Factors that Complicate the Selection of Software Requirements
Validating Factors from Literature in an Empirical Study

Hans Schoenmakers[1], Rob Kusters[2,3] and Jos Trienekens[3]
[1]*Software Development Centre, Ricoh Europe, Magistratenlaan 2, 5223 MD 's-Hertogenbosch, The Netherlands*
[2]*Management, Science and Technology, Open Universiteit Nederland, Heerlen, The Netherlands*
[3]*Industrial Engineering & Innovation Sciences, Eindhoven University of Technology, Eindhoven, The Netherlands*

Keywords: Requirements Selection, Prioritization, Software Release Planning, Stakeholders, Stakeholder Salience, Systematic Literature Review.

Abstract: In market-driven software product development, new features may be added to the software, based on a collection of candidate requirements. Selecting requirements however is difficult. Despite all work done on this problem, known as the next release problem, what is missing, is a comprehensive overview of the factors that complicate selecting software requirements. This paper aims at getting such overview. The authors performed a systematic literature review, searching for occurrences in the literature where a causal relation was suggested between certain conditions and the difficulty of selecting software requirements. Analyzing 544 papers led to 156 findings. Clustering them resulted in 33 complicating factors that were classified in eight groups. The complicating factors were validated in semi-structured interviews with twelve experts from three different industrial organizations. These interviews consisted of questions about participant's experiences with the complicating factors, and of questions how these factors complicated selecting requirements. The results aid in getting a better understanding of the complexity of selecting requirements.

1 INTRODUCTION

In market-driven software product development one frequently sees that new features are added to the software, based on a collection of yet unfulfilled requirements (Fogelström et al., 2009; Regnell and Brinkkemper, 2005). Selecting requirements for the next or, more in general, future software release, is a necessary but difficult task (Wnuk et al., 2015; Bagnall et al., 2001; Li et al., 2017). Given a collection of requirements, the organization is faced with the challenge of making a selection of the ones with highest priority, and skip or postpone the rest (Ruhe et al., 2002); a major theme in the area of release planning (Ruhe, 2005; Greer and Ruhe, 2004). In a context with many requirements, a multitude of stakeholders of different salience (Mitchell et al., 1997), multiple decision makers and changing circumstances, selecting requirements is difficult; a difficulty, known as the *next release problem* (Bagnall et al., 2001). Considering that developing the right product is essential and that developing software products uses scarce resources (Kabbedijk et al., 2010; Berntsson Svensson, 2011), it is evident that selecting the right requirements is important. This is even more the case because the effects of made decisions in selecting requirements will be felt (much) later and wasted effort cannot be undone. The lack of understanding which requirements to select justifies the question: what makes selecting requirements difficult? (Wohlin and Aurum, 2005; Barney et al., 2009). This problem has been addressed by different authors from different perspectives. A comprehensive overview of the factors that complicate selecting software requirements in a context of release planning, however, is missing. This paper documents a research that, starting with a systematic literature review (SLR), aims at getting such overview. It aids in getting a better understanding and may be used in initiatives to improve the practice of selecting requirements. Selecting requirements is done for various reasons. First, the collection of candidate requirements is usually much larger than what can be accomplished with the available resources (Li et al., 2007; Berander and Andrews, 2005; Sivzattian and Nuseibeh, 2001). Second, the requirements should not necessarily be developed in just one release. There has been a shift from developing infrequent releases, covering many require-

Schoenmakers, H., Kusters, R. and Trienekens, J.
Factors that Complicate the Selection of Software Requirements - Validating Factors from Literature in an Empirical Study.
DOI: 10.5220/0006836702130220
In *Proceedings of the 13th International Conference on Software Technologies (ICSOFT 2018)*, pages 213-220
ISBN: 978-989-758-320-9

ments, to developing frequent releases with few requirements (Fowler and Highsmith, 2001; Greer and Ruhe, 2004). Agile methods, supporting incremental and iterative development, appear to replace the Waterfall method (Royce, 1970; Racheva et al., 2010; Turk et al., 2014). Third, some requirements should not be developed at all, for example, requirements that have a negative return on investment or requirements that do not comply with the long-term product strategy (Regnell et al., 1998).

The difficulty of selecting the requirements with highest priority is the topic of this paper. In Section 4.1, a research question is formulated: RQ1: "Which factors complicate selecting requirements from a collection of candidate requirements?". In Section 4.3, a research question is formulated to guide the validation of the found factors: RQ2: "Are the found and classified factors experienced in practice, and if so, how do they complicate selecting software requirements?". The remainder of this paper documents the SLR, the classification of complicating factors, and the survey that validates these factors.

2 RELATED WORK

Agile methods, allowing for frequent software releases, covering relatively few requirements have, to some extent, replaced the Waterfall method (Royce, 1970) that relied on infrequent releases of a large number of requirements. One of the consequences of this trend of developing software products in different releases is the selection of requirements that have highest priority, from the collection of candidate requirements. So far, no clear-cut method is available to determine which those requirements are. This section discusses research on this issue, thus identifying the gap that exists between the factors that complicate selecting software requirements, and what is known about those factors. For each research area, main contributions to the aspect of selecting requirements are briefly listed, together with limitations to this aspect. The gap is determined by analyzing these limitations.

STAKEHOLDER THEORY. The role that stakeholders play in the selection of requirements follows from the role that stakeholders play for the organization. This role has been the subject of Stakeholder Theory (Freeman and McVea, 2001). The notion that the relevance of the stakeholders depends on a number of attributes (power, urgency, legitimacy) has made clear that not every stakeholder has the same relevance ('salience') to the organization (Mitchell et al., 1997). For the selection of software requirements, it can be concluded that stakeholder identification and assessing their importance is important. Stakeholder Theory and the concept of stakeholder salience however do not address the difficulty of determining who the stakeholders are nor what their salience is.

PROSPECT THEORY. With Prospect Theory, Kahneman addressed the limitations of Expected Utility Theory (Mongin, 1997); a theory that until then was generally accepted as a normative model of rational choice (Kahneman and Tversky, 1979). Where Expected Utility Theory assumes a rational decision maker, Prospect Theory recognizes that there are circumstances that cause that decision makers make seemingly irrational choices. The theory explains for example why organizations prefer requirements that deliver profit over requirements that avoid loss. Prospect Theory has contributed much to the understanding of some, but not all, of the problems, related to selecting requirements. It does not address issues like quality of requirements, requirements dependencies, the dynamic of market-driven software development, to name a few.

THEORY W: MAKE EVERYONE A WINNER. By introducing Theory W: "make everyone a winner" Boehm takes the standpoint that, by creating win-win situations, "every stakeholder should win" (Boehm and Ross, 1989). Based on this standpoint, Boehm constructed a method, 'WinWin' that takes all stakeholders into consideration. The method has been extended with quantitative methods to allow for better and more objective decisions (Ruhe et al., 2002).

SELECTION CRITERIA. Aurum and Wohlin investigated whether criteria could be defined that would help in prioritizing requirements (Wohlin and Aurum, 2005). They found that organizations have preferences for certain types of requirements. In general, they prefer business and management requirements over system requirements; a seemingly irrational preference. The authors recognize the importance of system requirements but argue that the decisions about such requirements should be handled "within the development and evolution of the software".

RELEASE PLANNING. Unlike the foregoing discussed work, release planning specifically addresses the problem of selecting requirements for the next software release. Release planning has been described as "to decide upon the most promising software release plans while taking into account diverse qualitative and quantitative project data" (Ruhe, 2005). Release planning assumes that software is developed in series of releases with additive functionality, and aims at selecting the right software requirements for the next release. Release planning is characterized as a 'wicked

problem' (Carlshamre, 2002), referring to a class of essentially unsolvable problems (Rittel and Webber, 1973).

Ruhe identifies ten difficulties, related to release planning (Ruhe, 2005): (1) features are not well specified and understood, (2) insufficient stakeholder involvement, (3) change of features and requirements, (4) size and complexity, (5) uncertainty of data, (6) lacking availability of data, (7) constraints —mostly resources, schedule, budget and effort, (8) unclear objectives, (9) efficiency and effectiveness, (10) lacking tooling support.

Khurum et al. identify six factors that, in a context of requirements triage and selection, complicate selecting requirements (Khurum et al., 2012): (1) difficulty in alignment of requirements with long-term business goals, (2) requirements dependencies, (3) difficulty in improving requirements triage decision quality, (4) difficulty in comparing functional and non-functional requirements, (5) creation of product value, (6) difficulty in selection of prioritization techniques.

There is however only a relatively small overlap between the overview of (Ruhe, 2005) and (Khurum et al., 2012), suggesting the existence of more factors.

The discussed research directions have all contributed to the understanding which aspects are relevant for selecting software requirements. Since —with the exception of release planning— they focus only on one aspect of the decision-making process, it is not surprising that they do not attempt to cover all aspects that complicate selecting software requirements.

To the best of the authors' knowledge, no comprehensive overview exists of factors that complicate selecting software requirements. The systematic literature review, the subsequent classification and validation resulted in such an overview, attempting to fill this gap.

3 METHODOLOGY

In order to obtain a reasonably complete and valid set of factors, a three-step strategy was chosen: (1) Perform a systematic literature review to get a good picture of what was covered by the literature on the topic of selecting software requirements. This part would result in finding occurrences in the literature where complications of selecting requirements were addressed. (2) Classify the findings —many, unsorted, with overlaps and duplicates— in order to obtain disjunct groups of complicating factors. (3) Perform a survey to determine if the found factors can be observed in practice.

The execution of the methodology and the results are addressed in the Results section.

4 RESULTS

4.1 The Systematic Literature Review

The SLR was performed conform Kitchenham's guideline (Kitchenham and Charters, 2007), and comprised the following tasks: (1) identifying the need for review, (2) developing a review protocol, (3) searching and analyzing promising papers, (4) defining and monitoring the stop condition.

IDENTIFYING THE NEED FOR REVIEW. Since an SLR may not be needed nor justified if one has been done before (Kitchenham and Charters, 2007), the literature was searched to find if such review had been performed before. Despite the vast number of publications related to the next release problem (e.g. (Wohlin and Aurum, 2005; Ruhe, 2005; Bagnall et al., 2001)), none of them qualified as an SLR —nor pretended to be one.

DEVELOPING A REVIEW PROTOCOL. A review protocol was compiled that would guide the SLR and assure its validity and reliability, covering following elements: (1) formulating a research question, (2) defining a search strategy, compiling a list of search terms and deciding about resources, (3) study selection criteria, procedures, (4) Study quality assessment, (5) data extraction strategy.
Research question: The goal of the SLR was formalized with following research question:

> RQ1: "Which factors complicate selecting requirements from a collection of candidate requirements?"

Interpretation of the used terms: (1) the word 'collection' is used to emphasize that no particular structure is assumed, nor ordering or the absence of duplicates. The only made assumption is that requirements are collected somehow. (2) with 'selecting from a collection', a process is assumed in which all members of the collection are considered, and are selected or not selected —for a coming software release. No assumption is made that this selection is done in one step. It could be done in multiple steps, for example via early requirements triage (Khurum et al., 2012). (3) IEEE defines requirement as "(3.1) A condition or capability needed by a user to solve a problem or achieve an objective. (3.2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other

formally imposed documents. (3.3) A documented representation of a condition or capability as in (3.1) or (3.2)". In the applicable release planning context, the requirements may be somewhere in the transition from being an abstract concept (interpretation 3.2, in the foregoing) and being a documented representation (interpretation 3.3).

Search strategy, search terms, resources: searching for literature was done in multiple ways: (1) by searching with Google Scholar, (2) by following references, encountered in found literature, (3) by searching for particular keywords, encountered in found literature.

Searching literature implies having to construct search queries. It was argued that, due to the wide scope of the problem, searching with terms, derived from 'selecting software requirements' would lead to an abundance of irrelevant results. To overcome this difficulty, a *conceptual model* (CM) was constructed, using terms from the problem context, augmented by terms, derived from related work (see Section 2), in particular from (Mitchell et al., 1997; Freeman and Reed, 1983; van de Weerd et al., 2006). The CM was implemented as information model, in the style of the NIAM natural language information method (Halpin, 1998). Search queries were created from keywords that followed from the resulting interconnected set of concepts.

References, found in literature items that addressed particular complications, extended the set of literature to be reviewed. Likewise, certain terms, found in the literature (like WinWin) led to new search terms, leading to relevant literature on the topic.

Study selection criteria, procedures: in order to judge whether found literature should be included in the review, the following study selection criteria were used: (1) discussing the theory or practice of selecting software requirements, but avoiding papers that had a very wide context, like requirements engineering, (2) articles, conference papers, theses, but no books (risk of being outdated), (3) not from tool vendors, expecting them to be less objective. Publication date was not used as selection criterion, arguing that dated publications would be recognized —and rejected— at analysis time.

Study quality assessment: Found papers were included in the set to be analyzed, 'on face value'. When in doubt, it was decided to "err on the side of inclusiveness" (Okoli and Schabram, 2010), trusting that reading and analyzing the papers would lead to rejecting material of lesser quality.

Data extraction strategy: The review was split in two parts: a part, in which literature was searched, using the constructed queries, and another part in which the found literature —that was stored in a document repository— was analyzed. Care was taken that the repository was free of duplicates.

SEARCHING AND ANALYZING PROMISING PAPERS. It was decided to search literature with Google Scholar (GS), a search engine, intended for searching scholarly literature. Google Scholar was validated as literature search tool, addressing three anticipated threats to validity: (1) GS does not find enough relevant material, (2) GS finds low-quality literature, (3) GS finds so much low-quality literature that the returned results obscure the high-quality material.

Research shows that GS finds enough literature (DeGraff et al., 2013; Shariff et al., 2013; Beel and Gipp, 2009; Gehanno et al., 2013; Falagas et al., 2008). Some authors state that GS does not discriminate much between older and newer literature (Beel and Gipp, 2009) or state that GS is somewhat biased to literature in the English language (Neuhaus et al., 2006). These concerns are less relevant for this literature review. (1) obsolete material will be recognized 'on face value', and not be added to the literature repository and (2) only English literature will be searched. The threat that GS also returns low-quality literature is a reality (Gray et al., 2012; Noruzi, 2005). This threat was mitigated by verifying the quality of found literature before adding it to the literature repository. The third threat is real too: if much of the returned literature has low quality, the high-quality material will be obscured by the low-quality literature. Investigations that have been done on GS however, do not suggest that this is the case.

DEFINING AND MONITORING STOP CONDITION. Even if it would be possible to review all existing literature, there is no need to do so. Continued searches will provide more papers on the topic and return more findings, but on one moment, new findings do not lead to new insights (Levy and Ellis, 2006). Eventually it takes too long to find anything new. This means that one has to define a 'stop criterion' to end the search not too early and not too late. It was decided to stop when after N consecutive searches no material was found that brought new insights —N being a number between five and ten.

The SLR was performed by one person. Searching resulted in 544 papers. These papers were analyzed, resulting in 156 findings, that is: passages in the analyzed literature, indicating how some condition complicates selecting software requirements. A finding would consist of (1) the found fragment of text, (2) a descriptive label that summarized —as one-liner— the content of the finding, (3) identification of the lite-

rature item. The label served as an aid to classify the findings.

4.2 Classifying the Findings

Reviewing the literature led to statements, pointing in the direction of factors that complicate selecting software requirements. Statements were unique or duplicated or overlapped others to some extent. In order to arrive at meaningful complicating factors, they had to be processed somehow. It was decided to cluster them, using 'similarity in complicating selecting software requirements' as the clustering criterion. Since the findings were statements in natural language, no automated algorithm could be used. Instead, clustering would rely on human judgment. It was decided to cluster by card sorting with Metaplan, a technique, suitable for this type of data, allowing for group-wise, consensus-based decision making (Capra, 2005; Dulle and Rauch, 2014). More specific: open card sorting since initially there would be no clusters. The clusters, representing complicating factors would emerge while clustering.

The number of found complicating factors was such that grouping them was needed to make the results comprehensible. It was decided to use the technique that was also used for the clustering activity, and also use 'similarity' as classification criterion, arguing that this criterion provided most insight.

Clustering the findings led to 33 complicating factors. Classification through a Metaplan session (Dulle and Rauch, 2014), performed with three participants, experienced in requirements engineering and management and/or methodology, led to a grouping of factors. Table 1 shows the resulting grouping, with the group as leftmost column, and complicating factor as rightmost column.

4.3 Validating the Complicating Factors

By determining whether the found factors are indeed experienced in practice, a survey was performed, thus validating the results of the SLR. The activity was formalized with following research question:

> RQ2: "Are the found and classified factors observed in practice, and if so, how do they complicate selecting software requirements?"

Three software product developing organizations took part in this activity, each organization being represented by a few participants, having different roles, and somehow involved in the selection of software requirements. The organizations were chosen, taking into account their expected level of professionalism with regards to dealing with requirements. Information was gathered through semi-structured interviews, thus combining the advantages of a structured approach and retrieving information. It was made clear to the participant that the questions were about the process of selecting requirements; not of requirements engineering in general. At the start of an interview, unbiased information was received with an open, explorative question: "Which problems do you experience with selecting requirements?". Purpose of this question was twofold: (1) finding if there were factors that were so pertinent that participant could immediately name them, (2) additional check of the completeness of the set of complicating factors. The interview continued by addressing each complicating factor, asking how often a complication was experienced (values: 'never', 'almost never', 'sometimes', 'often', 'practically always'), and how severe (values: 'not at all a problem', 'somewhat problematic', 'serious problem'). It was emphasized to the participants that their answers should be based on own experiences; not on opinions. Additional questions were asked to gain a deeper understanding of the problem: "How does this complicate selecting requirements?", find out if solutions or work-arounds were known, or verify if the participant personally experienced the factor. It was also asked if the organization had solutions or work-arounds to mitigate the complication. The order of the factors was randomized to avoid bias, caused by factors that depend on this order —like tiredness of the participant or interviewer.

Twelve participants were interviewed in semi-structured interviews —each one taking around two hours. The results of the interview were sent to the participant for correction, sometimes accompanied with questions for clarification. Interpretation of the data: for each factor, the frequency and severity values of all participants were plotted in a matrix. The rows represent the severity of the complicating factor, ranging from 'Not at all a problem' to 'Serious problem'. When a participant indicated that a factor was never experienced, the participant was not asked how serious he considered the factor, arguing that asking this, would be asking for opinions. The columns represent the frequency of occurrence of the factor, ranging from 'Never' to 'Practically always'. The cells of the matrix hold the participants, together with their organization and role. See Figure 1 for an example of such a matrix. A 'gray' area was defined of (frequency x severity) values that were considered complicating. First criterion to decide if a factor was complicating: enough scores in the gray area, 'enough' being chosen as 6. Second criterion: enough support from the statements in the comments,

made by the participants. The natural language aspect prohibited choosing a measurable criterion. It was observed however that the level of a score did not always match the textual comments. Some participants, for example, gave low scores to severity, but in their comments they named severe complications, and added that "such complications are just part of the job". The contrary also happened: high scores, but hardly any comment, supporting these scores. Therefore the decision about the 'level of complicatedness' ('+': 'Definitely complicating', '≈': 'Likely to be complicating', '-': 'Unlikely to be very complicating') was made by independent review of the scores and the comments by the authors, followed-up with a discussion to reach consensus.

Most —but not all— factors were recognized and experienced by the participants. Table 1 indicates, for each factor, whether the factor could be validated. It holds the level of complicatedness (column 2) and number of scores in the gray are (column 3). The initial, open question "Which problems do you experience with selecting requirements?" did not lead to new complicating factors. This provides support for the claim that the set of complicating factors is reasonable complete.

Some organizations had found work-arounds or solutions for some of the complicating factors. For example, 'A large number of requirements to select from' was not experienced by one organization because it used a way of grouping requirements, thus avoiding a large list of small, detailed requirements.

	Never	Almost never	Sometimes	Often	Practically always
N/A	B_{1t} L_{3t} A_{2t}				
Not at all a problem		K_{1m}	E_{1t}	I_{1e} J_{Et}	D_{1e}
Somewhat problematic		G_{Pt}		**F2t**	**C_{3e} H_{2t}**
Serious problem					

Factor 10. Having to plan further than just the current release

LEGEND First letter (capital) represents participant
subscript
1 1:Case1 2:Case2 3:Case3 P:Pilot case E:Expert interview
2 t:team member m:manager e:external stakeholder (representative)

Figure 1: Example of a matrix with participants' scores.

5 CONCLUSIONS AND DISCUSSION

The principal results consist of the found factors that complicate selecting software requirements. The classification activity resulted in a logical structure, thus aiding to the comprehensibility of the results. The follow-up survey confirmed most of the factors: 'Definitely complicating': 28, 'Likely to be complicating': 4, 'Unlikely to be very complicating': 1. One cannot exclude that the factors in the latter two categories could be complicating in other organizations.

The interviews did not hold any indication of overlooked complicating factors. Therefore it is concluded that the research resulted in a valid and substantially complete set of complicating factors. Missed factors, if any, are not expected to be the most serious ones. The results of the literature review provide insight and may be used in initiatives to find solutions to some of the identified problems.

Table 1: Classified complicating factors and survey results.

CLASS	R	S	COMPLICATING FACTOR
REQUIREMENTS	+	8	A large number of requirements to select from
	+	4	Requirements, holding a large amount of information
	+11		Requirements that are not explicit or not precise
	+	7	Requirements with different levels of abstraction
REQUIREMENTS ENGINEERING	+	5	Lack of structure in the requirements engineering process
	+	5	Difficulty in finding a balance between precision/completeness of requirements and the effort to create them
SELECTION PROCESS	+	9	Difficulty of getting the right information required for prioritization
	≈	5	Unavailability of suitable tooling to support the selection
	+	5	Lack of understanding the goals of the organization
	≈	8	Lack of trust that decision makers may have in the results of prioritization
	+10		Lacking availability of resources required for implementing particular requirements
	+12		Difficulty of estimating the effort needed to meet a
	+	6	Time stress in the process of selecting requirements
	+	8	A time-consuming requirements selection process
STAKEHOLDER BALANCING	+10		Difficulty of balancing conflicting stakes of stakeholders and resolving resulting conflicts
	+	7	A large number of stakeholders involved in the requirements selection process
	+12		A different degree of importance that different stakeholders have for the organization
STAKEHOLDER COMPLEXITY	+	9	Difficulty in accessing or involving the relevant
	-	7	Stakeholders in a subcontractor role, insufficiently understanding the organization's stakes
	≈	2	Unpredictable behavior of stakeholders within stakeholder groups
	≈	9	Lack of homogeneity within stakeholder groups
	+	8	Poor personal relationships between decision makers and stakeholders
STAKEHOLDER POOR UNDERSTANDING	+10		Lack of understanding how individual requirements contribute to stakeholders' needs
	+12		Lack of understanding what the stakeholders want
	+	6	Difficulty in identifying who the stakeholders are, over different releases
	+	8	Lack of communication between decision makers and stakeholders
ARCHITECTURE	+	6	Requirements that cannot be selected unless other requirements are also fulfilled (or not fulfilled)
	+	7	Dependencies with other software product family
	+	8	Dependencies with products from competitors
	+	3	Having to plan further than just the current release
	+10		Different requirements for different market segments, spreading around the world
CHANGING ENVIRONMENT	+	7	Evolving goals, goal priorities, plans and mission of the organization
	+	9	Volatility of requirements

Legend (R) +Definitely complicating ≈Likely to be complicating -Unlikely to be very complicating. (S) The number of scores in the gray area (see Figure 1).

Looking back at the work of others (see Section 2), one can see that the results confirm all difficulties, identified by (Ruhe, 2005) and (Khurum et al., 2012). The number of found complicating factors provides support for the claim that release planning is a 'wicked problem' (Rittel and Webber, 1973).

The conceptual model, discussed in Section 4.1, helped in creating effective search queries. Additionally, it turned out to be a useful tool to determine whether found literature fitted within the scope. If the literature item as a whole fitted within the boundaries of the model, the paper was used and analyzed. If it didn't, the paper was not considered any further.

The survey validated most factors. The occurrence of the factors appeared to depend on the particular organization. Some factors that were experienced as problematic in one organization, were not experienced as such in another organization. It remains to be investigated whether these differences are situational or are caused by a different level of process maturity. They are an indication however that even the 'not validated factors' may be complicating for certain organizations.

WEAKNESSES OF THE SLR. Reviewing and interpreting the literature was done by the first author only. Although the results were reviewed and discussed by the other authors, the quality of interpretation would have benefitted from a review by multiple reviewers.

WEAKNESSES OF THE CLASSIFICATION. The choice of the groups is arbitrary to some extent; others would have selected different labels. It is argued that this is unavoidable, since the material to be classified is in natural language. The well-defined classification criterion mitigated the weakness somewhat.

WEAKNESSES OF THE SURVEY. (1) All participants were working in organizations in the Netherlands. The results cannot be immediately extrapolated to the practice of selecting requirements in organizations in other parts of the world. (2) Although the results of the interviews were reviewed and corrected by the participants, the interviews themselves were not recorded. Recording them would have benefitted the accuracy of the results.

Despite the observed weaknesses, the research has resulted in a clearer picture of the factors that complicate selecting requirements, and that the literature review has contributed to a better understanding of the problem.

REFERENCES

Bagnall, A. J., Rayward-Smith, V. J., and Whittley, I. M. (2001). The next release problem. *Information and software technology*, 43(14):883–890.

Barney, S., Wohlin, C., and Aurum, A. (2009). Balancing software product investments. In *Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd International Symposium on*, pages 257–268. IEEE.

Beel, J. and Gipp, B. (2009). Google scholar's ranking algorithm: The impact of articles' age (an empirical study). In *Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on*, pages 160–164. IEEE.

Berander, P. and Andrews, A. (2005). Requirements prioritization. In *Engineering and managing software requirements*, chapter Engineering and managing software requirements, pages 69–94. Springer.

Berntsson Svensson, R. (2011). *Supporting Release Planning of Quality Requirements: The Quality Performance Model*. Lund University.

Boehm, B. W. and Ross, R. (1989). Theory-w software project management principles and examples. *Software Engineering, IEEE Transactions on*, 15(7):902–916. 7.

Capra, M. G. (2005). Factor analysis of card sort data: an alternative to hierarchical cluster analysis. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 49, pages 691–695. SAGE Publications.

Carlshamre, P. (2002). Release planning in market-driven software product development: Provoking an understanding. *Requirements engineering*, 7(3):139–151.

DeGraff, J. V., DeGraff, N., and Romesburg, H. C. (2013). Literature searches with google scholar: Knowing what you are and are not getting. *GSA Today*, 23(10).

Dulle, M. and Rauch, F. (2014). Toolbox for school-community collaboration for sustainable development. page 66.

Falagas, M. E., Pitsouni, E. I., Malietzis, G. A., and Pappas, G. (2008). Comparison of pubmed, scopus, web of science, and google scholar: strengths and weaknesses. *The FASEB journal*, 22(2):338–342.

Fogelström, N. D., Barney, S., Aurum, A., and Hederstierna, A. (2009). When product managers gamble with requirements: Attitudes to value and risk. In *Requirements engineering: Foundation for software quality*, chapter Requirements engineering: Foundation for software quality, pages 1–15. Springer.

Fowler, M. and Highsmith, J. (2001). The agile manifesto. *Software Development*, 9(8):28–35.

Freeman, R. E. and McVea, J. (2001). A stakeholder approach to strategic management.

Freeman, R. E. and Reed, D. L. (1983). Stockholders and shareholders: a new perspective on corporate governance. *California Management Review*, 25(3):88–106.

Gehanno, J.-F., Rollin, L., and Darmoni, S. (2013). Is the coverage of google scholar enough to be used alone

for systematic reviews. *BMC medical informatics and decision making*, 13(1):7.

Gray, J. E., Hamilton, M. C., Hauser, A., Janz, M. M., Peters, J. P., and Taggart, F. (2012). Scholarish: Google scholar and its value to the sciences. *Issues in Science and Technology Librarianship*, 70(Summer).

Greer, D. and Ruhe, G. (2004). Software release planning: an evolutionary and iterative approach. *Information and Software Technology*, 46(4):243–253.

Halpin, T. (1998). Object-role modeling (orm/niam). In *Handbook on architectures of information systems*, pages 81–103. Springer.

Kabbedijk, J., Wnuk, K., Regnell, B., Brinkkemper, S., et al. (2010). What decision characteristics influence decision making in market-driven large-scale software product line development? *Hildesheimer Informatik-Berichte*, 2010:42–53.

Kahneman, D. and Tversky, A. (1979). Prospect theory: An analysis of decision under risk. *Econometrica: Journal of the Econometric Society*, pages 263–291.

Khurum, M., Uppalapati, N., and Veeramachaneni, R. C. (2012). Software requirements triage and selection: state-of-the-art and state-of-practice. In *Software Engineering Conference (APSEC), 2012 19th Asia-Pacific*, volume 1, pages 416–421. IEEE.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. *Engineering*, 2(EBSE 2007-001). EBSE 2007-001.

Levy, Y. and Ellis, T. J. (2006). Towards a framework of literature review process in support of information systems research. In *Proceedings of the 2006 Informing Science and IT Education Joint Conference*, volume 26.

Li, C., Van Den Akker, J., Brinkkemper, S., and Diepen, G. (2007). Integrated requirement selection and scheduling for the release planning of a software product. In *Requirements Engineering: Foundation for Software Quality*, chapter Requirements Engineering: Foundation for Software Quality, pages 93–108. Springer.

Li, L., Harman, M., Wu, F., and Zhang, Y. (2017). The value of exact analysis in requirements selection. *IEEE Transactions on Software Engineering*, 43(6):580–596.

Mitchell, R. K., Agle, B. R., and Wood, D. J. (1997). Toward a theory of stakeholder identification and salience: Defining the principle of who and what really counts. *Academy of management review*, 22(4):853–886.

Mongin, P. (1997). Expected utility theory. *Handbook of economic methodology*, 342350.

Neuhaus, C., Neuhaus, E., Asher, A., and Wrede, C. (2006). The depth and breadth of google scholar: An empirical study. *portal: Libraries and the Academy*, 6(2):127–141.

Noruzi, A. (2005). Google scholar: The new generation of citation indexes. *Libri*, 55(4):170–180.

Okoli, C. and Schabram, K. (2010). A guide to conducting a systematic literature review of information systems research. *Available at SSRN 1954824*.

Racheva, Z., Daneva, M., Sikkel, K., Herrmann, A., and Wieringa, R. (2010). Do we know enough about requirements prioritization in agile projects: insights from a case study. In *Requirements Engineering Conference (RE), 2010 18th IEEE International*, pages 147–156. IEEE.

Regnell, B., Beremark, P., and Eklundh, O. (1998). A market-driven requirements engineering process: results from an industrial process improvement programme. *Requirements engineering*, 3(2):121–129.

Regnell, B. and Brinkkemper, S. (2005). Market-driven requirements engineering for software products. In *Engineering and managing software requirements*, chapter Engineering and managing software requirements, pages 287–308. Springer.

Rittel, H. W. and Webber, M. M. (1973). Dilemmas in a general theory of planning. *Policy sciences*, 4(2):155–169.

Royce, W. W. (1970). Managing the development of large software systems. In *proceedings of IEEE WESCON*, volume 26, pages 328–388. Los Angeles.

Ruhe, G. (2005). Software release planning. *Handbook of software engineering and knowledge engineering*, 3:365–394.

Ruhe, G., Eberlein, A., and Pfahl, D. (2002). Quantitative winwin: a new method for decision support in requirements negotiation. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pages 159–166. ACM.

Shariff, S. Z., Bejaimal, S. A., Sontrop, J. M., Iansavichus, A. V., Haynes, R. B., Weir, M. A., and Garg, A. X. (2013). Retrieving clinical evidence: a comparison of pubmed and google scholar for quick clinical searches. *Journal of medical Internet research*, 15(8).

Sivzattian, S. and Nuseibeh, B. (2001). Linking the selection of requirements to market value: A portfolio-based approach. In *7th International Workshop on Requirements Engineering: Foundation for Software Quality. Interlaken, Switzerland*. Citeseer.

Turk, D., France, R., and Rumpe, B. (2014). Assumptions underlying agile software development processes. *arXiv preprint arXiv:1409.6610*.

van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., and Bijlsma, L. (2006). On the creation of a reference framework for software product management: Validation and tool support. In *Software Product Management, 2006. IWSPM'06. International Workshop on*, pages 3–12. IEEE.

Wnuk, K., Kabbedijk, J., Brinkkemper, S., Regnell, B., and Callele, D. (2015). Exploring factors affecting decision outcome and lead time in large-scale requirements engineering. *Journal of Software: Evolution and Process*, 27(9):647–673.

Wohlin, C. and Aurum, A. (2005). What is important when deciding to include a software requirement into a project or release. In *International Symposium on Empiricial Software Engineering*, volume 186. IEEE.