respondents gave feedback to Qubus or software in general



The sixteen respondents that had experience with giving feedback were asked to compare the experience of giving feedback to Qubus with their previous experiences. They could indicate whether the current experience was less (1) or more (5) difficult, useful and engaging, on a Likert scale from 1 to 5. Difficulty scored a **2.69** (as difficult), usefulness a **4.19** (more useful) and engagement a **3.88** (more engaging). The frequencies of the scores are shown in Figure 8-2.

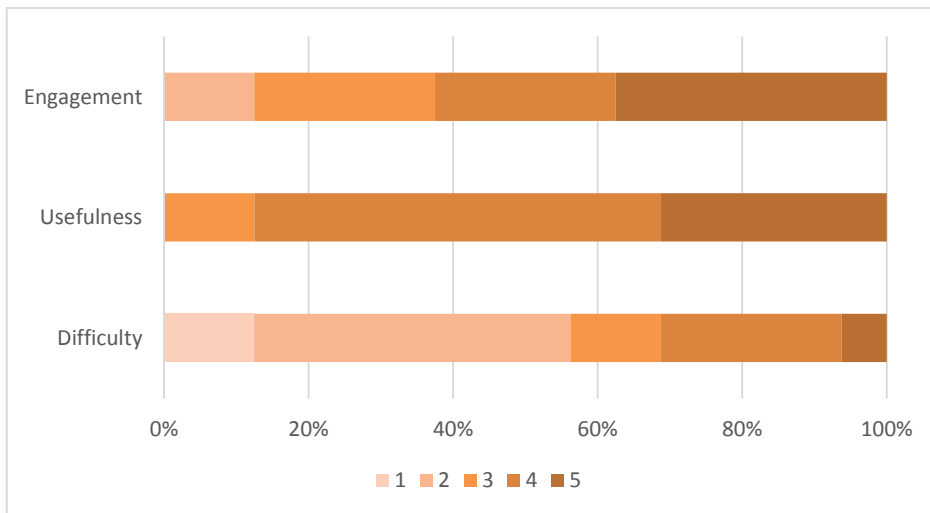


Figure 8-2: Frequencies of the comparison of feedback experiences

8.2.3. The CCRE process

On a similar scale as in the previous topic, respondents were asked:

- whether the process was difficult (very easy – very difficult);

- whether they thought their input would be taken into account (very unlikely – very likely);
- whether they thought their priorities were clear (very clear – very unclear);
- how motivated they were to provide feedback (not motivated at all – very motivated).

The frequencies of the answers are shown in Figure 8-3. On average:

- the process was easy (**1.94**);
- they thought it was likely that their input would be taken into account (**4.12**);
- the perception of the clarity of their priorities was neutral (**2.82**);
- they were somewhat motivated to provide feedback (**3.47**).

The respondents that were not really motivated gave several reasons: “I’m used to giving feedback personally, which is more effective, but also more involving for the developers”, “The test environment of Qubus seemed to artificial”, “Due to no experience with Qubus and the unclarity of the purpose, I was not able to give very useful feedback”, “after you’ve put your needs in the system, there isn’t really any attracting factor which makes you want to check back on your idea”. The respondents that found the process motivating pointed to the gamification elements and the overview of everyone’s needs as the motivators.

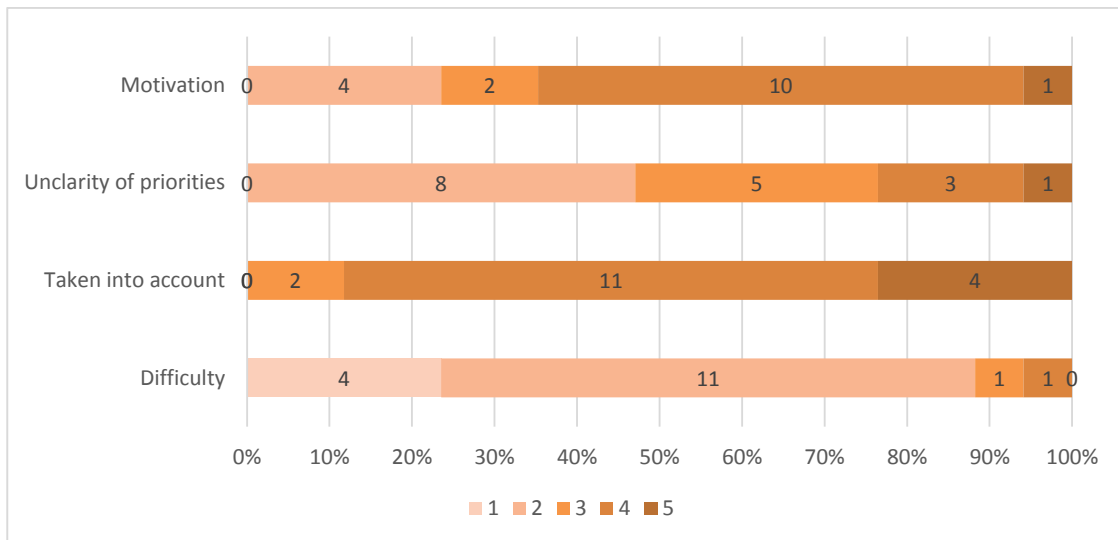


Figure 8-3: Frequencies of the responses on the CCRE process

8.2.4. Refine

Eight potential goals during usage of the platform were presented as check boxes. The frequency of the respondents’ answers is indicated after the option:

- Read the needs of other participants (16);
- Provide suggestions for Qubus 7 (13);
- Provide complaints on Qubus 7 (2);
- Get the most ideator points (2);
- Get the most assessor points (2);
- Get the most commenter points (1);
- Get the most overall points (1);
- Ask questions about Qubus 7 (0).

The respondents could indicate whether they (strongly) disagreed or agreed with the following three statements: “The platform was difficult to use” (1.65, disagree), “I want to use the platform more often” (3.76, agree) and “The weekly update motivated me to be active on *Refine*” (3.24, neutral). Figure 8-4 shows the distribution of answers. Additionally, respondents were asked for the experienced obstacles in using *Refine*. Five obstacles were mentioned, of which two were mentioned twice:

- The risk of duplicate entries (2);
- Not clear how coins were spent (2);
- No personal attachment;
- Running out of coins would stop people from sharing, while this might not be the case in a big group;
- No clear sign in location.

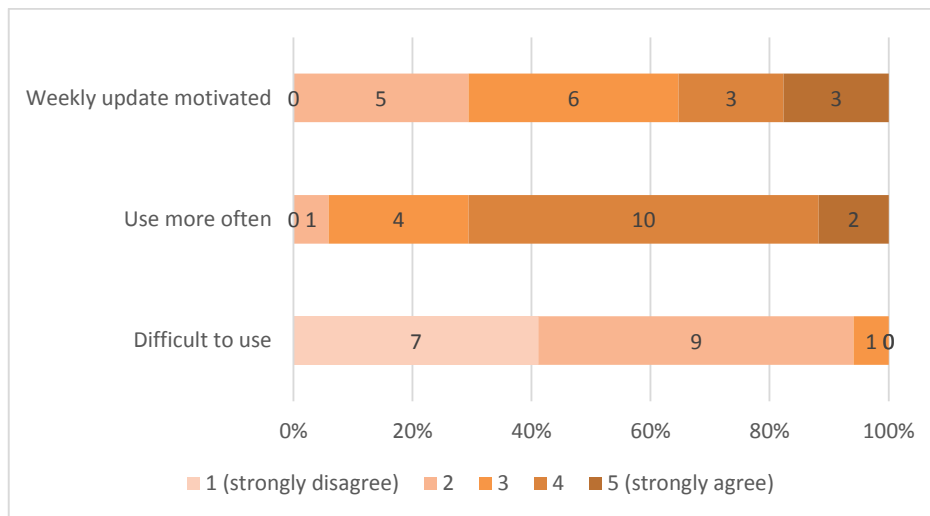


Figure 8-4: Frequencies of responses on the *Refine* platform

8.2.5. Game elements

The respondents were asked how often they looked at their points and leaderboards during the usage of *Refine*, on a scale from 1 (never) to 5 (always). The scores were average with a 2.29 for points and 2.47 for leaderboards. Subsequently, three statements were proposed: “I felt no degree of competition” (3.18, neutral), “The game elements made the experience more pleasant” (3.59, slightly agree) and “I was restricted by the limited amount of coins I received” (2.35, slightly disagree). While these are all relatively neutral averages, the frequencies in Figure 8-5 give an insight into the differences. Some respondents (2) were very restricted by coins, while others (6) were not at all restricted, leading to a neutral average. The answers on the degree of competition covered all possibilities, also leading to a neutral average. The answers on whether the game elements made the experience more pleasant were mostly 3 and 4, already neutral.

In further remarks on game elements, two respondents stated that the coins were limiting them while another considered it an “interesting element in RE by adding scarcity of change resources”. Comparing this with the average agreement with the statement about coins, we can assume that there were only several participants that were restricted. One respondent suggested that development team members should get a multiplier when commenting or voting.

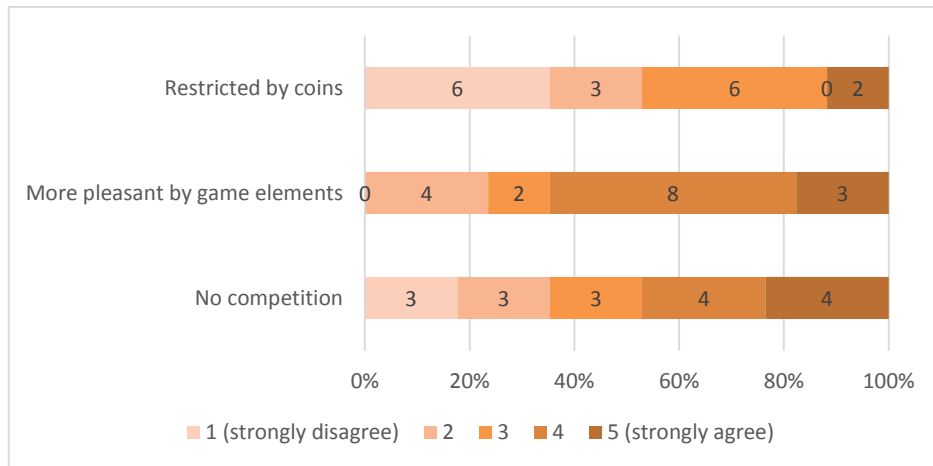


Figure 8-5: Frequencies of responses on the game elements

8.2.6. Functionality of *Refine*

Five respondents (29.4%) used the search bar. For branching, commenting and voting, users were asked to rate the usefulness on a scale of 1 (not useful at all) to 5 (very useful). Branching scored a **2.94** (neutral), commenting a **4.41** (useful) and voting a **4.71** (very useful). The frequencies are shown in Figure 8-6.

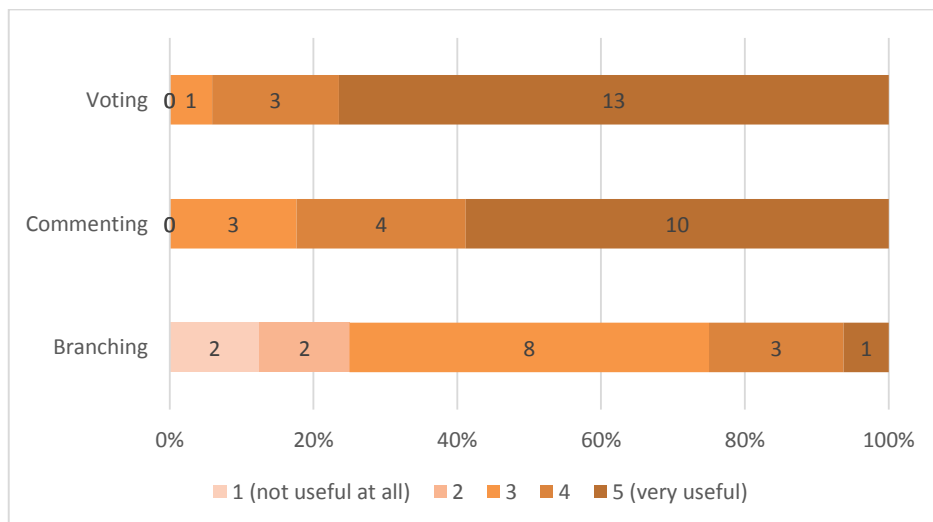


Figure 8-6: Frequencies of the responses on the usefulness of *Refine's* functionality

The next question assessed whether the respondents learned something from the interaction. Someone learned names for visualization techniques, while someone else got a broader view from the different perspectives of users. One respondent learned that “some users have different needs, or reasons to have the same need”, while another learned that “people have the same need”. The ability to see how ideas are refined and what other people experience was also mentioned as something insightful.

In response to the request for further remarks on functionality, one respondent suggested to donate coins to others. The respondent took the role of assessor and therefore wanted to give coins to people in other roles.

8.2.7. Further remarks

Four respondents indicated that there was a market for this initiative or that the prototype looked good. One respondent missed the functionality to delete votes and comments.

8.3. Product management interview

In the same meeting in which the ‘Requirements identification’ took place, the product principal and product owner were asked for the perceived advantages and disadvantages of CCRE compared to alternative methods, as well as the usefulness of the set of resulting requirements. A short unstructured interview of about twenty minutes was held. The questions can be found in Appendix B. As mentioned earlier, no other requirements engineering methods have ever been structurally applied by the SPO. The alternative methods that were considered in this interview were the sporadic questions about features to and from customers.

A summary with the results of this interview is given in the following subsections.

8.3.1. Advantages

- Approachable – There is a low barrier for users and customers to participate in the process. The external stakeholders should be rewarded for their input;
- Structured – The collection of requirements is continuous and the focus is on one software product. Conversations with individual customers are less focused;
- Improved user adoption – Customers and users have been prepared for the new product release and had the opportunity to provide their feedback. This will probably help during and after the product launch;

The interviewees noted that the crowd has done the work of suggestion and discussion, but the structuring and control of the method adds work. The net amount of work is therefore similar.

8.3.2. Disadvantages

- Little incentive to return – Stakeholders should have gotten notifications of likes and comments on their content. In this version, there was little reason to return after single usage of *Refine*;
- No need life cycle on *Refine* – There should be a mechanism to remove implemented requirements from the platform and notify the contributors;
- Risk of novice participants – Having too many non-expert users could increase the amount of trivial or even useless needs. The comments could however clean up these needs.

8.3.3. Usefulness of the outcome

- The amount of resulting requirements is sufficient for the number of involved stakeholders. There is much feedback that is not at the top of the list, which is good;
- The needs were largely trivial. There were only a few new insights gathered, probably because the product was unfinished. Needs were however not useless;
- The most important needs are detailed enough to use in a focus group. There is of course a large variety of detail in the needs and discussions, but the focus group can further refine this;
- *Refine* should be linked via a support button on Qubus, to make it easier for users to share their immediate needs.

8.4. Expert evaluation

Three external experts were interviewed to determine the usefulness of the CCRE method outside the case. These experts did not participate in the expert interviews that were described in Chapter 4. Screenshots of the prototype and a detailed need (Figure 7-4 and Figure 7-5) were shown and explained, after which the simplified method (Figure 5-1) was outlined. The experts could ask clarifying questions throughout the explanation. Twelve statements, to which they could agree on a scale from 1 (completely disagree) to 7 (completely agree), were subsequently given to the stakeholders. This was followed up by four open

questions and the opportunity for the experts to ask any remaining question or make suggestions. The list of statements and questions can be found in Appendix C. Since two statements were negatively phrased, the arrows behind the statements indicate whether a high (arrow up) or low (arrow down) rating was positive. The following subsections describe the results for each of the experts. Section 8.4.4 aggregates the ratings to give a general impression.

8.4.1. Expert #1

This expert works as a consultant at a provider of various IT-related services including *Software Solutions*, which are largely tailor-made for specific customers. The company has 5,000 employees.

During the explanation of the study, the expert noted that the B2B perspective of the company decreases the urgency to directly interact with users. When the prototype was explained, the merging functionality was suggested as an improvement. Another question focused on the moment that the crowdsourcing company would take the proposed needs to the next release. This was something the expert struggled with in his own SPM process. During the explanation of the simplified CCRE method, the expert was surprised that the initial prioritization was only done based on user value. During 'Requirements identification', product management would have to make a decision without the complexity estimation of developers. The involvement of developers in voting and commenting does improve this situation.

Subsequently, the statements were proposed. Table 8-3 lists the ratings and comments that were given in response to the twelve statements. The arrows are repeated to indicate whether a high or low rating was positive.

Table 8-3: The response on the statements in the first expert evaluation

Statement	Rating	Comments
1 ↑	7	-
2 ↑	6	-
3 ↑	2	Prioritization should happen internally, since there is more than user value to be determined. Developers and managers should make an estimation in costs and business value. Most organizations are not transparent enough to do this via crowdsourcing.
4 ↑	6	-
5 ↑	6	This is more difficult however. If you use a platform for this, the traceability of features to the original requirements should work perfectly.
6 ↑	7	It is important that it involves real gamification, including roles. Just giving points (for one type of input) is not gamification.
7 ↑	7	There are multiple effective ways to stimulate innovation. You should involve non-users to get real innovation. Children are a good example that show the value of a novel perspective.
8 ↑	6	Especially when we can get developers on the level of interacting with users.
9 ↓	2	The requirements that trigger a discussion and get points are by definition valuable requirements.
10 ↓	2	When stakeholders work in one requirements database, the quality of requirements is higher by definition.
11 ↑	7	Only when the developers are part of the discussion. Users don't know when something is a user story or a non-functional requirement.
12 ↑	7	-

The open questions focused on advantages, disadvantages and specific contexts in which the method would or would not be useful. The identified advantages were transparency, stakeholder involvement and traceability. The identified disadvantages were more detailed:

- Web as the channel – Sometimes you need to physically visit a customer and visualize ideas;
- The risk of an ‘incomplete’ crowd – What do you do when a specific customer does not give input? And how do you guarantee that the sample is representative? The tool is valuable, but it needs to be fitted to the customer and user situation;
- Transparency could be an obstacle – Everyone needs to be willing to publicly express their opinion.

As a suggestion, the expert would categorize the votes: business value from the financially responsible manager, user value from the users and complexity from the developers. This is however difficult to transparently communicate on the platform, not many companies would be willing to do this.

A context in which the method would work involves a transparent culture with open discussions and clear feedback to the stakeholders. Applications that already have a crowd (e.g. crowdfunding platforms) would be ideal. A closed culture with mainly one-on-one interaction would not allow for this method. Characteristics such as company size or product age are not determinants.

8.4.2. Expert #2

This expert works as a Product Owner at product software organization that focuses on credit risk management. The company is a subsidiary of a 300-employee IT service provider.

Before the statements, the expert already stated some questions that were triggered by the case description. How can the process remain fun, how is it scalable and how can it be made continuous? He mentioned that his organization uses questionnaires for its customers to find out what they want to have improved. The responses to statements are shown in Table 8-4.

Table 8-4: The response on the statements in the second expert evaluation

Statement	Rating	Comments
1 ↑	6	-
2 ↑	5	The discussion between a large number of people with varying expertise needs to be moderated. A small group would make the discussion easier.
3 ↑	6	-
4 ↑	5	Specification does not only entail the need, but more importantly the reason behind a need. The ‘why’ question is more important than the ‘what’ question.
5 ↑	5	For validation, you need a selection of people that have used your product before.
6 ↑	7	We don’t use it ourselves yet, but are eager to start applying it. Offline versions like ‘product boxing’, in which customers are asked to design the product box, are really interesting.
7 ↑	6	Engaging users is easier, you really need to apply the right method for this.
8 ↑	5	CCRE is a part, you need to complement it with contextual information from your system, e.g. the moment that users exit a process, the time spent on a page.
9 ↓	2	You always get useful requirements. Anything that a customer says is useful.
10 ↓	4	As an SPO, you also need to devise innovative solutions yourself. I’m not sure about the comparison with the requirements that result from our customer involvement.
11 ↑	3	Looking at the ‘mouse clicks’ example, there is still discussion on the right solution. You need to split the discussion in statements and rationale.
12 ↑	2	The example is not a requirement yet, just an idea. Somebody needs to research it and convert it to a user story. Maybe there are more possible solutions.

The advantages that were mentioned by the experts are:

- Customer feedback;
- Openness in rationale – you can say why you do or don't take a need into account. The choices need to be made anyway, so it is good to communicate this. When costs of implementing a requirements are high, you can ask customers whether it is worth a higher product price;
- Clear prioritization – You know what customers find important;
- Relation of a need to a customer – Some needs might be specific for a single group of customers.

However, some disadvantages were also perceived:

- Creation of expectations – You cannot say that you are planning to implement a requirement and not implement it;
- It costs time;
- Difficult to discuss – Developers, business people and users all speak a different language;
- Non-representative sample – There is always a small active percentage within a community. The question is whether this percentage is representative for the active and important users of your software.

In response to the question on the context in which CCRE would or would not be useful, the expert said that the method can be applicable in every situation. A smaller group of people might lead to a better discussion.

In addition to these notions, the expert argued for more specific scoping and questioning. Instead of focusing on the whole software product with all involved stakeholders, it would be better to focus on a set of features and discuss this with a select group of stakeholders. In the organization of the expert, customers are addressed group by group. The hypotheses that are created by one group are validated in the other groups. When groups disagree and have conflicting need, they get a customized shell around the standardized product.

The expert had his doubts whether developers should be immediately involved in the discussion and are able to discuss on the same level as users. On a different note, "The most important thing is to give feedback to the participants". As an ideal consequence, the users that suggested an implemented feature can explain that new feature to new users.

8.4.3. Expert #3

This expert works as a Chief Technology Officer (CTO) for a software development and implementation company that focuses on charitable organizations. The company employs eighteen people, of which six are developers. The company still has to involve their customers for the first time. In the session that is planned for this, the CTO will state known gaps and validate some hypotheses with the customers.

During the explanation of the method and prototype, the expert had few comments. The responses on the twelve statements are listed in Table 8-5.

The expert identified the following advantages:

- More appreciation by customers – When you do this well as an organization, your customers will appreciate both your organization and your product more. Customers are more content when you fail something first and correct it, than when you do something right at the first try. This method triggers that effect;
- Insight into the community's priorities – The method gives a validation of whether the things you consider important, are really important in your community;

- Identifying advocates – You see who has a connection with your product. Whether they’re positive or negative, it shows that they take the time to improve your product.

Table 8-5: The response on the statements in the third expert evaluation

Statement	Rating	Comments
1 ↑	7	-
2 ↑	6	This closely relates to requirements analysis, a crowdsourcing tool is useful for that.
3 ↑	5	Internally, it is very useful. Externally, it is a big risk, since an expectation is created when a need is popular in the crowd. You might not live up to that expectation.
4 ↑	7	When something is not clear, the community can respond.
5 ↑	2	I would not involve the crowd in validation, but a small selected group.
6 ↑	5	It works very well in forums, since gamification improves the drive to respond. I’m not sure whether this works on a requirements platform. It is probably better than nothing.
7 ↑	2	“If you let your user community drive your innovation, you would have a Ferrari behind a horse”. You need other groups (e.g. universities, other markets) to be innovative.
8 ↑	6	Yes, because there is no structural involvement at the company yet and using Sharepoint is not optimal.
9 ↓	1	Looking at the list as market requirements, it is definitely valuable.
10 ↓	3	By the possibility to give comments, the output would become a little better. Most customer would however just say that they agree.
11 ↑	4	I would not directly use this in a focus group, I would first extensively discuss it with the product management and think about the actual requirements.
12 ↑	4	Similar to the last statement, a step with the product management needs to occur in between the crowd involvement and the placement on the backlog.

Some disadvantages were also perceived, of which the first two were told to be the most important:

- Creation of expectations – When something is popular in your crowd, you give them the feeling that you will implement an idea. If you don’t do this, you disappoint the crowd;
- Time – To successfully conduct this method, you have to commit a significant amount of time (and experience) in managing the method;
- Overload of complaints – When there are many suggestions for improvement, prospects might think that your product is rather immature;
- Public to competitors – If the platform is open, competitors can use the information to improve their products.

According to the expert, the method would only work in a context where there is enough capacity to manage the tool and solve issues. There should be enough developers to work on new incoming needs in addition to the everyday work. Every type of product and company would benefit from the method, since there are always users that want to “drop” their ideas.

8.4.4. Combined ratings

In order to provide an overview of all responses to the statements in the expert evaluation, Table 8-6 shows all the ratings. The ratings are colored based on their positivity about CCRE; green is very positive, red is very negative and yellow is neutral. For the two negatively phrased statements (9 and 10), low numbers are positive. The colors show that the experts agree that crowdsourcing is useful for elicitation (1), negotiation

(2) and specification (4) and that CCRE improves the quality of the RE process (8) and gives a list of useful requirements (9). However, the quality of requirements is not much better than the quality of the expert's methods (10) and the requirements are maybe not detailed enough for a focus group (11) or Product Backlog (12). In their rationalization however, the experts argued that a meeting with the product management could improve this detail.

Table 8-6: The colored and combined responses to the statements in the expert evaluation

Statement	#1	#2	#3
1. Useful for elicitation	7	6	7
2. Useful for negotiation	6	5	6
3. Useful for prioritization	2	6	5
4. Useful for specification	6	5	7
5. Useful for validation	6	5	2
6. Effective for engagement	7	7	5
7. Effective for innovation	7	6	2
8. Improve the process quality	6	5	6
9. No useful requirements	2	2	1
10. Lower quality requirements	2	4	3
11. Detailed enough for focus group	7	3	4
12. Detailed enough for Product Backlog	7	2	4

8.5. Aggregated findings

The results from the previous sections can be placed into eight overarching findings. While the some findings show the positive potential of the method, others address risks, weaknesses and constraints:

Positive

- 1 The resulting requirements are useful
- 2 The process is perceived as useful from both a management and general stakeholder perspective
- 3 The stakeholders are engaged in the process
- 4 The interaction among stakeholders is valuable

Room for improvement

- 5 The functionality of *Refine* is useful, but can be improved by adding additional features
- 6 The gamification aspect of CCRE is unobtrusive and somewhat effective, but should be improved

Risks and constraints

- 7 Scaling CCRE brings some risks
- 8 CCRE might create, but not fulfill expectations
- 9 CCRE requires an open and transparent context

Table 8-7 maps the findings on the evaluation of the previous sections. The following subsections explain the findings in more detail.

Table 8-7: The mapping of the main findings on the evaluation

	Observation	Questionnaire	SPM team interview	Expert evaluation
Useful requirements	X		X	X
Useful process		X	X	X
Engaged stakeholders	X	X		
Valuable interaction	X	X		X
Useful functionality	X	X		
Improvable gamification	X	X	X	
Risks of scale			X	X
Expectations				X
Required transparent context				X

8.5.1. Useful requirements

The first case study resulted in 21 ordered and largely discussed needs. The instantiation of the method showed that the needs enable the smooth conduction of the subsequent phases of the CCRE method. The product management team confirmed the usefulness of the resulting needs. The experts only found the needs detailed enough for use in a focus group and placement on a Product Backlog when they would be further refined by the product management team. Since this happens during the ‘Requirements identification’ phase, we consider this ‘requirement’ covered. Sufficient experience in the SPM team is needed to do this identification properly.

8.5.2. Useful process

The participants found the feedback experience as difficult as previous experiences, but perceived it as more useful and thought their needs would be taken into account. This can probably largely be addressed to the crowdsourcing preparation, in which the value of user input for Qubus 7 was emphasized. The SPM team found the process structured and approachable due to its continuous and inviting character. The high ratings of experts on the statements about the applicability of crowdsourcing confirm the usefulness of this process.

8.5.3. Engaged stakeholders

Judging from the amount of needs, comments and votes, we can derive that stakeholders were engaged. Additionally, the stakeholders stated that this experience was more engaging than previous experiences.

8.5.4. Valuable interaction

Pacheco and Garcia (2012) showed that best practices in requirements elicitation have different stakeholder roles and constructive interactions. On *Refine*, a lot of interaction between the various types of stakeholders was observed. The questionnaire results show that the participants tried to get an improved insight into the needs of others. While this might be partly addressable to the large amount of internal stakeholders of Qubus, it shows the perceived value of interaction. Additionally, the stakeholders mentioned that they learned from the different perspectives on the platform and found it interesting to see the refinement of needs. To effectively conduct CCRE, the SPM team should know how to find the ‘why’ behind stakeholder needs, as suggested by Yu and Mylopoulos (1994).

8.5.5. Useful functionality

Looking at the amount of votes, it can be concluded that participants found it very easy to communicate their priority of needs. When asked about the usefulness of functionality, they considered commenting and voting very useful, while branching was considered less useful and was barely used. A definite explanation is difficult

to give, but one of the respondents stated that the ability to comment and post new needs might erase the need for branches. The feature might also not have been understood by participants. ‘Merging’ was suggested as a missing feature that would solve the problem of duplicate or similar needs.

8.5.6. Unobtrusive but improvable gamification

While the stakeholders were engaged and slightly agreed with the statement that the game elements made the experience more pleasant, they said to take little notice of the points and leaderboard. On the one hand, this indicates that the game elements were unobtrusive; they did not distract the participants. On the other hand, it might indicate that the elements had little effect. This suspicion is supported by the low incentive to return which was perceived in the observation, questionnaire and SPM interview. In a case with more external stakeholders, improved gamification is essential. Coins as a restricting resource worked well, since the stakeholders that shared many needs or comments at once were ‘paused’ and perceived it as restricting. A problem might be that these stakeholders won’t come back. Other stakeholders had an abundance of coins. The donation of coins was suggested as an opportunity to bridge the gap between these two extremes.

8.5.7. Risks of scale

Both the SPM team and the evaluating experts questioned the effect of a bigger scale on the CCRE method. Concrete risks that might emerge are:

- A high number of non-expert stakeholders that post trivial needs;
- An incomplete or unevenly distributed crowd, which is not a good representation of the complete stakeholder group;
- Managing the crowd effectively and giving feedback to each need is time-consuming.

However, we learn from industry examples such as Stack Overflow that large crowds are able to organize themselves and correct its members. This should make the first and third risk less likely. As expert #2 mentioned, the scale might also be controlled by scoping the question more precisely. With a more specific focus, the contributions of stakeholders will be more valuable.

8.5.8. Expectations

If external stakeholders suggest needs that are supported by many other members of the crowd, it might still be the case that the SPO has reasons not to implement the corresponding requirement. These reasons could be cost- or risk-related, which makes it difficult to explain the decision to the external stakeholders, who think that the many votes suggest that the requirement is a must-have. As identified by two experts, an expectation is created and not fulfilled, which will probably disappoint the crowd.

8.5.9. Required transparent context

Searching for the contexts in which the CCRE method would and would not work, it became apparent that the organization that implements the method should be highly transparent and willing to discuss features and priorities openly. This is probably difficult for companies in a highly competitive environment, since future decisions are largely secret. A public organization, or an organization that already has a crowd (e.g. a crowdfunding platform) would be an ideal place in which this transparency is present.

9. Conclusions

This thesis described an extensive study on the applicability of crowdsourcing and gamification in software requirements engineering. Guided by the question “How can crowdsourcing be used to improve requirements engineering in software production” we spent half a year searching for an answer and a validation of that solution. The answer comes in the form of the Crowd-Centric Requirements Engineering method; eight phases, 42 activities and 29 concepts that should help Software Producing Organizations to improve their RE practices by the application of crowdsourcing and gamification. The method is based on a literature study and interviews with experts from related fields.

A case study on the beta version of Qubus 7 allowed us to instantiate and thereby demonstrate a large part of the method. The demonstration was evaluated by an observation, a participant questionnaire, an interview with the product management and interviews with experts. This evaluation confirmed that the CCRE method was a potential way to use crowdsourcing for the improvement of requirements engineering in software production. In this study, the method provides a useful process and leads to useful requirements, engaged stakeholders and valuable interaction among the stakeholders.

The method does not work in every imaginable context; an SPO that wants to implement CCRE should be transparent and open to discussion. The ‘Feasibility analysis’ should however help to quickly determine the potential for crowdsourcing. Organizations should also beware for the expectations that the openness might create among their external stakeholders. Another constraint on the success of CCRE is that the SPM team should have a sufficient experience, commitment and training to make the right conversion from the suggested needs and comments to the requirements for the focus group and Product Backlog. Knowing what question to ask and how to search for the ‘why’ behind needs is essential during the ‘Crowd involvement’.

There is also room for improvement. While the basic functionality of the prototype – *Refine* – was highly valued, the ability to merge needs was missing and the game elements were not as effective as they could be. These improvements are specific to instantiations of the method; the presented version of CCRE can be used as a solid and proven template.

9.1. Discussion

While the study resulted in valuable conclusions, there are several points of discussion. These points are described in this section, grouped as threats to content validity, internal validity, external validity and reliability and some general issues.

9.1.1. Content validity

In this study, we pursued to improve the quality of RE in an SPO. “Content validity is the degree to which elements of an assessment instrument are relevant to and representative of the targeted construct for a particular assessment purpose” (Haynes, Richard, & Kubany, 1995, p. 238). RE quality was the targeted construct in this thesis. The composition and related elements of this construct were not found in the literature study. Section 2.2.5 described that we considered engaged stakeholders and useful requirements as the factors that characterized qualitative requirements engineering. Additionally, the perceived usefulness of crowdsourcing in specific RE phases and CCRE in the demonstration was determined by the external experts and the questionnaire respondents respectively.

While we have tried to measure relevant elements of our targeted construct, a missing body of knowledge on the exhaustive list of elements creates a threat to the content validity.

9.1.2. Internal validity

As explained in Section 2.3.1, internal validity refers to the authenticity of causal relationships. In our case, this means that the results can be addressed to the CCRE method and not to any other forces. The process and outcome of the method are of course very dependent on the organization's and product's context, which should be identified in the 'Feasibility analysis'. While this is a threat to internal validity, the situational adjustment of the method should make it possible to achieve the pursued results with the CCRE method.

Another threat to the internal validity was that participants of the demonstration knew that the case was part of a scientific study, which can be seen as an external force. This might have caused that the crowd behaved more appropriately than they would in a case that is not related to a study.

The small group eliminated the initial idea to have a control group, which would suggest needs in an alternative and more traditional RE process. Such a comparison would have made the internal validity of the case stronger since the differences in results are caused by the differences between the two RE processes.

9.1.3. External validity

External validity refers to the generalizability of results. We strived to make the results generalizable by explaining situational factors and asking three external experts for their opinion on the usefulness of the method and its results beyond the case. However, we acknowledge that the method was demonstrated and evaluated based on a single case study, which in general results in poor external validity. As suggested by the experts, the outcome of the CCRE method is questionable with a larger crowd.

9.1.4. Reliability

The reliability refers to the consistency and repeatability of the study. The process of developing, instantiating and evaluating the method has been elaborately described in order to make the study repeatable. However, the immaturity of the field and the dynamic situational factors make it impossible to get the same results in a second study. Experts will presumably change their opinion on crowdsourcing and gamification when the trends become more mature. Additionally, the organization and the product that were the subjects of research will also evolve, which results in a different crowd and different input from the crowd.

9.1.5. General

Some obstacles during the instantiation of the method negatively affected the quantity and quality of the results. Due to some delay in the development of Qubus and the launch of *Refine* just before the Christmas holidays, it was difficult to involve real clients and users. A larger group of (external) stakeholders would have strengthened the results and would have given us more insights about the experience of end-users.

The design choices in developing *Refine* were partly dependent on the limited time that was available for this study, in which the focus was on the overall method. The choice for game elements was partly motivated by the study of Hamari and Koivisto (2013), but could have been improved by a comparison of game elements and their effects. The exact values of the points and coins were somewhat arbitrary, but could also be determined in an experimental way.

Finally, we assume that the addition of notifications would have significantly improved the effect of game elements and thereby the motivation of users. Getting an email when someone comments or votes on your need or comment is an easy but personal way to keep stakeholders updated in real time.

9.2. Future work

The topics of crowdsourcing, gamification and SPM are young and immature; the realm of the combination of these concepts is in its infancy. While this study has provided a useful method for organizations to maneuver in this field, it is only a fundament for the scientific opportunities and optimization of Crowd-Centric Requirements Engineering. We distinguish these opportunities in two sections, future work on the method and a Product Backlog for *Refine*.

9.2.1. CCRE Method

The application of the CCRE method should be extended in various dimensions. As mentioned in the discussion, the first dimension is the size of the stakeholder group. A case with more stakeholders gives an answer to the uncertainty whether a larger crowd results in wisdom or chaos, although our findings so far suggest the former.

A second dimension is the duration of the case study. No Development Sprint has been included in this case and thus the crowd has not received feedback on the identification and focus group discussions involving their needs. It would be very insightful to see several iterations of crowd involvement, in which the crowd is expected to constantly learn. In a more elaborate case, it will also be possible to spend more time on a training for the SPM team to correctly use the platform and conduct the 'Requirements identification' phase more effectively. More work is needed to define the training that should be provided.

The third dimension is the amount of cases. Qubus 7 has been the subject of the demonstration in this study and while the context was described in detail, it was only one context. To completely validate CCRE, the method needs to be instantiated in varying contexts, with different instantiations of activities. The effects of situational factors can thereby be determined.

9.2.2. *Refine*

In addition to the research on game elements that has been proposed in Section 9.1, a list of requirements has been evolving during the instantiation of the method and the use of the prototype. This confirms the value of iterative development; *Refine* has provided great insight into the application of CCRE, but can be further improved to provide even more insight. To stay consistent with the method, Table 9-1 provides five requirements with their relative priority. A short explanation for each of the requirements follows the table.

Table 9-1: Some requirements for the next version of *Refine*

Requirement	Rel. business value	Rel. penalty	Total benefit	Total benefit %	Rel. costs	Costs %	Rel. risk	Risk %	Rel. priority
Need references	6	7	13	24%	2	8%	3	14%	0,415
Time element	5	6	11	20%	3	12%	2	9%	0,369
Merging	7	8	15	27%	4	16%	4	18%	0,199
Coin sharing	4	3	7	13%	6	24%	7	32%	0,035
Feedback mining	6	3	9	16%	10	40%	6	27%	0,029
Total	28	17	55	60%	25	52%	22	59%	

Need references allow stakeholders to hyperlink to a need within a need description or comment. This makes it easier to interrelate needs. A time element limits the lifetime of a particular need on the platform. After

this lifetime, it will remain in the needs database, but not be open for votes and comments. Merging is the opposite of branching; it combines multiple needs into one need and thereby provides a solution for duplicate or similar needs. Coin sharing has been suggested by a respondent to the questionnaire and allows 'rich' stakeholders to give coins to stakeholders with good ideas but a lack of coins (e.g. because they comment a lot). Finally, feedback mining would implement the mining activity during crowd involvement in *Refine*. Feedback shared on social media could for instance be converted to needs or comments on the platform. The cost and risk value in Table 9-1 show that this would be a complex feature.

10. References

- Adepetu, A., Ahmed, K. A., Abd, Y. Al, Zaabi, A. Al, & Svetinovic, D. (2012). CrowdREquire: A Requirements Engineering Crowdsourcing Platform. In *AAAI Spring Symposium: Wisdom of the Crowd*.
- Aliance, A. (2001). Manifesto for Agile Software Development. Retrieved from <http://agilemanifesto.org/>
- Ambati, V., Vogel, S., & Carbonell, J. (2010). Active Learning and Crowd-Sourcing for Machine Translation. In *Proceedings of the Seventh conference on International Language Resources and Evaluation*.
- Beck, K. (2000). *Extreme Programming Explained: embrace change*. Addison-Wesley Professional.
- Bekkers, W. (2012). *Situational Process Improvement in Software Product Management*. Utrecht University.
- Bekkers, W., Spruit, M., van de Weerd, I., van Vliet, R., & Mahieu, A. (2010). A situational assessment method for software product management. In *Proceedings of the 18th European conference on information systems*.
- Bekkers, W., van de Weerd, I., Spruit, M., & Brinkkemper, S. (2010). A framework for process improvement in software product management. *Systems, Software and Services Process Improvement*, 1–12.
- Berander, P., & Andrews, A. (2005). Requirements Prioritization. In *Engineering and managing software requirements* (pp. 69–94).
- Bourque, P., & Fairley, R. E. (2014). *Guide to the Software Engineering Body of Knowledge Version 3.0* (p. 335).
- Brabham, D. C. (2008a). Crowdsourcing as a Model for Problem Solving: An Introduction and Cases. *Convergence: The International Journal of Research into New Media Technologies*, 14(1), 75–90.
- Brabham, D. C. (2008b). Moving the crowd at iStockphoto: The composition of the crowd and motivations for participation in a crowdsourcing application. *First Monday*, 13(6). Retrieved from <http://pear.accc.uic.edu/ojs/index.php/fm/article/view/2159/1969>
- Brabham, D. C. (2010). Moving the Crowd At Threadless. *Information, Communication & Society*, (8), 1122–1145.
- Brabham, D. C. (2011). Reining in Crowdsourcing. *Crowdsourcing.org*. Retrieved from <http://www.crowdsourcing.org/editorial/reining-in-crowdsourcing/2547>
- Brewer, M. (2000). Research design and issues of validity. *Handbook of Research Methods in Social and Personality Psychology*.
- Brinkkemper, S. (1996). Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*, 38, 275–280.
- Bryant, B. R., & Lee, B. (2002). Two-Level Grammar as an Object-Oriented Requirements Specification Language. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences* (pp. 3627–3636).
- Cao, L., & Ramesh, B. (2008). Agile requirements engineering practices: An empirical study. *Software, IEEE*.

- Carlshamre, P. (2002). Release Planning in Market-Driven Software Product Development: Provoking an Understanding. *Requirements Engineering*, 7(3), 139–151.
- Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., & Natt och Dag, J. (2001). An industrial survey of requirements interdependencies in software release planning. In *Proceedings of the 5th IEEE International Symposium on Requirements Engineering* (pp. 84–91).
- Chandler, D., & Kapelner, A. (2013). Breaking Monotony with Meaning: Motivation in Crowdsourcing Markets. *Journal of Economic Behavior & Organization*, 90, 123–133. Human-Computer Interaction.
- Chemuturi, M. (2013). *Requirements Engineering and Management for Software Development Projects*. Springer.
- Cheng, B., & Atlee, J. (2007). Research directions in requirements engineering. *Future of Software Engineering*.
- Clarke, P., & Connor, R. V. O. (2012). The situational factors that affect the software development process: Towards a comprehensive reference framework. *Information and Software Technology*, 54(5), 433–447.
- Damodaran, L. (1996). User involvement in the systems design process—a practical guide for users. *Behaviour & Information Technology*, 15(6), 363–377.
- Dave, K., Way, I., Lawrence, S., & Pennock, D. M. (2003). Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. *Proceedings of the 12th International Conference on World Wide Web*, 519–528.
- Davis, F., Bagozzi, R., & Warshaw, P. (1989). User acceptance of computer technology: a comparison of two theoretical models. *Management Science*, 35(8), 982–1003.
- Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: defining gamification. In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments* (pp. 9–15).
- Ebert, C. (2007). The impacts of software product management. *Journal of Systems and Software*, 80(6), 850–861.
- Ebert, C., & Brinkkemper, S. (2014). Software product management – An industry evaluation. *Journal of Systems and Software*, 95, 10–18.
- Eickhoff, C., Harris, C. G., de Vries, A. P., & Srinivassan, P. (2012). Quality through Flow and Immersion: Gamifying Crowdsourced Relevance Assessments. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval* (pp. 871–880).
- Emam, K. El, Quintin, S., & Madhavji, N. H. (1996). User participation in the requirements engineering process: An empirical study. *Requirements Engineering*, 1(1), 4–26.
- Estellés-arolas, E., & González-Ladrón-De-Guevara, F. (2012). Towards an integrated crowdsourcing definition. *Journal of Information Science*, 38(2), 1–14.
- Fernandes, J., Duarte, D., Ribeiro, C., Farinha, C., Pereira, J. M., & Silva, M. M. Da. (2012). iThink: A Game-Based Approach Towards Improving Collaboration and Participation in Requirement Elicitation. *Procedia Computer Science*, 15, 66–77.

- Gartner. (2011). Gartner Says By 2015, More Than 50 Percent of Organizations That Manage Innovation Processes Will Gamify Those Processes. Retrieved from <http://www.gartner.com/newsroom/id/1629214>
- Gorschek, T., Gomes, A., Pettersson, A., & Torkar, R. (2012). Introduction of a process maturity model for market-driven product management and requirements engineering. *Journal of Software: Evolution and Process*, 24(1), 83–113.
- Grünbacher, P., & Hofer, C. (2002). Complementing XP with requirements negotiation. In *3rd Int. Conf. Extreme Programming and Agile Processes in Software Engineering* (pp. 105–108).
- Hamari, J., & Koivisto, J. (2013). Social motivations to use gamification: an empirical study of gamifying exercise. In *Proceedings of the 21st European Conference on Information Systems*.
- Hamari, J., Koivisto, J., & Sarsa, H. (2014). Does Gamification Work ? — A Literature Review of Empirical Studies on Gamification. In *Hawaii International Conference on System Sciences (HICSS)* (pp. 3025–3034).
- Harmsen, A., Brinkkemper, J., & Oei, J. (1994). *Situational method engineering for information system project approaches*.
- Harwell, R., Aslaksen, E., & Hooks, I. (1993). What is a requirement. In *Proceedings of the Third International Symposium of the NCOSE* (Vol. 2).
- Haynes, S. N., Richard, D. C. S., & Kubany, E. S. (1995). Content validity in psychological assessment: A functional approach to concepts and methods. *Psychological Assessment*, 7, 238–247.
- Hevner, A. R., March, S., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, 28(1), 75–105.
- Hofmann, H. F., & Lehner, F. (2001). Requirements Engineering as a Success Factor in Software Projects. *IEEE Software*, 18(4), 58–66.
- Holmström, H. (2004). Virtual Communities for Software Maintenance. In *37th Annual Hawaii International Conference on System Sciences* (pp. 1–10).
- Horrigan, J. (2008). Online shopping. *Pew Internet & American Life Project Report*.
- Hosseini, M., Phalp, K., Taylor, J., & Ali, R. (2014a). The Four Pillars of Crowdsourcing: a Reference Model. In *The IEEE Eighth International Conference on Research Challenges in Information Science*.
- Hosseini, M., Phalp, K., Taylor, J., & Ali, R. (2014b). Towards Crowdsourcing for Requirements Engineering. In *The 20th International Working Conference on Requirements Engineering: Foundation for Software Quality*.
- Howe, J. (2006). Crowdsourcing: A Definition. Retrieved from http://www.crowdsourcing.com/cs/2006/06/crowdsourcing_a.html
- Hull, E., Jackson, K., & Dick, J. (2011). *Requirements Engineering* (Third Edit.). Springer.
- Jalali, S., & Wohlin, C. (2012). Systematic Literature Studies: Database Searches vs. Backward Snowballing. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement* (pp. 29–38).

- Kabbedijk, J., Brinkkemper, S., Jansen, S., & van der Veldt, B. (2009). Customer Involvement in Requirements Management: Lessons from Mass Market Software Development. In *2009 17th IEEE International Requirements Engineering Conference* (pp. 281–286). Ieee.
- Kärkkäinen, H., Jussila, J., & Multasuo, J. (2012). Can Crowdsourcing Really Be Used in B2B Innovation? In *Proceedings of the 16th International Academic MindTrek Conference*, 134–141.
- Kaufmann, N., & Veit, D. (2011). More than fun and money. Worker Motivation in Crowdsourcing – A Study on Mechanical Turk. In *AMCIS* (pp. 1–11).
- Kaulio, M. A. (1998). Customer, consumer and user involvement in product development : A framework and a review of selected methods, *9*(1), 141–149.
- Keil, M., & Carmel, E. (1995). in *Software Development*, *38*(5), 33–44.
- Kittur, A., Chi, E. H., & Suh, B. (2008). Crowdsourcing User Studies With Mechanical Turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 453–456.
- Kujala, S. (2003). User involvement: a review of the benefits and challenges. *Behaviour & Information Technology*.
- Kujala, S., Kauppinen, M., Lehtola, L., & Kojo, T. (2005). The role of user involvement in requirements quality and project success. In *13th IEEE International Conference on Requirements Engineering* (pp. 75–84).
- Kujala, S., Kauppinen, M., & Rekola, S. (2001). Bridging the Gap between User Needs and User Requirements. In *the Panhellenic Conference with International Participation in Human-Computer Interaction PC-HCI* (pp. 45–50).
- Lamsweerde, A. Van. (2000). Requirements Engineering in the Year 00: A Research Perspective. In *Fifth IEEE International Symposium on Requirements Engineering* (pp. 5–19).
- Lim, S., Damian, D., & Finkelstein, A. (2011). StakeSource2. 0: using social networks of stakeholders to identify and prioritise requirements. In *Proceedings of the 33rd international conference on Software engineering* (pp. 1022–1024).
- Lim, S., & Finkelstein, A. (2012). StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation. *IEEE Transactions on Software Engineering*, *38*(3), 707–735.
- Lim, S., Quercia, D., & Finkelstein, A. (2010). StakeSource: harnessing the power of crowdsourcing and social networks in stakeholder analysis. In *32nd ACM/IEEE International Conference on Software Engineering* (Vol. 2, pp. 239–242).
- Loucopoulos, P., & Karakostas, V. (1995). *System Requirements Engineering*. McGraw-Hill Book Company Europe.
- Maglyas, A., Nikula, U., & Smolander, K. (2011). What Do We Know about Software Product Management ? – A Systematic Mapping Study. In *Proceedings of the Fifth International Workshop on Software Product Management (IWSPM)* (pp. 26–35).
- March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support Systems*, *15*(4), 251–266.

- Nawrocki, J., Jasinski, M., Walter, B., & Wojciechowski, A. (2002). Extreme Programming Modified: Embrace Requirements Engineering Practices. In *Proceedings of the IEEE Joint International Conference on Requirements Engineering* (pp. 303–310).
- Nerur, S., & Balijepally, V. (2007). Theoretical Reflections on agile development methodologies, *50*(3), 79–83.
- Newkirk, J., & Martin, R. C. (2000). Extreme Programming in Practice. In *Addendum to the 2000 Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications*, 25–26.
- Nicholson, S. (2012). A User-Centered Theoretical Framework for Meaningful Gamification. In *Proceedings of Games + Learning + Society 8.0*.
- Nuseibeh, B., & Easterbrook, S. (2000). Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering* (pp. 35–46).
- Pacheco, C., & Garcia, I. (2012). A systematic literature review of stakeholder identification methods in requirements elicitation. *Journal of Systems and Software*, *85*(9), 2171–2181.
- Pang, B., & Lee, L. (2006). Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, *2*(1-2), 1–135.
- Paulk, M. (1993). *Capability maturity model for software*. John Wiley & Sons, Inc.
- Peffer, K., Tuunanen, T., Rothenberger, M. a., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77.
- Pohl, K. (2013). The three dimensions of requirements engineering. In *Seminal Contributions to Information Systems Engineering* (pp. 63–80).
- Potts, C. (1993). Software-engineering research revisited. *IEEE Software*, *10*(5), 19–28.
- Regnell, B., & Brinkkemper, S. (2005). Market-driven requirements engineering for software products. *Engineering and Managing Software Requirements*, 207–308.
- Regnell, B., Höst, M., Natt och Dag, J., Beremark, P., & Hjelm, T. (2001). An industrial case study on distributed prioritisation in market-driven requirements engineering for packaged software. *Requirements Engineering*, *6*(1), 51–62.
- Regnell, B., Svensson, R., & Wnuk, K. (2008). Can we beat the complexity of very large-scale requirements engineering? In *Requirements Engineering: Foundation for Software Quality* (pp. 123–128).
- Ribeiro, C., Farinha, C., Pereira, J., & Mira da Silva, M. (2014). Gamifying requirement elicitation: Practical implications and outcomes in improving stakeholders collaboration. *Entertainment Computing*.
- Robertson, M. (2010). Can't play, won't play. Retrieved from <http://www.hideandseek.net/2010/10/06/cant-play-wont-play/>
- Sawyer, S., & Guinan, P. (1998). Software development: Processes and performance. *IBM Systems Journal*, *37*(4), 552–569.

- Schenk, E., & Guittard, C. (2009). Crowdsourcing: What can be Outsourced to the Crowd, and Why? In *Workshop on Open Source Innovation* (pp. 1–29).
- Schwaber, K. (1997). SCRUM Development Process. *Business Object Design and Implementation*, 117–134.
- Seyff, N., Graf, F., & Maiden, N. (2010). Using Mobile RE Tools To Give End-Users a Voice. In *18th Requirements Engineering Conference (RE)* (pp. 37–46).
- Snow, R., Connor, B. O., Jurafsky, D., Ng, A. Y., Labs, D., & St, C. (2008). Cheap and Fast — But is it Good?: Evaluating Non-Expert Annotations for Natural Language Tasks. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 254–263).
- Sommerville, I. (2011). *Software Engineering*. Pearson.
- Souer, J., van de Weerd, I., & Versendaal, J. (2007). Situational Requirements Engineering for the Development of Content Management System-based Web Applications. *International Journal of Web Engineering and Technology*, 3(4), 420–440.
- Spivey, J. (1989). An introduction to Z and formal specifications. *Software Engineering Journal*, 4(1), 40–50.
- Stapleton, J. (1997). *DSDM, Dynamic Systems Development Method* (p. 163). Cambridge University Press.
- Storey, M., Deursen, A. Van, & Cheng, L.-T. (2010). The Impact of Social Media on Software Engineering Practices and Tools. In *Proceedings of the FSE/SDP workshop on Future of software engineering research* (pp. 359–363).
- Surowiecki, J. (2005). *The Wisdom of crowds*. Random House LLC.
- Thayer, R. H., & Dorfman, M. (1997). *Software Requirements Engineerings* (Second Edi.). IEEE Computer Society Press.
- The Standish Group. (2009). *CHAOS Summary 2009*. Retrieved from <http://emphasysbrokeroffice.com/files/2013/04/Standish-Group-CHAOS-Summary-2009.pdf>
- Van de Weerd, I., & Brinkkemper, S. (2008). Meta-Modeling for Situational Analysis and Design Methods. In *Handbook of research on modern systems analysis and design technologies and applications* (pp. 35–54).
- Van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., & Bijlsma, L. (2006). Towards a Reference Framework for Software Product Management. In *14th IEEE International Requirements Engineering Conference* (pp. 319–322). Ieee.
- Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011). The agile requirements refinery: Applying SCRUM principles to software product management. *Information and Software Technology*, 53(1), 58–70.
- Wieggers, K. E. (1999). First Things First: Prioritizing Requirements. *Software Development*, 7(9), 48–53.
- Witt, M., Scheiner, C., & Robra-Bissantz, S. (2011). Gamification of Online Idea Competitions: Insights from an Explorative Case. *Informatik Schafft Communities*, 192.

- Xu, L., & Brinkkemper, S. (2007). Concepts of product software. *European Journal of Information Systems*, 16(August), 531–541.
- Yu, E., & Mylopoulos, J. (1994). Understanding “why” in software process modelling, analysis, and design. In *Proceedings of the 16th international conference on Software engineering* (pp. 159–168).
- Zand, D. E., & Sorensen, R. E. (1975). Theory of Change and the Effective Use of Management Science. *Administrative Science Quarterly*, 20(4), 532–545.
- Zave, P. (1997). Classification of research efforts in requirements engineering. *ACM Computing Surveys*, 29(4), 315–321.

Appendix A – Expert interview questions

Introduction

We are currently researching requirements engineering and more specifically, the potential impact of crowdsourcing and gamification in this process. With this expert interview, we want to learn from your expertise on this topic. The results will be used, in combination with theoretical findings, to determine which activities and artifacts should be present in a Crowd-Centric Requirements Engineering method.

The first part of the interview will focus on current industry practices, as you perceive them. Afterwards, I would like to hear your suggested improvements on these practices and finally, we will discuss two possible levels of crowdsourcing in requirements engineering.

Questions

The following lists provide the questions for a semi-structured interview. While the numbered questions are the base questions, the bullets provide more guidance when necessary.

We will roughly phase requirements engineering in collection, refinery into implementable system requirements, deciding on the implementation by assigning priorities and managing change.

- 1 Which activities are currently conducted in industry in order to collect software requirements?
- 2 How are these requirements subsequently refined to be implementable system requirements?
- 3 How are priorities of the requirements determined?
- 4 How is potential change (of content or priority) implemented in the RE process?

- 5 Looking at the activities we just identified *<repeat the identified activities>*, which activities lead to satisfactory results and should not be changed?
- 6 Looking at the activities that you do not find satisfactory, how could these be improved?
- 7 What should the role of the software's end-users be in the whole RE process?
- 8 When you think of crowdsourcing (a part of) RE, what are your ideas?

Assume we want to use a community of users to elicit requirements from.

- 9 How can a user community be created?
 - Who are the relevant stakeholders (internal & external) that should take part?
- 10 When the community is created, how can its users be engagement and motivated?
- 11 How can the concept of gamification trigger motivation?

Assume we also involve the users in the negotiation of requirements, which means taking part in a discussion, and the prioritization, which means that they influence the ranking of requirements.

- 12 What are your ideas on this involvement?
- 13 How could a product software company control the discussions?
- 14 How can we avoid bias in these activities (e.g. preventing campaigning for votes)?
- 15 In what other ways can we improve the quality of the results?

Definitions

Crowdsourcing

"An online, distributed problem solving and production model that leverages the collective intelligence of online communities for specific management goals"

Gamification

"The use of game design elements in non-game contexts"

Appendix B – Product Management interview questions

- 1 What are advantages compared to alternative ways of engineering requirements?
 - Do you have the feeling that the crowd has done some of your work?
 - Did you gain new insights from the needs and comments?
- 2 What are disadvantages compared to alternative ways of engineering requirements?
 - Did you receive unwanted/trivial/useless requirements?
- 3 How useful is the set of resulting requirements?
 - Is the amount of requirements satisfactory?
 - Are the requirements detailed enough to prepare a focus group?

Appendix C – Expert evaluation statements and questions

Statements

- 1 “Crowdsourcing is useful for the elicitation of requirements” ↑
- 2 “Crowdsourcing is useful for the negotiation of requirements” ↑
- 3 “Crowdsourcing is useful for the prioritization of requirements” ↑
- 4 “Crowdsourcing is useful for the specification of requirements” ↑
- 5 “Crowdsourcing is useful for the validation of requirements” ↑
- 6 “Gamification is an effective way to engage users in requirements engineering” ↑
- 7 “Gamification is an effective way to stimulate innovation in requirements engineering” ↑
- 8 “CCRE would improve the quality of my requirements engineering process” ↑
- 9 “*Refine* does not provide a useful list of requirements” ↓
- 10 “The quality of the requirements is lower than the input of our users/customers” ↓
- 11 “The resulting requirements seem detailed enough to use in a focus group” ↑
- 12 “The resulting requirements seem detailed enough to put on the Product Backlog” ↑

Open questions

- 1 What are the pros of this CCRE method?
- 2 What are the cons of this CCRE method?
 - What are obstacles in applying it?
 - What are weaknesses that are hard to solve?
- 3 In which context would the method be useful?
 - Can you provide a concrete example of a product of yours (or that you know)?
- 4 In which context would the method not be useful?
 - Can you provide a concrete example of a product of yours (or that you know)?

Appendix D – Interview Summaries

Contents

Expert 1	118
Expert 2	120
Expert 3	122
Expert 4	124
Expert 5	126
Expert 6	128
Expert 7	131
Expert 8	134
Expert 9	136
Expert 10	137

Expert 1

You know the story of the man who asked his wife to get bread, and if they have eggs, to get six? He came home with six breads. In general, software development starts with a good idea. It has to be determined whether the idea is feasible, technically possible and whether people want it. You don't want to make a round at people to ask what they want in the idea.

When people come to us, they have an idea. We use workshops and brainstorming to further develop those ideas, within a certain framework. You have to lead the discussion, but the art is to also be open. Priority in business is easy, an idea can be converted to money. They will choose what generates the most or costs the least money. For the government, it's already harder. Everybody thinks they have the most important function. I think you always have to support the customer instead of the sales engineer. In practice, you see that every exception is automated in a system. It would be easier to choose to only create the main streams, but nobody dares to do this.

When the requirement is delivered to the developer, it only contains half a specification. We still see a lot of waterfall projects, where there is no contact while software is being built. I'd prefer agile. Build the main streams, deliver every two weeks, interact every delivery and see where you are. You should also constantly have an expert in your team, to have continuous contact. You have to take care that you don't simplify too much, but that rarely happens.

Every agile project, there are still difficulties. You have personal preferences of the product owner, with his own situation as a reference framework. You cannot really solve that, it is how it is. The good thing of agile is that you cannot really make a mistake, cause you're immediately steered into the right way. You should have your demo for a bigger audience: the development team, the product owner and his manager and maybe a bigger audience of users. For a company with a million customers, it's of course very hard to involve users.

User Involvement

"You try to help users reach their goals, within certain conditions". A user is the best person to determine this. You should involve a user early, but that's complicated. Sometimes, there are customer panels, but you cannot have them every two weeks at deployment. One difficult factor is the scale, but also the lack of hierarchy. When we would send someone from our company, they will give feedback, because they get paid or a rating. Users miss the hierarchy, to structurally make time at the right moment. This is not only motivation, it is difficult to organize.

Crowdsourcing

Crowdsourcing has potential when there is an incentive for users. In crowdfunding, there is an incentive like interest or ownership. This has to happen for crowdsourcing too, free software or some other monetary incentive. That is often how it happens in services, time is money. A good feeling is the minimum that you should have. The crowd determines whether they get a good feeling.

Gamification

Gamification also helps, but only within a group and only limited. It works in a group in which I want to show off, but after a while I might lose interest. You should have a sustainable benefit. Open source software has peers that want to help and want to be seen for their help. Whether they program or give

feedback, it is all relevant and appreciated. As long as the group is close and fun, it works. You might have a flow of people, which could be good.

Negotiation & Prioritization

It is a good idea to let people discuss, also about priority. I don't know whether it will turn into a chaos. You should be open to be challenged by others. You obviously have to moderate it, especially when it's on the internet. There is a point in that you ignore your vision when you listen too much to users. But when I'm a software developer, I don't have a vision about online banking, I'm just the developer. This is different for a software product company. But still, let users guide you is not a weakness. It is very IT 1.0 to know what is good for your users.

Bias

When a crowd is heterogeneous, you won't have bias, then it's really good when people ask for votes. However, a crowd is never completely heterogeneous. Then you should ask yourself when the crowd is a good steering mechanism. The most difficult thing in a crowd is to insert people that are not part of it, to motivate them.

Lessons learned:

- In short cycles, do not simplify too much;
- Do not say that you know what is good for your users;
- Do not only motivate, but also get feedback structured;

- Have short development and feedback cycles with your customer;
- Have constant contact with an expert;
- Demo for a bigger audience, possibly involving users;
- Make sure that the crowd gets a good feeling from being involved;
- Create a sustainable benefit for the crowd;
- Be prepared to see crowd members coming and leaving;
- Let people discuss, also about priority;
- Moderate discussions on an online platform;
- Have a heterogeneous crowd.

Expert 2

The content of the RE process is very dependent on business processes and the kind of software. When software is tailor-made, there are often interviews with users, managers and process designers, leading to a set of requirements. In standard packages, customers, consultants and technological trends determine the software requirements.

For the first version of the Innovation Factory software, most requirements came from us. Consultants had certain experiences which were inspiration. We do involve customers, but you have to distinguish big and small things. For example, we are helping with the front side of innovation, the idea generation. Customers also asked us for the back side of implementing the ideas. We translated the customer wish, presented our translation and developed a first release which is now used. Customers often come up with something, but do not actually use it. Or they think they need A, while they actually need B.

I want to have one release for all our customers, so if there is a wish, we visit other customer to see whether they are also interested. We use our own software to elicit ideas internally. The usefulness of ideas is very dependent on the type of user. We often have five different roles involved: end-users, moderators, innovation managers, platform owners and technical maintenance, which has the effect that ideas evolve quickly. Ideas of users are often not very good. Software is looking good and is well used, but they think it doesn't look sexy.

All stakeholders have different wishes, so we first determine who we wanted to hear and then approach them. We always have idea challenges, in which we start with a workshop to determine the question and target group, we then select communication instruments, set up the platform and start an initial marketing campaign. This campaign often involves a video from the big boss to get people involved.

We often use our software internally at companies. Consumers are a completely different world, since they don't discuss. Employees get time and money to further develop ideas, consumers need monetary rewards. They often take someone else's idea and turn it around, to have their 'own' idea. When people are emotionally involved in a product, you can motivate them without money. We also invited people to companies as a reward.

We have some forms of gamification. You can see the most active user, which is weighted over time. We don't use badges since that makes it harder for new entrants to keep up with people that already have badges. You cannot take them away from people.

You can also use crowdsourcing for negotiation and prioritization, maybe you can have sequences of competitions then. First, users give their ideas for functionalities, then there is a challenges for employees to enhance those. You can present the specific elaboration to users. I would not let users determine priority. As a supplier, you have more input because you have an overview of all users.

We don't really see a chaotic platform, we ask a specific question and have moderators. When a comment or idea is not relevant, we put it in draft mode and say that it doesn't fit the purpose.

Lessons learned:

- Customer don't know what they need
- Determine who you want to hear
- Select communication instrument and invent an initial marketing campaign

- Prevent users from 'hijacking' ideas
- Invent rewards to thank people for getting involved
- Weigh 'points' over time, to allow new users to keep up
- Have a sequence of competitions to further develop requirements
- Don't let users prioritize, they don't have the overview
- Ask a specific question
- Put unwanted comments in draft mode with an explanation

Expert 3

Requirements elicitation is done through iterative workshops and prototyping. The first workshop starts with best guesses, based on an initial discussion. This is easier than starting with an empty sheet. In the workshop, a number of stakeholders is present to give an idea about the system's context and processes. Ideally, users are involved in the workshops, but this is not always possible. There are sometimes splitted sessions with different actors. The sessions are not only used for initial elicitation, but also cover the detailed execution of processes and the system's presentation on a screen.

Priorities are also discussed, things could be too expensive or have too little value. In the past, MoSCoW was used for this, but requirements are currently ordered, after which a line determines what is included and what's not. MoSCoW led to every requirement being a 'must'. The point system that is used in Scrum is somewhat abstract, but important to make decisions.

The best aspects of the process are the wealth of interaction and visualization. However, determining priorities is hard, as well as estimating the time that is needed to develop something.

User involvement

If possible, an end-user should always be present. We've experience an example in which a requirement from the customer was improved user-friendliness. Then we said: we should test the system with those users. This leads to clarities, for example whether they understand content or where to start in a system. Companies sometimes want to in preparation approve what is shown to these end-users.

Users should be motivated by knowing that their feedback is used.

Crowdsourcing

In requirements elicitation, it might be hard to get answers through crowdsourcing; questions will still remain at the client side. You will always need extra questions and interpretation. However, it might be good for initial idea generation, especially for an application like Facebook. An interesting example is the website of 'Allerhande'. They provided a link to the new website on the old one, with the ability to provide feedback. In this, I like democracy, in which everyone's voice is heard. This should however be determined by what you are looking for.

The results should become apparent in the next version, that's the best motivator.

Gamification

It could help. The danger of providing incentives however, is that you get a lot of rubbish. You should put the incentive on high quality.

For negotiation and prioritization, you need a faithful group of users, which is difficult. This group could select itself, because they like it. Customer panels have successful examples, but for an application or website this is difficult. In customer panels, people have the feeling that they contribute and help others. In website feedback, you're helping a company in building a better website. Early access is only useful for very popular applications, like Angry Birds.

Controlling discussions could be done with moderation. You could try to leave that to the crowd, but it might turn into an unexpected direction. This could lead to a completely uncontrolled discussion.

If the tax administration releases its first version and lets users provide feedback, discuss and vote, that might work, because it's a returning problem for users every year.

For a project in which we successfully involved users, people worked with an insufficient system every day, which was why it worked. They wanted to be involved.

Lessons:

- Do take the current user friendliness and application usage into account, since this influences users' preferred involvement;
- Do determine a preferred outcome, to decide who to involve;
- Do involve users when possible;
- Do ask extra questions, to get to the right interpretation of feedback;
- Do show users results of their feedback;
- Do order requirements, to prevent everything from being a 'must have';

- Don't use gamification to increase quantity of involvement;
- Don't let the crowd manage their own discussions;

- For certain types of applications, don't ask open questions without interaction.

Expert 4

Currently, product owners develop a list of initial requirements in cooperation with the client. This list is subsequently discussed with the product team in order to create an understanding about the requirements. This often happens with prototyping techniques. Priorities are assigned along the way, also in cooperation with the client. After validating the understanding of requirements, the product is being built in a sprint, in which there should be no changes. As a functional designer, it is your job to determine what should be built, not how it is built. This is also the level when you start a sprint. The best part about this process is the iterative nature and the ability to manage change. However, it is difficult to sell, since there has to be a certain amount of trust that the development team is conducting the right activities. For example, the points that are being used to predict effort are instead used to check the team. The product is only really 'finished' after a number of sprints, which has pros (e.g. ability to change) and cons (e.g. showing users a non-finished version might get unsatisfied responses).

User Involvement

The product owner interviews a number of stakeholders (also users), not only the client. Functional designers also join in workshops. In addition to this involvement in elicitation, users are sometimes involved in the acceptance of a release. Feedback is useful for upcoming sprints. Some customers only want to test with the end-user after the implementation of certain functionality, to prevent complaints. It is essential to interact with the users about their feedback, else they will quickly lose interest in contributing.

It is best to always involve users, to understand what the real problem is. Although user involvement is good, in the end it is about customer satisfaction. It is up to the customer whether he wants to involve users. However, if the customer is good, he acknowledges the importance of user satisfaction.

Looking at crowdsourcing, it might be useful to show users an early version in order to get feedback. ING did this with their mobile banking app, which had a feedback button. This information can be used to determine the next step.

Motivation

You can give a group of end-users access to an acceptance environment. A good example is *outsystems* in which users can click on an item in the acceptance environment to provide feedback. Users are generally motivated when they see timely result of their suggestions. A point system might also be a good idea, since it stimulates the generation of good ideas. When you let users suggest, discuss and rate requirements, it might be best to have a small group of representative users. A large group of users in an open source system would potentially lead to a big mess.

An interesting solution would be to let users suggest ideas, work those out with a smaller user group and bring it back into the community to validate the requirements. You could use a point system to let users rate each other's user stories. The representative smaller group could be selected by the product owner and customer. It might be a good idea to introduce guidelines, such as templates or rules of the community.

Another idea is to use part of your sprint for users' ideas. In that way, a company can still have its own vision but involve users.

Lessons learned

- Do not have a large group suggest, discuss and rate requirements;
- Do make your process iterative, allowing for change;
- Do involve users, to understand what the real problems are;
- Do show users an early version to get feedback;
- Do prepare your users for 'unfinished' versions;
- Do provide timely results from users' suggestions;
- Do interact about user feedback;
- Do stimulate good idea generation with points;
- Do allow users to rate each other's user stories;
- Do work with representative smaller user groups to further develop requirements;
- Do validate detailed requirements with the crowd;
- Do introduce guidelines on a crowdsourcing platform;

- Possibly use a part of your sprint for your users' ideas.

Expert 5

Requirements are generated in a number of ways. Sometimes a bug is reported. The interesting thing is to find the underlying problem of the bug. A second way is to look at competitors and their features and modules. A third generator are own ideas. This is difficult for us, since we aren't domain experts in the market that we're developing software for.

To elicit feedback, we'd like to talk to domain experts. When you elicit feedback from users (which are our customers) on a prototype, you only get feedback on the prototype. Your own design then limits what users can tell you. We don't prototype for every release, mainly for the bigger elements of a system. If we don't use prototypes and just asks users for their opinion, we get very specific suggestions, which might not be relevant for other customers. Most of the feedback arrives from the ticketing system.

Priority of requirements is user-driven. We aggregate the requirements and see how many customers can be satisfied with certain requirements and thereby increase business value. Ideally, we want to have a roadmap, but we need domain experts for that. They know where customers will be in one year.

After eliciting a requirements, we make a quick sketch. This is occasionally communicated to the customer. It would be good to test this sketch with domain experts earlier, to get useful feedback. You should actually test requirements with all your customers, but they all have their own priorities. I prefer not to ask users for their opinion, because that creates many one-on-one discussions.

We don't have a user group. Users mainly give suggestions about small things, like buttons that should be different. Functional managers however, will be more high-level and suggest features that software should have in the future. Crowdsourcing will therefore create a list with 2 kinds of feedback. If you give everyone access to this list, they will still have their own priorities. This is partly due to the software, but also due to the size of a customer. A childcare with 30 kids needs different functionality than a childcare with 200 kids. You would then need to give weight to different customers. One customer might have a very different opinion, but may be very important for the business.

There is a lot of variety in the maturity of customers. Some are just followers. When you look at ERP providers, they don't give customers a choice of what they want. The software is as it is and it's difficult to change.

Our customers do offer a good amount of feedback, mainly bugs and changes. We make them aware of the ticketing system during implementation. The implementation is done with key-users. They are the ones who have contact with the system. Not all users have that contact, because they will probably provide mostly 'how-to'-tickets. If you really want to be scalable, you don't want implementation, you only want configuration. Then you get and need the 'how-to' questions, so everybody can configure well.

If there is a button for everyone, it would be interesting to see what kind of questions exist. You might however also create expectations at customers. You could always take the top 3 of prioritized requirements into account and thereby motivate users to reach this top 3. You could then create weight, a customer that always makes the top 3 is a good initiator. You also still have the business value of customers.

Lessons Learned:

- Do not only use prototyping, since this limits the users' scope;
- Consider the variety in customers;
- If you need to incorporate business value, give weight to users;
- Include domain experts in RE;
- Possibly take the top 3 of crowd-prioritized requirements into account, to motivate users to reach it;
- Give weight to users based on the popularity of their additions.

Expert 6

Requirements Engineering is a very broad concept within our company, maybe even much broader than the concept in Software Product Management. When is something a requirement? We call something a wish, which emerges from a customer's need.

At some software companies, there is probably a serial process. I don't always work that way and don't want to work that way. I'll tell you how we work. At the start of the year, we have the customer involvement sessions. Customers are asked to take a list of their most important wishes. The sessions are organized per product group. There is also voting to see which wishes have the most supporters. We document all these wishes as 'document development requests' in our digital archive. There are two types of these requests: bugs and wishes.

I as a product manager make a selection of these requests and assign them to projects. We show these projects at the 'AFAS open'. We put the projects on a roadmap and subsequently realize the projects. Realization contains design, development, testing, documentation and eventually deployment. During the realization, there is no interaction with the customer. Although, with big projects we invite customers at the Quality Center after testing, to see whether the product complies with their expectations.

Customers have wishes, but the actual requirement, what they actually want, is not what they say. They say A, mean B, but want C. In registering the wish, support or I already make some translation through our interpretation. In addition to the involvement sessions, customers are able to contact support via the software. There are only a few customers that bypass the channels, I keep that door locked.

The projects and placing them on a roadmap is very volatile, almost chaotic. Strictly, we're not agile, but we're very flexible.

One of the best aspects of this way of working is that we have very good digital assurance of development requests. Although there is sometimes chaos in my head, the wishes are organized in the digital system. Sometimes, we have a prospect and we promise them a feature. This will also be secured in the digital system. Also, everybody arrives in the same workflow and can suggest wishes. A previous employer had excel sheets that we discussed every week. That was a worse situation.

Insight in the number of customers that have a specific wish could be improved. We strive to have one description per wish and we also have a counter for these wishes. I would like to see it visualized, we're currently working on that. The feedback from us to the customer could also be better. We occasionally do that, but not always. Now we have implemented a wish eight years after the suggestion, we don't know whether we should provide feedback. Via the AFAS open we give some feedback, but not always. We also consider giving customers insight into the wish list. I don't think it's necessary and not necessarily good, since it creates expectations. However, it creates transparency.

User Involvement

At the customer involvement session, everybody can come, but each company has only one vote. You prevent that customers bring a hundred employees to vote for their idea.

With our process, on the one hand we try to find the pain points of the daily processes of our customers. On the other hand, with big projects, we ask customers what they think of our solution.

I don't believe in crowdsourcing. What does the crowd have to do with my company? However, sometimes after the involvement sessions, we have small focus groups when the wish is occurring often. We get some very valuable feedback from those focus groups with a select group of customers. I have various questions then. The innovation does not come from the customer, it's mostly added by us because we think many steps further. As a product manager, you need to know the process of the customer. The most value therefore is in the focus groups.

We don't need a platform where people can give their suggestions, we already have our sessions and support. We don't need to motivate people more, because we already get a lot of input. The customer has a lot of knowledge but cannot make software. I don't see customers come with good ideas. The nature of their ideas is not innovative.

On LinkedIn, there is sometimes discussion around development requests. We respond very considerate, but you see from the responses that people have no idea of the technology behind it. Also functionally, they don't know what is possible and necessary. The discussions are valuable, sometimes people know what they are saying. During discussions, it's really hard to teach people something. That is also not really the role of a product manager. We sometimes make a strategic decision and customers don't like it.

We make user-centric software, but say no a lot. I reject 99,9% of the wishes, that's the art of requirements engineering, making software of which the customer doesn't know that he wants it. A famous example is the iPad and maybe even the Apple Watch. Some people get it, some people don't.

Sometimes we open up our functionality, people can make their own reports, analyses and presentations. Another example of openness are the focus groups. When we let customers choose, we let them choose from things that we have come up with. The degree of vision determines the openness for customer suggestions. We are 'pigheaded' (eigenwijs) and let people know that we make product software. I don't want to let customers determine priorities. For example, 90 percent of the customers might want a functionality that takes up a large share of our capacity. They don't have an insight in that. I don't go to the bakery and ask for a purple bread, which costs him a lot of money. You could tell your customer that it's not realistic to do it, but do you need the whole process for it, instead of telling them immediately that you're not going to do it?

We also have a discussion with bakeries, where some bakeries want one thing, some want the other. There a platform to learn from each other would have value. I don't know if you can get people on a LinkedIn discussion content, or that they just accept it. That's the game.

Lessons learned:

- Don't discuss strategic decisions;
- Don't let users bypass the chosen channel;

- Give customers the opportunity to input their wishes;
- Give each customer group one vote;
- Have small focus groups with a selection of customers;
- As a product manager, think many steps further than the customers do;
- Have a good organization and assurance of your requests;

- Give all stakeholders the same workflow;
- Have an insight in the number of customers that have a specific wish;
- Ask customers what they think of your solution;
- Make software that the customer doesn't know it wants;
- Determine your degree of vision;

- Possibly give feedback to your customers;

- Possibly give customers insight in your wish list.

Expert 7

Requirements Engineering largely happens at design time, pre-deployment. Occasionally, they might revise the system. It also has a heavy involvement of experts. With each domain expert, we have yet another version of the tools.

Research on methods and models has a noble goal, but in practice it's really ad hoc.

In practice, there is also a subset of users, presumed to be representative. This does not work for mobile apps. The set of users is very large and always changing, you don't know who is going to use it. Then you cannot decide on a representative. You also have to think about all possible contexts of use. The way of thinking about requirements engineering needs a revision to cope with this change.

In prioritization, it is assumed that there is a central point of reference in the organization which decides on priorities. Most research focuses on the boundaries of a certain enterprise. If you look at modern software, it is not in the boundary of an enterprise. Who would prioritize requirements in a distributed system? In principle, every system could be subject to change, so the problem does not only exist for distributed systems.

Contracts are not sufficient. You need a new business model for that, liquid contracts. This introduces flexibility.

Companies often use agile methods. They discuss prototypes with the customer. They read about the current business process and conduct interviews. There is no modelling, they don't have time for that. I don't know if it's bad or good, it shows that the current research does not attract the industry. They are implementation-oriented. They are not paid to make use models.

The agile methods in industry are flexible, which is good. Industry is also outcome-oriented, they are sort of pragmatic. Some pragmatism is also needed in research.

User Involvement

In Scandinavian countries, they really believe in co-design. They use workshops to elicit and validate requirements. They have a subset of users, which is at least better than 'representatives'. Another fashion is to look at social media and to look at previous versions (what did users want, where did they struggle). Data mining is also interesting for feedback, there is very little research on this. Social platforms are only used in a backward way, but it could also be used in a forward way: "this is why we engineered it". If you design this platform well, people will use it.

Crowdsourcing

Crowdsourcing is about involving everyone without bias, continuously and dynamically (no static crowd). Crowdsourcing is not only about how many you involve but also how long you involve them. The crowd members are continuous validators of requirements and the genuine source of public opinions.

Most of the communities now are forum-based or wiki-based. It's for a general purpose and very basic. We need a dedicated platform to annotate software with your feedback.

Gamification

Gamification is an umbrella term. Users would like to see how their feedback is used, which is exploration, a form of gamification. You can also allow them to form groups. Even if you are a minority with special needs, you can have a group. Number is very deceiving in crowdsourcing, you should not listen to the majority but look at the groups. You need to look at the reason why they object something.

For this, you need to look through all the input. You could have a peer review by the crowd. This poses a risk if you don't design it well. But we need to do it well, mistakes are part of the game.

Gamification is also culturally dependent. For some people, social recognition is as good as a voucher or t-shirt. There are four types of gamification: socializers, achievers, killers and explorers.

You can give some points or rewards for feedback. This could be used to give weight to the feedback in the future. The more points you get, the more trustworthy you are.

“Gamification is not only for motivating only, but also to help the process itself, help the reasoning itself. If you give a badge, that's not the end of the story. Someone's opinion is becoming more valid in the final decision.” The other thing is awareness. Visualization is critical in this. People would like to see trends for their community, colleagues or peers.

Involving the crowd in prioritization would solve the problem of prejudgments. Opening it to the crowd, opens possibilities to see other stakeholders and other reasons. It's creating transparency and a sort of fairness. You might end up with different versions of the system. It's natural to have groups in the crowd. Don't tread them as uniform, then you might as well take on of them.

You don't need to overcome bias, that's the nature of life, there are people who are influential and who are not. But if you have a company, with a goal for the main stakeholder, they need to have a say in the end. In this case, you need to prevent bias. You don't use a democratic crowdsourced approach but a blended approach.

There are four pillars in wisdom of the crowd: diversity, independence, decentralization and aggregation. These pillars need to be taken into account in conducting crowdsourcing. But if you have a strategic objective, you can interfere.

If people have a feeling that you don't use the feedback, it will die, even if you have given rewards. Like when you write an article in the newspaper and the newspaper is gone afterwards. You have to determine and communicate to the crowd which part of your system is open and what you are going to do with their feedback. Transparency is important.

Crowdsourcing has to be a strategy, it has to be consistent with all other things.

Quality has many dimensions, which are not all crowdsourcable. Like security, you still need design time with a very rigorous validation and verification and hard milestones. Otherwise, the whole system will fail. “This is why we have police”. You need to consider it case-by-case, you cannot say ‘this is going to work’. Losing trust is as important as security.

Lessons learned:

- Be agile and outcome-oriented;
- Involve real users, not representatives;
- Create a dedicated platform to annotate software with your feedback;

- In crowdsourcing, continuously involve people, without bias, dynamically;
- Consider minorities' opinions;
- Use exploration, group forming and rewards as gamification techniques;
- Create visualization of involvement;
- Allow peer reviewing in the crowd;
- Give weight to opinions;
- Involve the crowd in prioritization;
- Form your crowd based on the four pillars of crowd wisdom;
- Determine and communicate to the crowd which part of your system is open and what you are going to do with their feedback;
- Still use rigorous validation for quality dimensions;
- Use a blended crowdsourcing approach when there are strategic goals;
- Consider crowdsourcing case-by-case;

- Potentially use data mining to gather feedback.

Expert 8

Elicitation is engagement with the stakeholders, who could be the client, the user or other actors. The goal is to find out their needs. This could be through interviews, focus groups or even ethnographic analysis. The list of requirements is validated by a Subject Matter Expert, which could also be done by a focus group. For the prioritization, you could do a workshop activity in which the stakeholders get a drop down of requirements. After requirements specification, requirements are usually modelled. You could come up with use case diagrams or use case specifications.

You can get users involved in each stage. For example, they can be involved in requirements validation through focus groups around requirements modelling. Sometimes, scenarios or personas are used for the design of requirements. You need to have some design options that you present to your users. After implementation, you test it with the users. From this, new requirements might emerge. The amount of users determines the type of involvement. For a large-scale socio-technical system, this is the process that we're using.

Getting consensus might take a while. If you could get your user community in a room, it might be easier, compared to engaging with users one-by-one. The perception of the requirements might influence how they modify the requirement. If you involve people as a group, you introduce a decision-making process. You could create a baseline for a decision, e.g. saying that 80% must support a requirement.

In a user community, you want all stakeholders involved. The user group is the driving force, because they are the ones who should be using it. Technical people might give feedback when something is difficult to implement.

The idea of crowdsourcing is really good. If you have different groups (user community, architects, clients, etc.) in the crowd, that's beneficial. You could give a scenario to the crowd. The users might not know how to write requirements, so give them a scenario or something that they could use, think about or discuss. This is especially useful for a complex system.

You could incentivize them by showing the progress of requirements. You could also increase their learning (about the context of the system and from other stakeholder groups) and change their systems thinking. This can help solve conflicts, decision making and an understanding of the problem space.

A scenario could be turned into a serious game. That might increase their system thinking. Let's say you need a new sensor technology for a helicopter that is going to be used by the police and ambulance people. Stakeholder groups might do it in different ways. You could gamify the scenario by creating the different steps. The difference between the stakeholder groups, new requirements might evolve. Based on the decisions in the game, the design solution and exploitation can be optimized.

Crowdsourcing is used to enhance consensus and decision making, gamification is used to enhance the understanding of the problem. You can simulate the real-world environment through gamification to enhance consensus. You might be able to use a ranking or rating and a rationale behind that ranking to structure requirements. You could use scenarios to support ranking, 'in this situation I would rank it this high'. Depending on the requirement, you select which groups of people (communities of practice) should be involved.

Lessons Learned:

- The amount of users determines the type of involvement;
- Include all relevant groups in the crowd;
- Incentivize users by showing the progress of requirements;
- Let stakeholders learn from other stakeholder groups;
- Give users a (gamified) scenario, with different steps;
- Use crowdsourcing to reach consensus;
- Use gamification to enhance problem understanding;
- Simulate the real-world environment using gamification;
- Select the groups of people in the crowd based on the requirement;
- Use a ranking and rationale to structure requirements;

- Possibly create a baseline for requirements.

Expert 9

Requirements Engineering does not really exist of phases, but activities which are intertwined. Requirements Engineers are not translators anymore, but can be seen as the facilitators of the co-design activity. In this, the problem has to be defined together and finally a solution has to be created together. Stakeholders (including clients, programmers, security officers, etc.) are not passive, but active. They sit together at a table instead of being interviewed. While the trend is valid for custom-made software as well as market-driven software, the techniques are very different.

When you discuss functionality, prioritizing is interrelated.

Often, people don't try to be participative which results in a suboptimal solution or use a fake participative approach in which stakeholders are involved but the business analyst decides in the end. In many situations, the people and environment are not ready. Real participative design is hard. It requires trust, freedom, time, a specific plan, people that can listen and so on. The Collaborative Creativity Canvas lists the important features.

"It is less easy to be collaborative in teams remotely". I think it's however absolutely necessary to involve crowd and this has its own field of research.

If you have a market-driven product, you have representatives as stakeholders. You can also hire people to represent users. Improv is a game, which teaches people to be creative. This turns improve in an agile user experience driven prototyping technique. It is fun, challenging and people laugh a lot. It's a form of gamification. People can "free their imagination" completely.

The technique is combined with different techniques, like making personas and documenting. Of course there are field to improve: should real users be used, how to document it, how to capture value, how long?

Lessons learned:

- RE should be a set of intertwined activities;
- Do not use fake participation in which the business analyst still decides;
- Do make requirements engineering a participative co-design activity;
- Do create a facilitating role for requirements engineers;
- Do take the type of software into account in deciding on an involvement technique.

Expert 10

Requirements are currently elicited through talking with consultants, focus groups and internal brainstorming sessions. Subsequently, requirements are saved in a central backlog (with a wide range of applications) and organized. Sometimes they are directly transcribed, sometimes described in a consultant's words.

It is determined whether a requirement is relevant for a product, which is usually followed up by functional and technical design. This sometimes happens implicitly, when something is clear. Validation is often also implicit, in preparation for a sprint.

Priority is given after a small user story, requirements are not completely described in detail yet. A sprint that follows the requirements is a black box, although company owners or directors sometimes interfere when they don't agree. Companies often try to prevent this.

The previous activities are not isolated steps, but rather a mix.

Managing different channels and structuring the input often goes wrong. "In practice, they [product managers] are overloaded with ideas, questions, suggestions, requirements, etcetera. If you don't properly structure this, you misjudge it". You want to control the channels that are used. Users like to bypass the usual channels. It helps a lot to identify the right channel, e.g. customer involvement sessions or opening up your requirements database. It would be best to shape all your channels in the right way. In customer involvement sessions, customers hear whether their idea will be implemented or not. You won't hear them the rest of the year. Internally, it's difficult. When internal stakeholders give their opinion, it sometimes bypasses the structured processes.

Priorities are always a fight, for which you can use something like planning poker. A quantified priority is often not explicitly given, but functionality is moved around. The leaner the process, the better. A very lean process can however cause that you lose the overview. Organization is the key then.

The process is dependent on many factors. The size of an organizations influences a lot, as well as the diversity and the number of stakeholders. Another factor is the type of product. Compare the scientific market where everything moves slowly and innovations come from conferences, with banking products, where everybody is a manager and has an opinion.

User Involvement

The role of the user is also dependent on the product type. In general, the customer is more important than the user, because he pays. To solve problems, the end-user is more important. Is it strong to have your own vision (based on what you've seen and heard) as a company. Users are important in the evaluation of this vision. If you don't have this vision, you get a large variety of input, especially with early involvement. For many products it is however still important to elicit input. This is very dependent on the type of product (compare an ERP package with software that logistic drivers use in a car).

Crowdsourcing

There are already many ways to involve end-users. It gives useful, but plentiful information. Filtering already occurs: voting as well as software that is built by the crowd (open-source plugins). It is good to involve people, but it should be useful.

A community needs to emerge from free will, there must be something in it for the stakeholders. This rewards can for example be early access or discounts, which often happens in games. “You have to be open for suggestions. If that’s clear, people will find you”.

“You need to do something with your platform, else it’s worse than having no platform at all”.

Gamification

Gamification should work. Ubuntu uses statuses for people that help other people, that works. The status that you get by helping people can be used in other activities. Helping should be rewarded. However, the more you request from users, the more it feels like working for which you earn money. Naturally, users come to you because they have a pain, that has nothing to do with statuses. There should be useful techniques though and companies are probably going to develop those techniques.

Negotiation and prioritization is probably a good way, since it is less effort for users. Coming up with ideas is hard. Spreading ideas of others is a challenge. Subsequently you can motivate other stakeholders to give their opinion about those ideas. You have a number of roles (suggesters, criticisers, refiners). That is something you can gamify. You have to deliver things mainly ready-made. Suggesting things yourself [as a software company] is more dangerous, because it shows low self-confidence.

You should not steer people by using moderators. People should be free, which has the risk of leading to a lot of data. Voting would be a way of structuring.

Bias is something you should always take into account. A group of users could be ‘isolated’ into sessions, which is presumably already happening. This would lead to a sub-community.

Lessons Learned:

- Do not isolate the steps in RE, but create a mixed approach;
- Do not request users to perform difficult work;
- Do not steer involved users;
- Do not suggest your own ideas;
- Do structure the stakeholder (internal & external) input channels;
- Do identify the right channel;
- Do use a lean prioritization technique, while assuring organization;
- Do create your own product vision;
- Do evaluation of your vision with users;
- Do take your situational factors into account, this determines the techniques to use;
- Do communicate your openness to suggestions;
- Do provide feedback on the user involvement (!);
- Do reward help;
- Do provide an incentive to be part of a community;
- Do spread ideas that are created by others;
- Do gamify the existence of different roles;
- Do deliver things ready-made to participators;
- Possibly create subcommunities.

