2015 41st Euromicro Conference on Software Engineering and Advanced Applications

A Mapping Study on Requirements Engineering in Agile Software Development

Ville T. Heikkilä*†, Casper Lassenius†, Daniela Damian*, Maria Paasivaara†
*Department of Computer Science, University of Victoria, Victoria, B.C., Canda
Email: {vtheikki, danielad}@cs.uvic.ca
†Department of Computer Science, Aalto University, Helsinki, Finland
Email: {ville.t.heikkila, maria.paasivaara, casper.lassenius}@aalto.fi

***Abstract*—Agile software development (ASD) methods have gained popularity in the industry and been the subject of an increasing amount of academic research. Although requirements engineering (RE) in ASD has been studied, the overall understanding of RE in ASD as a phenomenon is still weak. We conducted a mapping study of RE in ASD to review the scientific literature. 28 articles on the topic were identified and analyzed. The results indicate that the definition of agile RE is vague. The proposed benefits from agile RE included lower process overheads, a better requirements understanding, a reduced tendency to overallocate development resources, responsiveness to change, rapid delivery of value, and improved customer relationships. The problematic areas of agile RE were the use of customer representatives, the user story requirements format, the prioritization of requirements, growing technical debt, tacit requirements knowledge, and imprecise effort estimation. We also report proposed solutions to the identified problems.**

***Keywords*-requirements engineering, agile development, scrum, mapping study**

I. Introduction

Traditional requirements engineering (RE) approaches, sometimes known as up-front RE approaches, are based on a sequential execution of the activities of elicitation, analysis, specification, verification and validation of requirements [6], [S7]. In such approaches the communication between the requirements engineering process and the software construction process is mostly based on formal documents [6].

Agile software development (ASD) methods, such as Scrum [20], [21] and Extreme Programming [2], take a different approach to requirements engineering and communication. Although the traditional approaches allow for iteration between the steps of the process [6], agile methods are based on the rapid iteration of the whole software development process from the elicitation of the requirements to the release of a new version of the software [2], [20], [21].

Although ASD methods have been intensely studied [7], [8], [10], [12], [14], [17], [22], [23], [25], research on RE in ASD has not been previously mapped in order to identify strong areas of knowledge and gaps in knowledge.

The overall research questions we aim to answer are the following:

1) What has been researched regarding requirements engineering in an agile context?
2) What are the reported key benefits of agile requirements engineering?
3) What are the reported problems and corresponding solutions related to agile requirements engineering?

We answer the research questions by conducting a mapping study on research literature of RE in ASD. The rest of the article is structured as follows: Section II provides background information on the traditional RE process, RE in ASD and previous literature reviews and mapping studies. Section III presents the mapping study protocol and data extraction. Section IV presents the results of the mapping study and Section V discusses the results and the limitations of the study. Finally, Section VI concludes the article and provides directions for future work.

II. Background

This section provides background information. First, the traditional RE process is described in order to define fundamental RE terminology. Second, the normative ASD literature is reviewed and an overview of RE in the literature is presented in order to provide a basic understanding of RE in ASD. Third, previous secondary studies on ASD are reviewed briefly.

A. The traditional requirements engineering process

Traditional requirements engineering literature views the RE process as a set of sequential activities [6], [16]. Iteration between the activities is possible, if it is necessary. The process begins with *requirements elicitation*, where the initial information regarding the requirements and the context are gathered. Then, during *requirements analysis*, the information gathered is analysed in order to create an understanding of the requirements. The next activitiy in the sequence is the *requirements specification*, where the understanding of the requirements is turned into precise specifications of system behaviour. The actual construction of the software system based on the specifications is considered out of the scope of the RE process. *Requirements validation* supports the three other activities by identifying and correcting errors in the requirements. *Requirements prioritization* supports requirements elicitation and analysis by identifying the most valuable requirements [4]. Finally, the *system validation* validates the output of the software construction by comparing it to the requirements [16].

B. Requirements engineering in agile software development

Requirements engineering in agile development is informal and based on the skills and knowledge of individuals [11]. Although many different agile methods have been proposed [12], Scrum [9], [20], [21], Extreme Programming (XP) [2] and their derivatives are the most popular and relevant at the time of writing [24], [26].

In Scrum [9], [20], [21], a single person in the role of a product owner (PO) is responsible for requirements elicitation and for requirements prioritization. Requirements in Scrum reside in a product backlog, which is a prioritized list of all work items imagined for the software, which also can include technical improvements. The work items in the product backlog are called backlog items. Only the product owner can add new items to the backlog. The product owner works with a development team of five to nine cross-functional software developers. The PO and the development team conduct requirements analysis, requirements specification and requirements validation informally and in collaboration. In the beginning of each two to four-week development iteration, based on the development team's previous performance, the PO and the development team decide which backlog items are implemented during the iteration. At the end of the iteration, the system validation is done by the PO, who reviews the new system behaviour. [9], [20], [21]

Extreme Programming (XP) concentrates on software construction and there is little guidance for requirements engineering [2]. The actors in XP are an on-site customer and a team of three to twenty developers. The main RE practice in XP is the planning game, which begins with an on-site customer writing requirements. Thereafter, the on-site customer and developers decide which of the requirements are to be implemented during the following two-week development iteration. The implemented requirements are also validated by the on-site customer. [2]

C. Literature reviews and mapping studies of agile software development

The terms *systematic literature review* and *mapping study* have been used interchangeably in many articles. Below, the terms are used following the definitions by Kitchenham and Charters [18] regardless of how the articles characterize themselves. By their definitions, mapping studies provide broad overviews of the research done regarding a topic, while literature reviews synthesize or meta-analyse the results based on a more narrow research question.

The most reviewed subject area has been the usability or user centred design (UCD) in the ASD context. Sohaib and Khan [23] conducted a mapping study of integrating usability and ASD. They identified a conflict between up-front usability requirements engineering processes and ASD. Jurca et al. [17] performed a mapping study of integrating agile and UCD. They disseminated challenges in the integration of ASD and UCD, and organizational recommendations for the integration. The RE related recommendations included the close collaboration between usability designers and developers, using lightweight usability methods with XP, and using lightweight usability requirements engineering methods with ASD. Salah et al. [19] reviewed literature on challenges in integrating ASD and UCD. They identified two RE-related challenges, which were the lack of time for up-front usability requirements activities and the lack of usability requirements documentation. They identified better collaboration between software developers and usability specialists as a mitigation strategy to the former challenge, and additional usability documentation as a mitigation strategy to the latter challenge.

Two reviews have been done on agile methods in global software engineering (GSD). Hossain et al. [14] reviewed research on challenges and mitigation strategies related to Scrum in the GSD context. The only RE related challenge they identified was the lack of a proper backlog tool, which could be mitigated by a shared electronic backlog tool. They also found that additional requirements documentation may mitigate challenges in team collaboration. Britto et al. [7] performed a mapping study of effort estimation methods in the agile GSD context. They identified different cost drivers and size metrics that have been studied previously.

The scope and focus of the rest of the previous reviews and mapping studies varies widely. Dybå and Dingsøyr [12] reviewed empirical studies of ASD. They found that agile RE was more flexible and reactive than a traditional, incremental approach. Panagiotis and Stamelos [22] reviewed studies on the effect of agile software construction practices on software quality. Only RE related result was that the planning game practice was found to lead to higher quality. Diebold and Dahlem [10] conducted mapping study of the use of ASD practices in the software industry. Their RE related findings were that customer involvement practices were often used only partially or adapted from the normative agile practices, and that planning meeting and evolving and hierarchical specifications were commonly used practices. Usman et al. [25] performed a mapping study of effort estimation in ASD and describe the frequency of use of different estimation techniques that are employed in an Agile RE process. Inayat et al. [15] conducted a systematic literature review on how traditional RE challenges were resolved using agile RE. They found that agile RE resolved issues related to communication, overscoping, requirements validation, requirements documentation and rare customer involvement.

With the exception of Inayat et al. [15], agile RE has been a tangential or incidental subject in the previous reviews and mapping studies. Although the subject area of their study is alike the subject area of our study, the goals of our study and theirs differ. Our goal is to map the subject area of RE in ASD and identify gaps in the scientific knowledge, while they conducted a systematic literature review in order to answer a specific research question. Based on the previous mapping studies and literature reviews, there is clearly a need for a mapping study of agile RE in order to identify gaps in the literature and prospective research topics.

III. Methodology

The study was conducted as a mapping study [18]. Although ASD research has been previously reviewed, it is not evident how much RE specific research there has been, and a mapping study was considered the most appropriate method at this stage.

The mapping study was conducted using the Elsevier Scopus abstracts database. The search was executed in September 2014. The search string was the following: `TITLE-ABS-KEY(("requirements analysis" OR "requirements engineering") AND (agile OR scrum)) AND NOT KEY("agile manufacturing")`[1]. The search produced 241 results of which 46 were not labelled as journal articles or conference proceedings and were excluded. Furthermore, eight articles not written in English were excluded.

The titles and abstracts of the remaining 187 articles were read and articles were excluded from further analysis based on the following exclusion criteria:

1) It was not possible to gain access to the abstract of the article.
2) The document was not a research article (but a tutorial introduction or a poster, for example).
3) The subject area of the article was not software engineering.
4) The main subject of the article was not requirements engineering/analysis in agile software development context.
5) Article was published in a source that had been identified as a "predatory" [1] or vanity publisher, or that otherwise lacked a full-text peer review process.

In total 123 articles were excluded based on the title and abstract exclusion criteria. The full text of the remaining 65 was acquired, if possible. The articles were read and excluded based on the following exclusion criteria:

1) It was not possible to gain access to the full text of the article.
2) The article was extended by a later article that was also included in the review (i.e. the earlier article was redundant).
3) The main subject of the article was not requirements engineering/analysis in agile software development context.

37 articles were excluded based on the exclusion criteria above. The remaining 28 articles were then read and the article metadata, context, methods and results were extracted from the articles. The metadata, context and methods were summarized. The extracted results were categorized under the following four subject areas: Definition of RE in the agile context, benefits identified in agile RE, problems identified in agile RE and solutions proposed for the aforementioned problems. The benefits, problems and solutions were collated, analysed and categorized under thematic areas identified in the analysis.

[1]"Agile manufacturing" refers to the manufacturing of physical goods.

IV. Results

This section presents the results of the mapping study. The first subsection provides an overview of the included articles. The following subsections present an overview of research of RE in ASD, the benefits the agile RE approach is proposed to create, the problems identified in agile RE and the solutions proposed to those problems.

A. Overview of the included articles

The included 28 articles were published in a wide variety of venues. See Table I for a summary of the venues. Conference proceedings were the most prominent venue ($N = 15, \approx 53\%$) followed by journal articles ($N = 8, \approx 29\%$) and magazine articles ($N = 5, \approx 18\%$).

The context in the majority of articles was unspecified agile development ($N = 20, \approx 71\%$). Scrum was the context in seven articles ($\approx 25\%$) and FDD in one ($\approx 4\%$). See Table II for the context of each article.

The most common article type was the method proposal ($N = 8, \approx 28\%$), followed by the multiple ($N = 6, \approx 21\%$) and single ($N = 5, \approx 18\%$) case study. Three experience reports and three position papers were included ($\approx 11\%$ each). Finally, two papers were method evaluations ($\approx 7\%$) and one paper was a tool evaluation ($\approx 4\%$). See Table III for the type of each article.

Table I
Summary of the publication venues

Type	Venues (number of publications)
Conference proceedings	AREW (2), EmpiRE (1), HICSS (1), ICCIT (1), ICGSE (1), ICWE (1), IWPSE (1), P3S2E (1), RE (3), SERA (1), WET ICE (1), XP (1)
Journal	Inform. Software Tech. (3), Inform. Syst. J. (1), Inform. Technol. Manag. (1), J. Emerg. Tech. Web Int. (1), J. Object Tech. (1), J. Syst. Software (1)
Magazine	IEEE Software (5)

Note: Alphabetically ordered by type and venue name.

B. Overview of the RE in ASD research

The research on RE in ASD has varied greatly. This section presents an overview of the RE in ASD research, including discussion on the nature of agile RE, definition of agile RE and differences and commonalities between agile RE and traditional RE.

Table II
SUMMARY OF AGILE METHODS DISCUSSED IN ARTICLES

Method	Articles	N
Unspecified agile	[S1], [S4], [S6], [S7], [S8], [S10], [S11], [S13], [S14], [S16], [S17], [S19], [S20], [S21], [S22], [S23], [S24], [S25], [S26], [S28]	20
Scrum	[S2], [S3], [S5], [S12], [S15], [S18], [S27]	7
FDD	[S9]	1

Table III
SUMMARY OF TYPES OF ARTICLES

Article type	Articles	N
Multiple case study	[S4], [S5], [S7], [S21], [S22], [S23]	6
Single case study	[S2], [S3], [S10], [S18], [S27]	5
Experience report	[S14], [S24], [S28]	3
Tool evaluation	[S6]	1
Method evaluation	[S15], [S26]	2
Method proposal	[S1], [S9], [S11], [S12], [S13], [S17], [S20], [S25]	8
Position paper	[S8], [S16], [S19]	3

Cao and Ramesh [S4] identified seven practices that were used in 16 US software development organizations they considered constituting agile RE. The practices were face-to-face communication, extreme prioritization, prototyping, iterative RE, reviews and tests, constant planning, and test-driven development. In a later re-analysis of the same data, Ramesh et al. [S23] omitted test-driven development as a RE practice and discussed the differences between traditional RE and agile RE. The found that unlike in traditional RE where requirements were usually prioritized at the beginning of the project, the requirements were continuously re-prioritized in agile RE. They also found that requirements were prioritized in agile RE typically based solely on the customer defined business value, while traditional RE took into account many factors. They claimed that the agile RE prioritization approach was better in terms of meeting customer needs. They concluded that the fundamental assumptions of traditional RE, such as the expectation to have a complete and verifiable set of requirements before the development begins, do not hold in agile software development environments. Furthermore, they found that agile RE processes blend together the traditional RE sub-tasks instead of treating them as separate steps or phases.

Some advocates of traditional RE methods (e.g. traditionalists) have understood the agile development approach as the abolishment of RE in its entirety [S19]. Disregarding the most radical agilistas and traditionalists, the actual dispute between the two sides is not as extreme as first seems. In practice, most traditional RE activities can be or are performed somehow when using agile methods [S20]. Many traditionalists have recognized that change in the requirements is often inevitable [5], [S4], [S25] and that the traditional, lengthy RE process has become uncompetitive since the Internet enabled the rapid delivery of new versions of software [S25].

Paetsch at al. [S20] compared traditional RE methods and different agile development methods in order to identify commonalities and differences between the approaches and to identify the ways agile methods could benefit from traditional RE methods. They found that most traditional RE activities are carried out in some way in agile methods.

Traditionalists and agilists agree that the communication with the users and customers of the software system is critical for the success of the system, but disagree regrading the specific methods. Agilists purport direct, informal face-to-face communication [3], while traditionalists have suggested different kinds of more or less formal communication approaches [S25]. Requirements elicitation, especially in market-driven projects, in agile and traditional RE may employ similar or even the same methods, such as interviews, observations and workshops [S6].

Requirements prioritization is considered important in traditional RE, but crucial in agile RE, since it is one of the key methods for producing value quickly. In Scrum, the product owner is responsible for prioritizing the requirements. Empirical research has found that the practice of requirements prioritization often differs from the normative guidance in Scrum. Racheva et al. [S22] reported from a multiple case study on requirements prioritization in ten bespoke agile development projects and described differences to the normative agile literature. They found that developers often had to interpret the needs of the client and take responsibility for prioritization, which was against the guidance given by Scrum and Extreme Programming, and that a single client representative did not always present the needs of the client accurately. The negative value or loss of value was considered the most important prioritization criterion instead of the (positive) business value recommended by Scrum. Some clients had difficulties accepting high effort estimates based purely on ad hoc estimation methods employed in agile methods. Daneva et al. [S5] studied agile requirements prioritization in large-scale outsourced system projects. They found that the priority decision maker was usually from the client organization, instead of belonging to the provider organization.

C. Benefits of the Agile RE approach

Many articles describe claimed benefits created by the agile RE approaches in comparison to the traditional RE processes. These are listed below and summarized in Table IV.

1) Lower process overheads (B1): Process overheads are lower due to the lack of time-consuming documentation and approval processes [S4], [S20], [S25] and reduced need to rework requirements [S2].

Table IV
BENEFITS FROM AGILE RE AND RELATED ARTICLES

Code	Benefit	Articles
B1	Lower process overheads	[S2], [S4], [S20], [S25]
B2	Improved requirements understanding	[S2], [S3], [S4], [S7], [S18], [S23]
B3	Reduced overburden	[S2], [S3], [S7]
B4	Responsiveness to change	[S4], [S7], [S20], [S23], [S25]
B5	Rapid delivery and validation	[S4], [S20], [S25]
B6	Improved customer relationships	[S4], [S7]

2) Improved requirements understanding (B2): Requirements and their priorities are understood better and requirements are rapidly validated due to the immediate access to customers [S2], [S3], [S4], [S7], [S23] and due to the direct communication between sites in distributed or global software development contexts [S18].

3) Reduced overallocation (B3): Agile RE approaches reduce the tendency to over allocate (overburden) development resources [S2], [S7] and enable better management of the scope of the development [S3].

4) Responsiveness to change (B4): Agile RE approaches are more responsive to change in both the content and priorities of the requirements, especially when the environment or the customers' understanding about the software solution changes [S4], [S7], [S20], [S23], [S25].

5) Rapid delivery and validation (B5): Agile RE approaches allow more rapid delivery of value and validation of the solution system [S4], [S20], [S25].

6) Improved customer relationships (B6): Agile RE approaches improve relationships with customers and users [S4], [S7].

Benefits from using specific agile RE practices outside their original intent have also been reported. Ludlow [S14] describes how user stories were successfully used for strategic planning, the identification of gaps and extending existing systems in addition to using them for defining new functionality. Waldmann [S28] described how the principles of agile development methods were applied to a requirements engineering process that fed requirements to a development organization that was using the V-model for development.

D. Problems and solutions related to the Agile RE approach

Several researchers have studied problems caused by the Agile RE approach and proposed solutions to the problems. These problems and solutions are grouped and summarized under thematic areas in the following. The thematic areas and the related articles are summarized in Table V.

1) Problems with client or customer representatives (P1): Agile RE relies on the intensive interaction and trust between the developers and the customers, which may be hard to achieve [S4], [S22], [S23]. Bespoke agile development relies on the availability and competency of the client representative(s) [S7], [S22] and on the client's willingness to spend resources on the constant validation of the requirements and the solution system [S18]. Large projects with many customers rely on a customer representatives (such as product owners in Scrum or product managers) who must divide their attention between the customers and the developers [S12]. If such customer representatives do not exist, the valuation and prioritization of requirements is performed by the developers who may lack understanding about the market [S7]. The following solutions were proposed to these problems:

A requirements engineer who accompanies the on-site customer can perform RE related tasks for the on-site customer to reduce the on-site customer's work burden and skill requirements [S11]. Roles such as the domain owner and business analyst may be beneficial when agile RE is performed in large organizations to improve the understanding of the requirements [S5]. Ethnography can be used for requirements elicitation and analysis in an agile software development project in order to better understand customer needs [S26]. Enhancing Scrum with goal-oriented requirements engineering and IT governance processes may mitigate issues with the single product owner model in large-scale Scrum development contexts [S12].

Additional requirements documentation may be beneficial in large agile development projects due to the overhead created by face-to-face communication between the client or customer representatives and the development teams [S20]. Mind-mapping is one solution for requirements elicitation in Scrum when the team has multiple customers [S15]. Initial results suggest that customers who receive training in requirements elicitation using a mind-map produce more requirements, measured in function points, than the customers who receive no training in requirements elicitation [S15].

Test-driven development ideas can be expanded to requirements elicitation and analysis and to system validation by having the customers write requirements as storytests [S17]. The solution system can then be automatically validated against the storytests, which reduces the customers' work burden [S17]. Acceptance test-driven development (ATDD) covers the middle ground between traditional RE and agile RE [S10]. However, ATDD requires more up-front work and discipline from the developers and customers than pure agile RE [S10].

2) Insufficiency of the user story format (P2): Only simple, customer visible functional requirements can be described as basic user stories [S4], [S24]. Properties such as consistency and veriafibility of user stories are difficult to validate [S4], [S23]. When the software system is large and complex or hardware-dependent, user stories do not convey enough information for software design, and separate system and subsystem requirements are required [S5], [S24]. User stories lack the power of expression to describe differences between product families for roadmapping purposes [S24]. (Basic) user stories and agile RE practices do not explicitly support

Table V
PROBLEM THEMES AND RELATED ARTICLES

Code	Theme	Articles
P1	Problems with client or customer representatives	[S4], [S7], [S12], [S18], [S22], [S23]
P2	Insufficiency of the user story format	[S4], [S5], [S7], [S13], [S23], [S24]
P3	Difficulties in the prioritization of requirements	[S2], [S4], [S12], [S22], [S23]
P4	Growing technical debt	[S7], [S23]
P5	Reliance on tacit requirements knowledge	[S4], [S12], [S20], [S22], [S24]
P6	Imprecise effort estimates	[S4], [S23]

requirements traceability which creates traceability problems when requirements, code and tests change over time [S7], [S13], [S24]. Non-functional requirements are often ignored or their implementation may rely on tacit knowledge [S4], [S23]. Requirements decisions are not documented which makes backtracking difficult [S18]. The following solutions were proposed to these problems:

Delivery stories, that extended user stories with functional specifications, high level design and test scenarios, may mitigate the insufficiency of the user story format in large client-provider networks [S5]. Delivery stories improve the understanding of the requirements and help in the design of the system [S5]. A hierarchical requirements model and explicit product and portfolio management processes may improve the clarity of the requirements and create requirements that are sufficiently detailed for developers [S27]. An aspect-oriented requirements approach can be used to identify and manage crosscutting requirements [S1]. A feature-driven development based requirements analysis process may be used to explicitly addressed web application security requirement concerns [S9].

After initial requirement elicitation from the users, user stories should be transformed into complete, unambiguous and verifiable requirements using the traditional requirements analysis and specification process [S8]. When large and complex or hardware-dependent systems are developed, the most important (traditional) RE practices, such as requirements traceability management, should be preserved in agile transformation instead of relying solely on user stories [S24].

3) Difficulties in the prioritization of requirements (P3): Requirements prioritization is based on the immediate business value, which might cause system architecture or system improvement related requirements to be initially ignored [S2], [S4], [S7]. The rapid delivery of simple, customer visible requirements may create unrealistic expectations in the customers [S4]. Different customers may have conflicting needs and reaching consensus on the priorities of the requirements may be difficult [S4], [S7], [S12], [S23]. In bespoke agile development, the clients may be unwilling or incapable of directly prioritizing requirements [S22]. The quantitative Real Options approach to requirements prioritization might be infeasible due to the lack of detailed requirements analysis in agile methods [S21]. No solutions to P3 were proposed in the articles.

4) Growing technical debt (P4): Agile RE practices may lead to inadequate or inappropriate architecture due to the short planning time horizon [S7], [S23]. Refactoring the system architecture may require a complete rewrite of the system [S23]. No solutions to P4 were proposed in the articles.

5) Reliance on tacit requirements knowledge (P5): Agile RE requires highly skilled people and personnel turnover is troublesome since most requirements knowledge is tacit [S4], [S12], [S20], [S22], [S24]. One solution was proposed to this problem: Additional requirements documentation may mitigate the knowledge loss created by personnel turnover in large agile development projects [S20].

6) Imprecise effort estimates (P6): Precise cost and schedule estimation, often required by management, is difficult to do without extensive requirements analysis and specification [S4], [S23]. No solutions to P6 were proposed in the articles.

V. DISCUSSION

This section presents the discussion of the results. First the article metadata, content and type is discussed. Then a definition of agile RE is proposed. Finally, the benefits, problems and solutions from agile RE literature are discussed.

A. Article metadata, context and type

The distribution of articles in different publication venues suggests that there is no primary venue for articles on RE in ASD. Although the RE conference series was the most prominent conference venue, it still contained only three of the fifteen (20%) conference articles. Most of the journal articles were published in general software engineering journals and no articles were published in the Requirements Engineering journal. The list of publication venues suggests that factors other than an RE focus have been more important in the selection of a publication venue, and that RE in ASD as a subject has not yet found a comfortable home in scientific publication venues. However, the five articles published in IEEE Software suggest that the topic has been considered interesting and important for software professionals [13].

Most articles have unspecified ASD as the context ($N = 20, \approx 71\%$). This is somewhat problematic, since there are many agile methods and their practices, purpose and philosophy vary greatly [11]. Many articles, that do not explicitly give the ASD method context, discuss user stories or the role of an on-site customer or a product owner, which suggest an implicit Scrum or Extreme Programming context. An explicit statement of the particular ASD method context would improve the generalizability of the results. Overall, Scrum is the most common explicitly stated context ($N = 7, 25\%$) and Feature Driven Developments gets one mention ($\approx 4\%$).

The majority of the included articles had an empirical part. Case studies, experience reports, and method and tool evaluations total approximately 60% ($N = 17$) of the papers. Descriptive, empirical knowledge is especially valuable when a new and interesting phenomenon, such as RE in ASD, is studied in order to create groundwork for later, more theoretical and explanatory research. Eight papers ($\approx 29\%$) were method proposals without empirical evaluation. Such papers have limited relevance on their own. Further empirical evaluation research is required to study the applicability, effectiveness and efficiency of the proposed methods. Finally, three of the articles ($\approx 11\%$) were position papers. Such articles have value in bringing forward interesting phenomena and gaps in the scientific knowledge.

B. Towards a definition of agile RE

The definition of agile RE has been difficult since the most studied agile methods, Extreme Programming and Scrum, do not explicitly discuss RE but include RE practices in their overall method description. The approach taken by researchers has been to identify requirements related activities and name them agile RE practices. Based on the results, it is unclear how much agile RE actually differs from traditional RE.

There is no existing universal definition of agile RE. Based on the synthesis of the included articles, we propose the following definition:

> In agile RE, the requirements are elicited, analysed and specified in an ongoing and close collaboration with a customer or customer representative in order to achieve high reactivity to changes in the requirements and in the environment. Continuous requirements re-evaluation is vital for the success of the solution system, and the close collaboration with the customer or customer representative is the essential method of requirements and system validation.

C. Benefits, problems and solutions

The identified benefits from the agile RE approach are similar to the benefits agile methods have been purported to create in general [9], [12], [20], [21]: The lack of documentation improves development efficiency. Close collaboration with customers or customer representatives produces software that fits the customers' needs and improves customer relationships. Iterative development and strict scope management create sustainable work practices. Development is highly reactive to change and the produced software is rapidly validated.

Many of the identified problems mirror the proposed benefits. The ideal collaboration between the development and the customers or customer representatives may not be feasible in complex, large environments. Similarly, when complex and large software is developed, user stories may not suffice as the only source of requirements documentation and prioritization of requirements may be very difficult. The rapid implementation and release of new functionality may create unsustainable customer expectations and increase technical debt.

The solutions proposed to the aforementioned problems have concentrated on the problems with client or customer representatives (P1) and on the insufficiency of the user story format (P2). Most solutions proposed to P1 are related to the introduction of traditional RE inspired practices. These included a traditional requirements engineering role, explicit requirements elicitation and additional requirements documentation. A few articles suggested extending automated testing ideas to requirements validation as a solution to P1. Many solutions to P2 also originated from traditional RE research. These included extending the user story format with more details, hierarchy or traceability and limiting the use of user story format to requirements elicitation. Additional requirements documentation was suggested as a solution to the reliance on tacit requirements knowledge (P5). No solutions to the difficulties in prioritization of requirements (P3), the growing technical debt (P4) and the imprecise effort estimation (P6) were proposed. These results suggest that additional research on solving problems related to the prioritization of requirements, managing technical debt, tacit requirements knowledge and effort estimation in ASD is required. However, many of the proposed solutions to problems related to client or customer representatives (P1) and to insufficiency of the user story format (P2) have not been empirically evaluated, and additional research on the topics is also warranted.

D. Limitations

The literature search was constrained to the Elsevier Scopus abstracts database. Although Elsevier claims that Scopus is the most comprehensive database of peer-reviewed research abstracts[2], additional databases could have produced additional included articles. However, all high-quality SE journals and conference proceedings are included in Scopus, and it is unlikely that additional articles would have change the overall results of this study. The search string was confined to a small set of keywords. These keywords were considered the most relevant to the research question of

[2]http://www.elsevier.com/elsevier-products/scopus

this study. Additional keywords might have produced more articles, but the information in those articles would have most likely been peripheral to the research question and provide very little additional insight.

VI. Conclusion

This paper presents the results of a mapping study on requirements engineering in agile software development. In total, 28 articles were included in the mapping study. Although the publication years of the articles span from 2004 to 2014, there is still considerable uncertainly of what requirements engineering in agile software development is. The case study method has the power to create understanding about a new phenomenon, but many of the included case study articles focused on method proposal and evaluation instead of creating understanding about the phenomenon. The proposed benefits from agile requirements engineering follow the overall benefits agile methods are purported to have. Many of the identified problems were related to the agile development of complex, large software systems and to ASD in large organizations, and this topic clearly needs more research. The evaluation of the methods that were proposed to solve the problems was weak in general, and more effort needs to be put into empirically evaluating the solutions.

Acknowledgements

This research was partially funded by TEKES as a part of the Need for Speed research program of DIGILE.

References

[1] J. Beall. Medical publishing triage – chronicling predatory open access publishers. *Annals of Medicine and Surgery*, 2(2):47–49, 2013.

[2] K. Beck and C. Andres. *Extreme programming explained : embrace change*. Addison-Wesley, Boston, MA, USA, 2nd edition, 2004.

[3] K. Beck, M. Beedle, A. van Bennekum, A. Cocburn, W. Cunningham, M. Fowler, J. Grenning, A. Hunt, R. Jeffries, J. Kern, B. Maric, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas. Manifesto for agile software development. http://agilemanifesto.org/, 2001. Retrieved Sep 17th, 2014.

[4] P. Berander and A. Andrews. Requirements prioritization. In A. Aurum and C. Wohlin, editors, *Engineering and Managing Software Requirements*, pages 69–94. Springer Berlin Heidelberg, 2005.

[5] B. W. Boehm. Requirements that handle ikiwisi, cots, and rapid change. *Computer*, 33(7):99–102, 2000.

[6] I. K. Bray. *An introduction to requirements engineering*. Addison-Wesley, Harlow, UK, 2002.

[7] R. Britto, M. Usman, and E. Mendes. Effort estimation in agile global software development context. In T. Dingsøyr, N. Moe, R. Tonelli, S. Counsell, C. Gencel, and K. Petersen, editors, *Agile Methods - Large-Scale Development, Refactoring, Testing, and Estimation*, volume 199 of *Lecture Notes in Business Information Processing*, pages 182–192. Springer, 2014.

[8] I. F. da Silva, P. A. da Mota Silveira Neto, P. O'Leary, E. S. de Almeida, and S. R. de Lemos Meira. Agile software product lines: a systematic mapping study. *Software: Practice and Experience*, 41(8):899–920, 2011.

[9] P. Deemer, G. Benefield, C. Larman, and B. Vodde. The Scrum Primer — a lightweight guide to the theory and practice of Scrum, version 2. Technical report, 2012.

[10] P. Diebold and M. Dahlem. Agile practices in practice: A mapping study. In *Proc. 18th International Conference on Evaluation and Assessment in Software Engineering (EASE Í4)*, pages 30:1–30:10, New York, NY, USA, 2014. ACM.

[11] T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe. A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, 85(6):1213–1221, 2012.

[12] T. Dybå and T. Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9-10):833–859, 8 2008.

[13] H. Erdogmus. Tips for software authors. *IEEE Software*, 24(5):5–7, Sept 2007.

[14] E. Hossain, M. A. Babar, and P. Hye-young. Using Scrum in global software development: A systematic literature review. In *Proc. Fourth IEEE International Conference on Global Software Engineering (ICGSE 2009)*, pages 175–184, July 2009.

[15] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband. A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior*, in press, 2014.

[16] ISO/IEC/IEEE. Systems and software engineering – life cycle processes – requirements engineering. *International Standard 29148:2011(E)*, pages 1–94, Dec 2011.

[17] G. Jurca, T. D. Hellmann, and F. Maurer. Integrating agile and user-centered design: A systematic mapping and review of evaluation and validation studies of agile-UX. In *Proc. Agile Conference (AGILE)*, pages 24–32, July 2014.

[18] B. A. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007-001, Keele University and Durham University, 2007. Version 2.3.

[19] D. Salah, R. F. Paige, and P. Cairns. A systematic literature review for agile development processes and user centred design integration. In *Proc. 18th International Conference on Evaluation and Assessment in Software Engineering (EASE '14)*, pages 5:1–5:10, New York, NY, USA, 2014. ACM.

[20] K. Schwaber and M. Beedle. *Agile software development with Scrum*. Prentice-Hall, Upper Saddle River, NJ, USA, 2002.

[21] K. Schwaber and J. Sutherland. The Scrum Guide — the definitive guide to Scrum: The rules of the game. Technical report, Scrum.org, July 2013.

[22] P. Sfetsos and I. Stamelos. Empirical studies on quality in agile practices: A systematic literature review. In *Proc. Seventh International Conference on the Quality of Information and Communications Technology (QUATIC)*, pages 44–53, Sept 2010.

[23] O. Sohaib and K. Khan. Integrating usability engineering and agile software development: A literature review. In *Proc. 2010 International Conference on Computer Design and Applications (ICCDA)*, volume 2, pages 32–38, June 2010.

[24] S. Stavru. A critical examination of recent industrial surveys on agile method usage. *Journal of Systems and Software*, 94(0):87, 2014.

[25] M. Usman, E. Mendes, F. Weidt, and R. Britto. Effort estimation in agile software development: A systematic literature review. In *Proc. 10th International Conference on Predictive Models in Software Engineering (PROMISE '14)*, pages 82–91, New York, NY, USA, 2014. ACM.

[26] VersionOne, Inc. 8th annual "State of Agile Development" survey. Technical report, 2014.

Primary Sources

[S1] J. Araújo and J. C. Ribeiro. Towards an aspect-oriented agile requirements approach. In *Proc. Eighth International Workshop on Principles of Software Evolution*, pages 140–143, Sept 2005.

[S2] E. Bjarnason, K. Wnuk, and B. Regnell. A case study on benefits and side-effects of agile practices in large-scale requirements engineering. In *Proc. 1st Workshop on Agile Requirements Engineering (AREW '11)*, pages 3:1–3:5, New York, NY, USA, 2011. ACM.

[S3] E. Bjarnason, K. Wnuk, and B. Regnell. Are you biting off more than you can chew? a case study on causes and effects of overscoping in large-scale software engineering. *Information and Software Technology*, 54(10):1107–1124, 10 2012.

[S4] L. Cao and B. Ramesh. Agile requirements engineering practices: An empirical study. *Software*, 25(1):60–67, 2008.

[S5] M. Daneva, E. van der Veen, C. Amrit, S. Ghaisas, K. Sikkel, R. Kumar, N. Ajmeri, U. Ramteerthkar, and R. Wieringa. Agile requirements prioritization in large-scale outsourced system projects: An empirical study. *Journal of Systems and Software*, 86(5):1333–1353, 5 2013.

[S6] S. Dimitrijević, J. Jovanović, and V. Devedžić. A comparative study of software tools for user story management. *Information and Software Technology*, (in press).

[S7] N. A. Ernst and G. C. Murphy. Case studies in just-in-time requirements analysis. In *Proc. Second International Workshop on Empirical Requirements Engineering (EmpiRE)*, pages 25–32. IEEE, 2012.

[S8] D. Firesmith. Generating complete, unambiguous, and verifiable requirements from stories, scenarios, and use cases. *Journal of Object Technology*, 3(10):27–39, 2004.

[S9] X. Ge, R. F. Paige, F. A. C. Polack, H. Chivers, and P. J. Brooke. Agile development of secure web applications. In *Proc. 6th International Conference on Web Engineering (ICWE '06)*, pages 305–312, New York, NY, USA, 2006. ACM.

[S10] B. Haugset and T. Stålhane. Automated acceptance testing as an agile requirements engineering practice. In *Proc. 45th Hawaii International Conference on System Science (HICSS-45)*, pages 5289–5298, 2012.

[S11] E. Hochmüller. The requirements engineer as a liaison officer in agile software development. In *Proc. 1st Workshop on Agile Requirements Engineering (AREW '11)*, pages 2:1–2:4, New York, NY, USA, 2011. ACM.

[S12] O. Ktata and G. Lévesque. Agile development: Issues and avenues requiring a substantial enhancement of the business perspective in large projects. In *Proc. 2nd Canadian Conference on Computer Science and Software Engineering (C3S2E '09)*, pages 59–66, New York, NY, USA, 2009. ACM.

[S13] A. Lucia and A. Qusef. Requirements engineering in agile software development. *Journal of Emerging Technologies in Web Intelligence*, 2(3), 2010.

[S14] L. Ludlow. The application of user stories for strategic planning. In G. Concas, E. Damiani, M. Scotto, and G. Succi, editors, *Agile Processes in Software Engineering and Extreme Programming*, volume 4536 of *Lecture Notes in Computer Science*, pages 198–202, 2007.

[S15] I. Mahmud and V. Veneziano. Mind-mapping: An effective technique to facilitate requirements engineering in agile software development. In *Proc. 14th International Conference on Computer and Information Technology (CIT-2014)*, pages 157–162, 2011.

[S16] N. Maiden. Cherishing ambiguity. *Software, IEEE*, 29(6):16–17, 2012.

[S17] R. Mugridge. Managing agile project requirements with storytest-driven development. *Software, IEEE*, 25(1):68–75, 2008.

[S18] R. Noordeloos, C. Manteli, and H. van Vliet. From RUP to Scrum in global software development: A case study. In *Proc. Seventh International Conference on Global Software Engineering (ICGSE)*, pages 31–40. IEEE, 2012.

[S19] K. Orr. Agile requirements: opportunity or oxymoron? *Software, IEEE*, 21(3):71–73, May 2004.

[S20] F. Paetsch, A. Eberlein, and F. Maurer. Requirements engineering and agile software development. In *Proc. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE)*, pages 308–313, 2003.

[S21] Z. Racheva and M. Daneva. How do real options concepts fit in agile requirements engineering? In *Proc. Eighth ACIS International Conference on Software Engineering Research, Management and Applications (SERA)*, pages 231–238, 2010.

[S22] Z. Racheva, M. Daneva, K. Sikkel, A. Herrmann, and R. Wieringa. Do we know enough about requirements prioritization in agile projects: Insights from a case study. In *Proc. 18th IEEE International Requirements Engineering Conference (RE 2010)*, pages 147–156, 2010.

[S23] B. Ramesh, L. Cao, and R. Baskerville. Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal*, 20(5):449–480, 2010.

[S24] J. Savolainen, J. Kuusela, and A. Vilavaara. Transition to agile development — rediscovery of important requirements engineering practices. In *Proc. 18th IEEE International Requirements Engineering Conference (RE 2010)*, pages 289–294, Sept 2010.

[S25] I. Sommerville. Integrated requirements engineering: a tutorial. *Software, IEEE*, 22(1):16–23, 2005.

[S26] N. C. Surendra. Using an ethnographic process to conduct requirements analysis for agile systems development. *Information Technology and Management*, 9(1):55–69, 2008.

[S27] K. Vlaanderen, S. Jansen, S. Brinkkemper, and E. Jaspers. The agile requirements refinery: Applying SCRUM principles to software product management. *Information and Software Technology*, 53(1):58–70, 1 2011.

[S28] B. Waldmann. There's never enough time: Doing requirements under resource constraints, and what requirements engineering can learn from agile development. In *Proc. 19th IEEE International Requirements Engineering Conference (RE '11)*, pages 301–305, 2011.