Using Conceptual Models in Agile Software Development: A Possible Solution to Requirements Engineering Challenges in Agile Projects

ABSTRACT Studies on requirements engineering with Agile methods for software development have shown difficulties in managing the quality of the requirements and communicating with users. Some of these studies have proposed conceptual modeling as a solution to these problems. However, the effort that is required to create conceptual models conflicts with Agile values. In this paper, we propose an approach for using conceptual models in projects while adhering to Agile principles. This approach focuses on projects in which requirements are expressed as user stories that are the main artifacts of the requirements used for software development with Agile methods. First, the paper presents a literature review in which we have systematically searched for the challenges to requirements engineering with Agile methods. Next, we report on a survey study in which we interviewed 16 experts in the Agile methodology to confirm the identified challenges and find new ones that are not covered in the literature. Based on a thematic analysis of the challenges, we argue that most of them map to the two main purposes of using conceptual models in software development: improving communication and understanding requirements. To effectively use conceptual models in projects that use the Agile methodology, several conditions must be met, which we make explicit in the paper. The paper ends by illustrating how these conditions can be met demonstrating the models that can be automatically generated from a given set of user stories. This demonstration was subsequently used to obtain feedback from the experts on the perceived benefits of conceptual models in addressing the challenges of requirements engineering.

INDEX TERMS Agile Software Development, User Stories, Conceptual Models

I. INTRODUCTION

The purpose of the manifesto for Agile Software Development [1] was to uncover better ways of developing software. The manifesto proposed the following values: individuals and interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation; and responding to change over following a plan.

Based on these values, Agile methods like Extreme Programming, Scrum, Kanban, and SAFe, prescribe shorter and more incremental development iterations than other methodologies for software development do. After each iteration the requirements can be modified [2]. Therefore, developers have adopted Agile methods in practice for their ability to embrace change rather than to avoid it and to respond efficiently and effectively to changing requirements [3].

The main artifact of requirements in the Agile methodology is the user story [4]. A user story is a simple textual description of a desired feature of the working software that is written from the user's perspective and is typically formulated using a standardized text template [5, 6] (see Table 6 in Section 7 for examples). In the Agile methodology, developers most often use user stories to describe functional requirements [7] (as in Table 6) but they can also use them to describe nonfunctional requirements. Although user stories provide an easy-to-use mechanism for documenting and communicating the desired system features, user stories as used in practice are prone to ambiguity that leads to risks of the imprecision, inconsistency, incompleteness, and redundancy of the requirements [8].

Other software development methodologies, for instance the rational unified process (RUP) or that follow an object-oriented methodology to software development, use conceptual models such as use case diagrams in the unified modeling language (UML), activity diagrams, state machine diagrams, and class diagrams (see Figures 1 and 2 in Section 7 for examples) to conceptualize the domain that the system supports and to visually represent the functionality expected from the system. Conceptual models support requirements analysis and system design [9]. As visual representations, they facilitate communication among the project team members and help create a shared understanding of how a system should support a domain [10].

In the Agile methodology, the use of conceptual models is not common as the effort required to create the models conflicts with Agile values. Instead, the focus of requirements engineering in the Agile methodology is to efficiently and flexibly transfer ideas from the customer to the development team but without creating excessive documentation of the requirements [11].

In practice, however, requirements engineering in the Agile methodology is challenging, as many surveys and literature reviews have reported [11-14]. So, while visual models have proven their effectiveness in supporting requirements engineering, their use is not advocated by Agile proponents. Therefore, we explore in the following question: *how can conceptual models address the challenges of requirements engineering in the Agile methodology for software development without conflicting with its values?* Recently, studies have suggested that the use of conceptual models to understand and analyze requirements that developers have formulated as a set of related user stories (e.g., a theme or epic in Scrum) is a research opportunity [8]. We not only investigate this broad research opportunity but also explore a more focused one: *how can conceptual models address the challenges of requirements engineering in the Agile methodology for software development that are related to user stories?*

To investigate these research questions, we first reviewed the recent literature and conducted expert interviews to identify these challenges. In doing so, we have updated existing surveys and review studies (e.g., [11-14]) as the Agile methodology for software development is a rapidly changing area [15] that has become more complex with projects involving multiple team members in multiple locations (e.g., offshore premises) [16]. In our investigation, we focus specifically on the challenges related to the documentation of user stories as an artifact for requirements.

Next, through a thematic analysis of the identified challenges, we show that most challenges can be mapped to the main purposes for using conceptual models: improving communication among project stakeholders and improving the understanding of the requirements.

Subsequently, to further explore how conceptual models can be used without conflicting with Agile values, we identify the conditions that need to be met for the use of conceptual models in projects that document user stories for requirements. We also demonstrate an example in which we are able to meet these conditions when models can be automatically generated from the information captured by the user stories.

Finally, to evaluate the usefulness of conceptual models in addressing the challenges to requirements engineering in the Agile methodology for software development, we demonstrate a set of models to experts that we interviewed to seek their opinion on how these models could benefit requirements analysis in the Agile methodology. We also map these benefits to the challenges of requirements engineering that we identified.

To summarize, this paper contributes to the state-of-the-art by providing an up-to-date overview of the challenges to requirements engineering that are observed in the current practice of the Agile methodology for software development: we identify those challenges that can potentially be addressed by using conceptual models; identify the conditions that need to be met for adopting conceptual models in the practices of the Agile methodology; and we demonstrate that it is possible to meet these conditions in projects that use user stories to document requirements. With these novel contributions, our explorative research provides a basis from which researchers can further investigate the use of conceptual models in the Agile methodology to support requirements engineering.

In Section 2, we provide a background on requirements engineering in the Agile methodology for software development. In Section 3, we describe the methodology of our explorative study. Section 4 presents the literature review of the challenges to requirements engineering for the Agile methodology. Section 5 presents the results of interviews with 16 experts in the Agile methodology for software development. In Section 6, we synthesize the results of the literature review and the interviews. In this section, we also map the identified challenges to the purposes of using conceptual models to explore how the use of conceptual models can address them. Following this mapping, in Section 7 we identify conditions for using conceptual models without compromising adherence to Agile values. We also demonstrate in this section how these conditions can be met for projects using user stories, and we report on the perceived benefits of models generated from user stories after being shown to our experts. Section 8 presents our conclusion and suggestions for further research.

II. REQUIREMENTS ENGINEERING IN THE AGILE METHODOLOGY

One of the most difficult tasks in developing software is requirements engineering [16]. As a result, software development has used the Agile methodology as a management tool. The Agile methodology comprises a set of principles that values individuals and interactions over processes and similarly, prefers working software

over documentation [1]. The Agile methodology prescribes an iterative and incremental development process in which the Agile team works closely with the customer [16]. The Agile methodology focuses on continuous and iterative improvement of the software development as driven by the actual experience of using the software [17]. Each iteration has phases of design, implementation, and testing as well as an requirements analysis [17].

To organize the Agile process, the team uses several methods such as Extreme Programming, Kanban, SAFe, and Scrum; Scrum is generally recognized as the most adopted method [16]. Scrum defines three roles for project teams: product owner, developer, and scrum master [16] . The scrum master guides and coaches the team to the proper understanding and use of Scrum. The product owner is responsible for managing the requirements and identifying the features to be implemented in the iterative development phases, also known as sprints [16, 18]. The implementation of the features is the responsibility of the developer.

In combination with methods such as Scrum, teams use several techniques such as behavior-driven development (BDD) and test-driven development (TDD) in the Agile methodology to organize the activities in requirements engineering. BDD comprises the use of user stories and acceptance criteria to specify a system's requirements [19]. In the TDD, the team writes tests before the developers produce the actual code, and this test writing is considered as part of the requirements engineering [4].

Using conceptual models in the Agile methodology is by itself not a novel idea. Recker and Green [20] have pointed out that although the Agile methodology has gained popularity recently, the use of conceptual models has not disappeared from software development. Some researchers suggest using even more documentation (including visual models) and tools in the Agile methodology. For instance, contrary to Agile practices, researchers have found that tacit knowledge is not sufficient and formal documentation is therefore necessary [21]. Vithana [16] argues that since formal documentation is missing, verification of requirements might not be adequately addressed. To alleviate the communication problems in Agile projects, Sundararajan et al. [21] suggest the creation of design documents for the overall architecture of the systems, the design of the database, and the interfacing needs of the systems. Daneva et al. [22] conduct a survey and find that the sharing of domain knowledge is an important characteristic for prioritizing requirements in Agile projects. Consequently, they suggest the introduction of the role of domain owner to the Agile team that is responsible for acquiring knowledge of the business processes. This knowledge is usually reflected in business process models.

Related to user stories, Kannan et al. [23] have proposed the use of use case diagrams in the UML that are obtained from user stories where each user story is represented as a use case. Based on a survey of the use of models, Schön et al. [15] finds that some organizations that use the Agile methodology are using use case diagrams, story cards (i.e. details of user stories), and mind maps to create shared understanding and getting potential users involved in the development process. Along a similar vein, Helmy et al. [24] argue that developing high-level domain models as part of the initial efforts at architectural modeling can guide both the design of the physical data model and class design. Trkman et al. [25] suggest that using business process models leads to a better understanding of the dependencies among user stories. In their systematic literature review, Amna and Poels [8] find 13 studies that propose the use of models for tackling problems related to the ambiguity in user stories. They also find that the literature has only validated a few of these proposed solutions. Further, the question of how to develop and use the models within an Agile project is still an open one.

III. METHODOLOGY

We conducted an exploratory study to investigate our research questions. Our study consisted of three steps. First, we conducted a literature review to identify current challenges to requirements engineering in Agile projects. The literature reviews and surveys [11-14] indicated that in spite of the popularity of applying the Agile methodology to software development, there were key challenges in eliciting, documenting, analyzing, verifying, and validating requirements in these projects. Some of these studies reviewed research papers published in the early development of Agile when its use was not yet common. As of today, the Agile methodology is used extensively in software development which is an area that has increased in complexity (e.g., global software development) [16]. Therefore, the verification of whether the challenges to requirements engineering that these past reviews and surveys have identified are still relevant and whether studies or practitioners have found new challenges in the development of Agile.

As a second step we conducted semi-structured interviews with experienced practitioners of the Agile methodology to validate the challenges identified by the literature and to identify new ones it had not uncovered. In these interviews we also focused specifically on the practice of describing requirements with user stories.

The synthesized list of literature-based and interview-based challenges was then mapped for the purpose of using conceptual models to provide some initial insights into the answers to the research questions.

As a third and last step, we reflected on the conditions that needed to be met for using conceptual models in the Agile methodology. To further explore the answers to our research questions, we then demonstrated how the information captured in user stories could be used to generate conceptual models, and how these models could then be useful for addressing the identified challenges. To evaluate whether expert practitioners of the Agile methodology would also perceive this process as useful, we continued some of the interviews of the second step by showing the example set of related user stories and the models generated from them and asking the participants how they thought these models could benefit the analysis of requirements.

A. LITERATURE REVIEW

To update the current knowledge of the challenges that Agile faces in requirements engineering, we performed a systematic search of papers published in the academic journals in the Elsevier and Science Direct libraries. These libraries include numerous field journals that frequently publish peer-reviewed papers related to software development (e.g., Journal of Systems and Software, Information and Software Technology, IEEE Software, and Software Quality Journal). They also contain the flagship journals of the broader field of software engineering and information systems (e.g., MIS Quarterly and Information Systems Research).

To manage the scope of the literature review and effort required to analyze papers, we searched for those that explicitly discussed the challenges to requirements engineering for the Agile methodology. In other words, we did not intend to review all published work on the Agile but to look ourselves for elements that could be interpreted as challenges to engineering and managing requirements that could also introduce a certain degree of subjectivity. To search for papers, we built search strings that contained the words “agile”, “requirement(s)”, and “challenge(s)”. The “challenges” searches used three synonyms “difficulties”, “obstacles”, and “hindrances” (and the singular version of these plural words) as alternates. As agile is a term also used in other domains, we added “software” or “information systems” to these search strings to focus exclusively on papers related to the development of software or information systems.

Our searches were performed at the end of 2020 and were limited to the most recent 11 full years from 2009-2019. The automated search returned 865 papers. Our inclusion criteria for the review were "peer-reviewed papers" (e.g., excluding book reviews and editorials) and "relevancy". The latter criterion was evaluated by reading the papers' abstracts and verifying whether the term "agile" referred to software development and its challenges (or difficulties, obstacles, etc.) while also referring to requirements engineering related activities or concerns. The exclusion criteria used language (only papers in English) and when the full text of the paper was not available. After verifying these criteria, 57 papers were selected for further analysis (see Table A2 in the Appendix for the full references of these papers).

The analysis itself involved extracting the challenges to requirements engineering discussed in the papers. Next, a unified list of these challenges was compiled. As we ended up with a list of 22 challenges, much longer than the 8 challenges in each of the previously mentioned reviews [11-14], we decided to group challenges into several themes in our analysis. The thematic analysis [26] is a qualitative data method that relies on coding techniques to make sense of the data and discover underlying themes. Although these coding techniques are like those used in the grounded theory method [27], the thematic analysis does not aim at constructing hypotheses about phenomena observed in the data. We took a reflexive and inductive approach to the thematic analysis by using the open coding technique that has no a priori theoretical definition of the themes to guide the coding and summarization process.

B. INTERVIEWS

Practitioners of the Agile methodology were interviewed to validate the identified challenges to requirements engineering found in the literature and to identify any new ones not mentioned in the papers that were reviewed. We looked for practitioners that should be knowledgeable about these challenges due to their expertise and experience. To get access to these practitioners, we used a convenience sampling method. We searched for members of the Agile community on LinkedIn and based on their profiles, selected members that had at least five years of experience working with Agile, described themselves as a senior business analyst or a technologist, and that had served on Agile teams in different roles.

TABLE 1. Participant Profiles.

Interviews were conducted using a semi-structured interview protocol. The first guiding question was: "What challenges do you face with requirements in terms of using Agile for software development?" Based on our second research question, as the use of user stories is the most popular requirements artifact in the Agile methodology, we introduced a second question on user stories: "What challenges do you face with the development and management of user stories in the Agile methodology?" Finally, we demonstrated a set of models generated from an example set of user stories to 11 of the interview participants (F1 to F11) – we did not consider doing this in the five first interviews which was the reason why we only got 11 expert opinions. We then asked them to comment on the benefits such models would add to the requirements analysis in Agile if they were available. In total we recruited 16 participants to interview (Table 1). The first five interviewees (who were not shown the models generated for the demonstration) were coded with IDs E1 to E5, and the later 11 interviewees were coded with IDs F1 to F11.

Participant ID	Years in IT	Years in Agile Experience	Current Job Title
E1	19.5	7	Product Manager, Business Technologist
E2	10	6	Lead Business Systems Consultant
E3	24	11	Senior Business Analyst
E4	17	9	Business Analyst
E5	17	8	Sr. Business Systems Analyst
F1	12	10	Senior Manager- Technology Transformation
F2	20	8	Application and Scrum Master
F3	21	11	Delivery Manager
F4	24	12	Delivery Lead
F5	24	12	Deputy Chief Microsoft Technology Associate
F6	20	18	Senior Manager- QA Delivery
F7	12	8	Delivery Lead
F8	16	10	Senior Project Manager
F9	30	14	Agile Transformation Coach
F10	15	11	Project Manager
F11	21	8	AVP Project Manager

The average length of for the first half of the interviews was 20 minutes (i.e., Agile challenges) and 22 minutes for the second half of the interviews (i.e., model benefits). All interviews were recorded, and transcripts were generated. For each statement, words or phrases related to what could be identified as challenges or benefits were identified.

IV. RESULTS OF THE LITERATURE REVIEW

Table A1 in the appendix contains the list of 22 challenges to requirements engineering and the papers that mention them. Some challenges were found in just one or a few papers but often there were several papers

mentioning the same challenge, up to a maximum of 11. The thematic analysis resulted in five overarching themes: project team, customer involvement, requirements quality, user stories, and testing.

If a challenge was associated with requirements engineering or management issues that related to project teams, their management, or any specific role within the team (e.g., Scrum master, product owner); then this challenge was categorized in the project team theme. There were five challenges identified in this theme (1-5).

If a challenge related to the involvement of customers in requirements processes, then it was categorized in the customer theme. We found two challenges under this theme (6-7).

Challenges related to analyzing or verifying requirements were categorized in the requirements quality theme. These challenges (8-15) referred to the specific problems of managing requirements in Agile projects or using them for purposes like estimation or prioritization.

The fourth theme was user stories and contained those challenges that referred specifically to this requirements artifact (16-18).

The final theme was about the role of requirements during testing in Agile projects (19-20).

Two challenges, (21) external visibility on project tasks, and (22) inadequate or inappropriate architecture and interfaces could not be categorized in these themes as they seemed to be stand-alone challenges that were not related to the other challenges. All the challenges are summarized in Table 2.

TABLE 2. Thematic analysis of challenges to requirements in the reviewed papers.

Challenge IDs	Theme	Challenges
1-5	Project Team	Team's lack of involvement and motivation, breakdown of team communication and coordination, difficulty in managing dispersed teams, sharing of knowledge, and lack of management involvement
6-7	Customer Involvement	Difficulty in customer interaction, customer inability and disagreement
8-15	Requirements Quality	Difficulty in estimating time and costs, minimal documentation, incomplete nonfunctional requirements, incomplete and missing requirements, ambiguous requirements, requirements volatility, prioritizing requirements, and inadequate requirements verification
16-18	User Stories	Detailed user stories not created, user stories are not integrated, difficulty in decomposing user stories
19-20	Testing	Availability of testing resources, reducing testing and test coverage
21		Inadequate or inappropriate architectures and interfaces

22		External visibility on project tasks

The 22 challenges presented here are neither mutually exclusive nor independent from each other. One challenge might be the effect or a cause of another challenge.

A. CHALLENGES RELATED TO THE PROJECT TEAM

1. Team's lack of involvement and motivation

Project management needs to motivate team members to respond reasonably quickly to changes in requirements and with enough detail and understanding of the situation [28]. This motivation is reflected by team members' willingness to commit to a decision [29, 30] and take risks [31]; therefore, the members rely on the Scrum master's decisions [32]. Developers are less likely to adopt the Agile methodology if it is not made mandatory [33]. Tessem [34] shows that empowering and rewarding developers in decision-making drives the success of projects. McHugh et al. [35] find that the trust between developers and product owners is important for project success. Schön et al. [15] find that involving end users is crucial to the success of Agile for software development.

2. Breakdown of team communication and coordination

The lack of open communication is the root cause for many project failures [36]. A lack of detailed documentation can also lead to communication issues. Ramesh et al. [11] point out that when there is a breakdown in communication (e.g., turnover of personnel) then lack of documentation leads to the inability to scale the software, evolve the application over time, and to add new members to the development team. As Agile projects do not scale well, it requires much more effort in team coordination [37, 38]. This coordination is particularly difficult when the developer uses the outsourcing model [21, 39] or disperses the project team [4, 40, 41]. Heck and Zaidman [42] argue that using collaborative tools facilitates team coordination.

3. Difficulty in managing dispersed teams

Due to the globalization of software-intensive, high-technology businesses, developers are building Agile projects that involve teams in different geographic locations and time zones [43]. In a survey, Misra et al. [44]

find that the use of dispersed teams creates problems for project success. Communication and team coordination among the members is particularly challenging in these teams [45, 46]. Iqbal et al. [41] use a Delphi study to understand the challenges to requirements engineering in outsourced software development projects and find that the most common challenge is related to the communication among project teams, stakeholders, and customers. An example of such challenges is when stakeholders are unable to express the requirements clearly among themselves and to the team. Llyod et al. [47] argue that communication between the onshore and offshore sites is a key challenge to Agile projects for software development.

4. Sharing of knowledge

Spreading the knowledge across the project team is a challenge [48]. Drury et al. [32] find that decisions are made on the incomplete understanding of functionality by the team. Minimal documentation often reduces effective knowledge transfer [3, 48], and the team members often lack motivation to share their knowledge [49]. The use of short iterations, daily stand-up meetings, and the presence of customers onsite reduces the amount of time for sharing ideas outside the team [48]. In this context, Serrador and Pinto [50] show that having a clear goal of the project helps in successful project implementations. Yang et al. [51] argue that project teams often depend on tacit architectural knowledge, which is a challenge.

5. Lack of management involvement

Identifying and engaging managers in projects is a challenge [48, 52]. Dikert et al. [53] argue that due to a lack of management involvement, high-level requirements are often missing. Gregory et al. [54] find that project teams struggle to communicate the progress of development to the management team. On the other hand, support from top management improves the team's acceptance of the Agile methodology [33, 55], particularly if the management team promotes the perceived benefits of using that methodology [56].

B. CHALLENGES RELATED TO CUSTOMER INVOLVEMENT

6. Difficulty in customer interaction

Interaction with the customer in each iteration of the Agile methodology for software development is difficult [13]. This is because customers are busy and are typically not available in each iteration [11, 57]. Moreover, it takes more effort to negotiate the requirements with multiple representatives of the customer as it is challenging to unify the perspectives of these representatives [58], and at times customers are unaware of their own requirements [43, 59]. Also a very high level of customer interaction can cause conflicts [60].

7. Customer inability and disagreement

Customer inability refers to the incompetence of customers in terms of decision-making and complete domain knowledge, and customer disagreement is about the lack of consensus among more than one customer group involved in a project [12, 13, 16]. In this respect, Drury et al. [32] find that customers cannot always communicate accurately what they want to the project team. As Hess et al. [57] point out, the communication gap happens due to a lack of documentation. The disagreement between customer groups affects the team performance [11].

C. CHALLENGES RELATED TO REQUIREMENTS QUALITY

8. Difficulty in estimating time and costs

It is often difficult to estimate an accurate cost at the beginning of the project [11, 13, 59]. Several researchers [13, 61, 62] have shown that costs, resources, and time estimations are key challenges to Agile projects. A primary reason for these challenges is that the initial estimation is based on the set of user stories known at that time [13]. However, over time the team adds new user stories that affect the resources, costs, and time required for the project. McHugh et al. [35] find that teams have difficulties in accurately estimating unknown tasks.

9. Minimal documentation

Creating documentation is a challenge that many projects face [15, 54]. Ramesh et al. [11] mention that minimal documentation is a vital challenge that the Agile methodology poses to project teams. Lack of documentation raises communication gaps, and the gaps are exacerbated by large and global projects [22]. This is particularly acute in situations such as dispersed teams, large teams, and complex projects [12]. Because of incomplete,

inaccurate, or non-existing documentation, teams often make decisions based on poor intelligence [32, 57]. Drury-Grogan et. al [63] find that the lack of documentation results in poor decisions as teams have an incomplete understanding of the system's functionalities. Similarly, using case studies, Saito et al. [64] suggest that undocumented knowledge in Agile projects for software development is an important long-term challenge.

10. Incomplete nonfunctional requirements

Capturing nonfunctional requirements (NFRs) is a key challenge to the use of the Agile methodology for software development [12-14] [24, 45]. Inayat et al. [12] argue that user stories generally focus on system or product features that ignore NFRs such as security and scalability. Ramesh et al. [11] also mention this as a key challenge to producing requirements.

11. Incomplete and missing requirements

When the iterations are in large numbers, there is the possibility of missing important requirements [14]. High-level requirements are generally missing in Agile projects [53]. This omission usually happens because of a lack of access to all stakeholders and stakeholders' inability to communicate the requirements clearly [41].

12. Ambiguous requirements

Dikert, Paasivaara, and Lassenius [53] argue that Agile projects for software development are especially complicated because of ambiguous requirements. They also find that the tester often struggles to breakdown ambiguous requirements for testing. Torrecilla-Salinas et al. [65] show that the uncertainty in definitions of requirements is a hindrance to projects.

13. Requirements volatility

Although changes in requirements are an inherent part of the Agile methodology, frequent changes can cause trouble for the development team [12, 14]. These changes can increase costs that thus lead to failure [22]. The teams generally struggle to adapt to changes as there is a lack of tracking mechanisms for change management [32]. Any documents that the team produces in the initial stages can quickly become irrelevant because the

Agile principles encourage changes in requirements [66]. Hess et al. [57] find that a sudden change in requirements results in communication lapses. Inayat et al. [12] find that increased communication and clear specifications of requirements can resolve this issue. When a new change affects the existing design, the user stories and unit tests should be sufficient to address the change [67]. But Knauss [67] claims that as user stories represent a delta of the requirements, collapsing all the deltas is insufficient for understanding the overall features of the system. He also argues that this lack of understanding has a negative effect on testing. In dispersed global software development projects, management of changes in requirements is particularly challenging [47, 68].

14. Prioritizing requirements

In Agile projects, priorities change very fast, and these changes affect software development [69, 70]. Prioritizing the list of requirements is challenging as the list itself has to be flexible to reflect changing customer needs [37].

15. Inadequate requirements verification

Consistency checking or formal inspections are seldom performed during requirements engineering in Agile projects, which makes software development based on requirements lacking verification risky [11].

D. CHALLENGES RELATED TO USER STORIES

16. Detailed user stories not created

Teams usually do not describe user stories in much detail. User stories may not be detailed enough to capture vulnerabilities, bugs, unexpected termination, and undefined behavior [62]. This is especially a problem when new members join the project team [16].

17. User stories are not integrated

Managing user stories is a challenge when their number is large. Drury-Grogan et al. [63] find that linkages between user stories are difficult to maintain. Trkman et al. [25] suggest using business process models to better

understand the dependencies among user stories as the models can provide the missing context to better understanding those dependencies.

18. Difficulty in decomposing user stories

Teams often struggle to break down user stories to a size that facilitates estimation [53, 65]. Those studies also show that such a task is especially complicated with ambiguous requirements.

E. CHALLENGES RELATED TO TESTING

19. Availability of testing resources

The Agile methodology assumes that there is plenty of fast testing resources available in each iteration but this is generally not true [62]. Although many projects have adopted TDD methods [71], the testing team often struggles to breakdown ambiguous requirements for testing [53].

20. Reducing testing and test coverage

Getting the developers and testers to verify and to validate the code is difficult. In this respect, Petersen and Wohlin [37] find that the lack of independent verification affects the test coverage. They provide an example where designers can influence testers to only focus on parts of the system by arguing that the other parts do not need to be tested as they did not touch those parts.

F. OTHER CHALLENGES

21. Inadequate or inappropriate architecture and interfaces

Architecture receives little attention in the Agile methodology for software development that leads to bad design decisions [37]. Architectural decisions by the project team in the early cycles often becomes redundant as they identify new requirements and thus reworking the architecture increases the project cost significantly [11, 16]. Also, the overall architecture is hard to envision as understanding the dependencies of the parts of the system is

difficult [37]. Because eliciting the complete requirements upfront is a problem, then it becomes a considerable rework to design the interfaces of the application at the later stages of development [24].

22. External visibility on project tasks

McHugh et al. [35] find that making the project's progress visible to organizational members is difficult if they are not part of the Agile team. The non-team members lack visibility of the statuses of the project tasks, and thus non-team members are unaware of the reasons for their delays. Fagerholm et al. [30] show that having clear communication with non-team members is important for project success.

V. RESULTS OF THE INTERVIEWS

TABLE 3. Samples of the coding of the challenges to requirements engineering in the Agile methodology for software engineering.3 gives examples of the coding done on the interview transcripts to identify the challenges to requirements engineering.

TABLE 3. Samples of the coding of the challenges to requirements engineering in the Agile methodology for software engineering.

Participant ID (time in interview)	Coded fragment	Challenge ID	Challenge
E1 (2:00)	…transforming from Waterfall to Agile is a big challenge…	25	Waterfall mindset in Agile
E1 (3:18)	…external input coming continuously in their [developer's] way… is telling them that this is not the right thing to do… think about it in a different way...	13	Requirements volatility
E1 (3:30)	…there is, I think, this continuous friction between product/business [team] and the development [team]…	2	Breakdown of team communication and coordination

In Table 4, the theme column categorizes each challenge. The practitioner reference column identifies the interviewee that made a statement related to that challenge – statements are identified by the time passed since the start of the interview (e.g., E1 (6:56)). Note that several challenges were alluded to by more than one expert.

If a statement by an interviewee indicated a challenge that was not already identified in the literature review, then that new challenge was added to the list of 22. In this analysis, the interviewed practitioners identified three

new challenges that were not discussed in the literature review. They were *lack of vision in planning sprints*, *lack of domain and application knowledge*, and *waterfall mindset in Agile.* Thus, we ended up with 25 challenges in total. The interviews also confirmed the presence of 20 challenges (out of 22) that were identified in our literature review. For challenges 4 and 22, we did not find support for in the interviews. The theme column for challenges 21, 22, and 25 was empty as these challenges could not be grouped or categorized under a theme. The newly identified challenges 23 and 24 were categorized in the project team theme.

TABLE 4. Final list of challenges to requirements engineering in the Agile methodology for software engineering.

Challenge ID	Theme	Participant ID (time in interview)	Challenge
1	Project Team	E1: (13:25), F11 (2:21)	Team's lack of involvement and motivation
2	Project Team	E1 (15:46), F2 (7:55)	Breakdown of team communication and coordination
3	Project Team	E1 (17:36), E3 (9:06, 16:45), F2 (8:20)	Difficulty in managing dispersed teams
4	Project Team		Sharing of knowledge
5	Project Team	E2 (12:43), F4 (3:25)	Lack of management involvement
6	Customer Involvement	E1 (4:34), E5 (4:25, 4:57)	Difficulty in customer interaction
7	Customer Involvement	E1 (3:30), E2 (11:26, 20:12), F8 (2:09)	Customer inability and disagreement
8	Requirements Quality	E2 (7:41), E4 (5:14), E5 (2:31), F1 (7.37), F2 (1:26), F3 (1:21), F4 (2:13), F10 (4:39)	Difficulty in estimating time and costs
9	Requirements Quality	E1 (6:56, 9:57), E3 (0:50), E4 (8:31, 10:48), F2 (11:00)	Minimal documentation
10	Requirements Quality	F1 (10:11)	Incomplete non-functional requirements
11	Requirements Quality	E1 (3:18), E3 (2:04), F1 (3:08), F3 (1:03), F4 (5:01), F7 (4:45), F9 (1:30)	Incomplete and missing requirements
12	Requirements Quality	E1 (6:14), E3 (3:22), F1 (0:31), F3 (13:11), F5 (2:02), F6 (0.23),	Ambiguous requirements

		F7 (0.19), F9 (1:57), F10 (2:45)	
13	Requirements Quality	E1 (6:03, 8:03), F2 (2:03), F3 (1:12), F4 (6:02), F11 (0:11)	Requirements volatility
14	Requirements Quality	E2 (15:28), E3 (14:02), F1 (1:23), F2 (2:40), F3 (5:45), F11 (3:10)	Prioritizing requirements
15	Requirements Quality	E2 (19:28), E5 (3:16)	Inadequate requirements verification
16	User Stories	E1 (4:44), E3 (3:50), F1 (6:15), F3 (3:29), F5 (5:56), F6 (1.10), F7 (2:10), F9 (0:24), F10 (0:51, 4:16), F11 (3:46, 5:49)	Detailed user stories not created
17	User Stories	E2 (13:40), E3 (11:11), F3 (2:02), F4 (0:25), F5 (8:30), F7 (3:55), F9 (2:45)	User stories are not integrated
18	User Stories	E1 (4:18), E3 (12:13), E5 (1:42), F2 (1:13), F8 (5:32)	Difficulty in decomposing user stories
19	Testing	E1 (10:14), F2 (5:00)	Availability of testing resources
20	Testing	E1 (9:25, 9:43, 11:53), E2 (3:00), E5 (3:50), F1 (8:46), F2 (4:02), F4 (5:42), F9 (1:41)	Reducing testing and test coverage
21		E3 (2:45)	Inadequate or inappropriate architecture and interfaces
22			External visibility on project tasks
23	Project Team	F7 (5:13), F8 (0:30, 1:03), F9 (5:17)	Lack of vision in planning sprints
24	Project Team	F1 (7:00), F4 (7:01), F5 (6:15), F9 (4:06)	Lack of domain and application knowledge
25		E1 (2:00), E2 (1:37), E3 (4:13, 5:11), E4 (5:02),	Waterfall mindset in Agile

		E5 (1:19), F8 (1:30)	

VI. DISCUSSION

The previously mentioned literature review and survey studies [11-13] were published between 2010 and 2017. All these studies identified challenges related to our themes of customer involvement and requirements quality.

To investigate our general research question (i.e., *How can conceptual models address the challenges of requirements engineering in the Agile methodology for software development without conflicting with its values?*), we further categorized the five themes that we had identified in the literature review and confirmed through the interviews into two broad high-level themes: *challenges to requirements engineering related to human communication and collaboration* and *challenges related to understanding and clarifying the requirements*.

Challenges 1 to 5 and 23 to 24 are related to the project team theme, and challenges 6 and 7 are related to the customer involvement theme. These challenges refer to obstacles to effective requirements engineering that can be traced back to problems in human communication and collaboration that were observed in Agile projects. Challenges 8 to 15 are related to the requirements quality theme, and challenges 16 to 18 are related to the user stories theme. These challenges are different as they refer directly to problems with the requirements or their analysis or the user story technique as an artifact. We broadly categorized these challenges as related to understanding and clarifying the requirements. Further, we recognized the challenges related to testing (i.e., 19 and 20) and the challenges not categorized by a theme (i.e., 21, 22 and 25) as referring to other problems or obstacles than those categorized by the two broad themes.

These two broad themes are also explicitly discussed in the literature on the Agile methodology for software development. Based on a qualitative survey, Schön et al. [15] find that enhancing collaboration between the stakeholders, developers, and end users is important, while building a shared understanding of requirements from the users' perspective is not very well established in Agile projects. Collaboration [4] and shared understanding [15] are essential to developing Agile projects. Too much collaboration is harmful while too little is insufficient [4, 36]. The issue of collaboration becomes more critical when stakeholders including potential users are actively involved in developing Agile projects [15]. A lack of collaborative tools was observed to be a hindrance for sustained use of the Agile methodology [55].

In a case study on the Agile methodology, Moe et al. [72] find that the project team lacked a shared mental model on what the outcome of the project should be. Knowledge of the big picture from project goals could be wrongly understood by the stakeholders and the development teams [49]. As a system consists of components which in turn change rapidly, understanding the state of the system at any point of time can be difficult [67]. Thus a deep understanding of the domain and sharing that knowledge are crucial factors for the success of Agile projects [22]. Managing requirements for inter-dependent project teams is a challenge as it relates to the overall understanding and dependencies of the requirements [67]. When the domain knowledge is tacit and therefore difficult to articulate and share with others, requirements may appear unanalyzable and unstable [39]. Drury-Grogan et al. [63] find that in Agile projects, poor decisions are made because of an incomplete understanding of functionality. They argue that these poor decisions happen because the necessary data is lost, and decisions are forgotten because of the lack of documentation. Insufficient and inappropriate understanding of the requirements and quick changes in requirements are the leading reasons for the failure of global Agile projects for software development [41].

Coming back to the research question, conceptual models facilitate communication between users and analysts and support the analysts' understanding of the domain [73]. These two model purposes directly refer to the two higher level themes that we identified in our analysis of the challenges to requirements engineering in the Agile methodology for software development, that is, human communication and collaboration and understanding and clarifying the requirements. Table 5 shows that potentially, 20 of the identified challenges (i.e., those related to the project team (1 – 5, 23, 24) customer involvement (6 - 7), requirements quality (8 – 15), and user stories (16 – 18) themes) can be addressed using conceptual models.

TABLE 5. Linking the purpose of using conceptual models to the identified challenges to requirements engineering in the Agile methodology for software development.

Purpose of using conceptual models (according to [74])	Challenges to requirements engineering in the Agile methodology for software development
Communication – as a point of reference for requirements and clear understanding to various stakeholders	(1) Team's lack of involvement and motivation (2) Breakdown of team communication and coordination (3) Difficulty in managing dispersed teams (4) Sharing of knowledge (5) Lack of management involvement (6) Difficulty in customer interaction (7) Customer inability and disagreement (23) Lack of vision in planning sprints

	(24) Lack of domain and application knowledge
Understanding – high level of understanding of the system and purpose	(8) Difficulty in estimating time and costs (9) Minimal documentation (10) Incomplete nonfunctional requirements (11) Incomplete and missing requirements (12) Ambiguous requirements (13) Requirements volatility (14) Prioritizing requirements (15) Inadequate requirements verification (16) Detailed user stories not created (17) User stories are not integrated (18) Difficulty in decomposing user stories

VII. HOW TO USE CONCEPTUAL MODELS IN THE AGILE METHODOLOGY?

The above discussion points out that although the research has recognized the opportunities that conceptual models offer to address the challenges related to communication and domain understanding, the recommendations of how to incorporate conceptual modeling in Agile practices are quite varied and inconsistent (e.g., ranging from informal models like mind maps to more formal models like the use case diagrams in UML). The literature review and the expert interviews did not indicate that conceptual models were widely practiced in Agile. Despite little priority in documenting the Agile methodology, studies have discovered that Agile practitioners rate documentation as important and that too little documentation is available in their own projects [64]. Further, the software documentation that is available is often incomplete, inconsistent, difficult to maintain, and in practice out of date [75]. Williams [17] reports that overall not much documentation is prepared in the Agile methodology. Given that limited effort is spent on this documentation, for project team members to develop, maintain, and update conceptual models when sprints last only about two weeks is unrealistic, despite the potential benefits of using those models.

So, the question rises, how to use conceptual models in the Agile methodology? In this section we explore a vision on how to use conceptual models to address the challenges related to requirements engineering and management in the Agile methodology without contradicting its values. In our exploration, we focus on our more specific research question (i.e., *How can conceptual models address the challenges to requirements engineering in the Agile methodology for software development that are related to user stories*?) by developing a tentative answer through a demonstration experiment. We next present the feedback on this demonstration

that were given by interview participants F1 to F11. We end the section by reflecting on what is needed to provide more definite answers to the research questions that we investigated in this explorative study.

A. GENERATING CONCEPTUAL MODELS FROM USER STORIES – A DEMONSTRATION EXPERIMENT

We believe some conditions need to be fulfilled to introduce the use of conceptual models to the Agile methodology. This methodology prefers using working software over documentation, if conceptual models are created then they should be created within the current framework of requirements engineering and not as an additional activity requiring extra effort. Simply, team members cannot be forced to create conceptual models as an additional activity. Therefore, the creation of conceptual models must be automated to the largest possible extent. As the Agile methodology promotes continuous development of the software, the conceptual models should also be continuously updated such that they always codify the most current domain knowledge as reflected by the requirements.

In what follows, we demonstrate an example of the elements needed to construct conceptual models already being present in user stories. We also illustrate how conceptual models generated from user stories could be of use in Agile projects. Also, other researchers have suggested that organizing user stories and extracting information from them can be useful: “As a succinct, readily understandable description, a user story could promote shared understanding of a newly proposed CDS [Clinical Decision System] tool among diverse clinical and nonclinical stakeholders, resolving a common challenge” (pp. 1346) [23]. Daneva et al. [22] find that understanding the dependencies of the user stories is very important in the Agile methodology. They suggest maintaining traceability between user stories all the time that facilitates the vision of how a high-level business process translates into small chunks that are represented as user stories.

User stories represent the bird’s eye view of how everything fits together [63] in terms of requirements. The standard user story template is “As a <role>, I want <feature> so that <benefit>” [6]. We extend this template with behavior-driven development (BDD) scenarios that consist of a feature title, a user story, and a scenario that is defined by three segments – “Given <precondition>, when <triggering event>, then <postcondition>” [19]. Using not just user stories but also their associated BDD scenarios facilitates the generation of a wider set

of conceptual models, as we will illustrate in what follows. We focused on multiple types of conceptual models as a recent survey by van der Linden et al. [74] has shown that different types of UML diagrams and business process model and notation (BPMN) diagrams are the most common conceptual models used in practice. Surveys also indicate that practitioners use more than one conceptual model for different types of tasks [20]. This is because information systems are getting more complex and interrelated models can be used to offer different perspectives of the system and represent different aspects of it [76].

In our demonstration we focused on four types of conceptual models whose information could be identified in user stories and their associated BDD scenarios. These four types of models were a use case model, domain model, state machine, and a process model. Using the concepts of actor and use case, a use case model provides a description of the users' possible interactions with the system [77]. These interactions involve actions on objects that are described in a domain model. The domain model thus shows the concepts that a system needs to process and store data on their relationships and properties [78]. A state machine is a model that shows the different states that a single object, as an instance of a domain concept described in the domain model, passes through during its life in response to events [79]. A process model has a description of the possible orderings of these events and how they trigger actions on objects [80].

For our demonstration, we used the set of related user stories in Table 6 as our example and consider them as written for a software system that handles service requests.

TABLE 6. An example set of related user stories.

User story ID	User story and associated BDD scenario
1	As a customer[1], I want to create[4] a service request[3] so that I can have my problem solved. Given that the customer is active, when they decide to submit a service request[3], then a service request[3] is submitted[10].
2	As a support assistant[2], I want to accept[5] a service request[3] so that the team can start working on the service request. Given a service request is submitted[10], when the team agrees working on the service request, then the service request is open[11].
3	As a support assistant[2], I want to resolve[6] a service request[3] so that the customer's problem is solved. Given a service request[3] is open[11], when the team solves the problem described in the service request, then the service request is fixed[12].
4	As a customer[1], I want to approve[7] the service request[3] so that it can be closed. Given a service request[3] is fixed[12], when I approve the solution to my problem, then the service request[3] is closed[13].
5	As a customer[1], I want to reject[8] the service request[3] so that it can be reopened. Given a service request[3] is fixed[12], when

	I reject the solution to my problem, then the service request[3] is open[11].
6	As a customer[1], I want to cancel[9] a service request[3] so that the team can focus on other active requests. Given a service request[3] is submitted[10] or fixed[12], when the customer decides to cancel the service request, then the service request will be canceled[14].

Moreover, these segments of the user stories in Table 6 were annotated with numbers so that these numbers could be used to trace the mapping from user stories to conceptual models.

Figure 1 shows a use case model and a domain model that are based only on the information contained in the standard user story template. The use case model shows that customer and support assistant are the only two actors (i.e., roles in the user stories), and the actions that these two actors perform (i.e., use cases) are the features specified in the user stories. Thus, a use case model provides an overview of the roles and related features described in a related set of user stories and allows a visual grouping of user stories per role.

The domain model distinguishes among the objects to which the actions of the use case model are applied – in our case this is just the service request. The different roles are related via their actions to the objects, clearly showing which role wants to perform which action on which object. Like the use case model, the domain model only relies on the information captured by the role and features segments of the user stories but provides a clear visual overview of this information.

Like Figure 1, the models in Figure 2 can be traced to the user stories in Table 6. For example, the state machine shows that the "accept action" changes the state of the service request from submitted to open, and the "approve action" changes the state of the service request from fixed to closed. Similarly, the process model shows that the prerequisite of the accept action is the "create action" (by the customer), and the prerequisite of the approve action is the "resolve action" (by the support assistant). To generate these two models, we also need to document the pre- and postconditions in the BDD scenarios of the user stories.

How can these models now help in improving communication and domain understanding? Let's consider the following situation. It might not be clear to the project team when a customer is allowed to cancel a service request (i.e., user story 6). The state machine can provide a basis for discussion among the team members and with the customer to clarify what the actual expectation of the system is. The current interpretation obtained from the understanding of the state machine is that cancellation of a service request is only allowed before the service request is approved, however not in the state of "open". So, in a state of open, cancellation is not allowed,

while it is in states of “submitted” and “fixed”. Also, the process model shows that once the service request is accepted, it needs to be resolved and the customer cannot cancel it before the support assistants have done their work. Based on the use case model, a further discussion can be held on which type of user can cancel service requests. Is only the customer allowed to cancel service requests or is a support assistant also allowed to cancel based on certain conditions (e.g., when a customer repeatedly rejects the work performed to resolve the service request as is clearly shown by the resolve-reject loop in the state machine)?

Therefore, when the number of user stories increases, such insights on the user expectations and hence system requirements can be difficult to obtain purely based on the textual user stories themselves. Although the stakeholders might have developed individual mental models of the domain to be supported by the system, structured visual representations (such as Figures 1 and 2) can help to align these mental models consistently for all stakeholders. Further, the models can also be used to obtain an overall understanding of the requirements for members of the Agile team who join the project in later stages.

B. FEEDBACK FROM EXPERTS

In this subsection, we also evaluate whether the usefulness of conceptual models is acknowledged by expert practitioners of the Agile methodology. To understand the perceived usefulness, interview participants F1 to F11 were asked how the conceptual models could benefit the project team under the assumption that these models would be made available without the team having to invest effort in creating and updating them. Specifically, we asked what benefits were available from which type of conceptual model and whether a particular role in the project team benefited more than the rest of the team. The participants were shown the set of models (Figures 1 and 2) and the user stories (Table 6) but without explaining to them how the models were obtained from the user stories.

Table 7 shows the analysis of the perceived benefits of the models as mentioned by the participants. We mapped these benefits to the challenges to requirements engineering that we identified earlier in the literature review and interviews. As most benefits could be mapped to challenges for which we proposed that conceptual models could help (i.e., challenges 3, 4, 9, 11, 15, 16, 17, 18, 24; see Table 5), the expert opinions provided empirical support for our proposed approach.

TABLE 7. Mapping of potential benefits of conceptual models.

Participant ID (time in interview)	Model type	Potential user within the team	Benefit	(ID) Challenge
F1 (16:44, 32:39), F7 (27:44)	Process Model		Training	(4) Sharing of knowledge (24) Lack of domain and application knowledge
F1 (32:56)	Use Case Model	Product manager	To go back and review and relate the existing stories	(17) User stories are not integrated
F2 (18:01), F7 (09:30)	Use Case Model		Brings visibility	(9) Minimal documentation
F8 (17:32)	Use Case Model		To give visual representation of the solution.	(9) Minimal documentation
F3 (13:52)	Use Case Model	Developers	Gives you a high level view of the entire solution	(9) Minimal documentation
F8 (14:44)	Process Model		Can tell the flow and dependencies of the user stories	(17) User stories are not integrated
F5 (14:32)	Process Model		Get full understanding of the end to end process	(17) User stories are not integrated
F4 (18:19)	Use Case Model		To identify the number of user stories	(18) Difficulty in decomposing user stories
F9 (23:48)	Use Case Model		Can help in writing test cases	(19) Availability of testing resources
F4 (19:32)	State Machine		Can help to write end to end test scenarios.	(19) Availability of testing resources
F7 (10:02)	State Machine	Product owner	To verify the status of the stories that have been implemented. It can help to visualize what are the missing stories or the new stories that should be added.	(11) Incomplete and missing requirements (16) Detailed user stories not created
F4 (19:32)	State Machine	Testers and developers	Can look at it and see if there are scenarios that they have to cover for it when they are coding or testing	(20) Reducing testing and test coverage

F7 (12:32)	Process Model		Can help if it is a new initiative	(4) Sharing of knowledge (24) Lack of domain and application knowledge
F7 (16:42)	State Machine		Helpful for software enhancements	
F8 (14:44)	Process Model	Development and QA team	Develop a kind of navigational flow properly	(24) Lack of domain and application knowledge
F8 (15:21)	Use Case Model		Can help in defining the scenarios. It's a quick check to see whether all scenarios are covered	(20) Reducing testing and test coverage
F8 (6:30)	Process Model		Can help in identifying what role is going to perform what activity	(3) Difficulty in managing dispersed teams
F5 (17:32)	State Machine		To identify the validation criteria and the criteria for exceptions	(15) Inadequate requirements verification

C. REFLECTION

User stories are more than just an artifact of requirements. In the Agile methodology, teams plan and allocate user stories for implementation (e.g., in Scrum they are a key element in composing the product/spring backlogs that detail the implementation work to be performed). User stories thus decompose the system design into units whose implementation can be managed individually [8]. The conceptual models generated from user stories do not focus on the individual user story but span a set of related user stories. These models are not used for managing the implementation of each desired system feature individually but provide a visual overview of dependencies and relationships between individual user stories which is hard to obtain just based on the text which user stories basically comprise. The use case model and process model are types of conceptual models useful for analyzing requirements as we illustrated with the “cancel service request” scenario that was sketched in the demonstration experiment. Other types of conceptual model, like the state machine and especially the domain model, can also be useful for software design [9]. For instance, the business logic captured by user stories provides the basis for the domain model that can, during software design activities, be further extended to a class diagram. Here the advantage is that software classes can be implemented with functionality that relates to more than one user story that ensures an adequate modularization of the software. We did not explore this

use of conceptual models in the demonstration experiment but, for instance, referred to [81] who proposed a mapping of user stories into agent-oriented and object-oriented software architectures.

Regarding the demonstration experiment, Figures 1 and 2 illustrate some things about the models. First, these models are solely based on the information that is present in the user stories and their associated BDD scenarios. Therefore, some constructs that are usually found in these types of conceptual models are absent (e.g., attributes in the domain model, extends and adds relationships between use cases in the use case model). Second, to be an effective aid to communication and domain understanding, the conceptual models must be syntactically correct and semantically accurate as well as provide a pragmatically relevant and understandable representation of the domain. As these conceptual models are solely based on information captured in the user stories and BDD scenarios, the completeness and consistency of the user stories is important. Quality problems with the user stories will probably come to surface when the models are generated, hence hidden quality problems might be discovered when analyzing the models (e.g., when the graph shown in the state machine is not connected or when an end event in the process model cannot be reached from a start event). Third, we demonstrated that four types of conceptual models can be constructed using the information that is present in the user stories. We did not use other types of conceptual models, but they could certainly be explored in the future. For instance, future studies could investigate if a goal model could be constructed using the information in the benefit segment of the user stories. Fourth, if we had only used the original standard template of user stories (without the BDD scenarios), then it would not have been possible to construct the models that show and allow analyzing dependencies between user stories (i.e., the state machine and the process model).

Regarding the example scenario for the validation and possibly further elicitation of the "cancel service request" user story, we note that this scenario illustrates how visual conceptual models like process models and use case models can help address some of the challenges to requirements engineering for the Agile methodology that were mapped to the purposes of using conceptual models in Table 5, like *sharing of knowledge*, *incomplete and missing requirements*, and *inadequate requirements verification*. Mapping the purposes for using conceptual models to the high-level themes of the challenges to requirements engineering does not mean that the use of conceptual models is equally useful for all challenges that are categorized in these themes. The benefits mentioned by the experts did not cover all those challenges. For instance, for

challenges like *lack of management involvement*, *difficulty in estimating time and costs*, and *incomplete nonfunctional requirements*, it would be harder to demonstrate the usefulness of conceptual models. This usefulness also depends on the type of conceptual model generated from the user stories. For instance, in our demonstration experiment, all six user stories articulated desired system features that could be classified as functional requirements. In the case of nonfunctional requirements (e.g., "As a customer, I want to have 90% of my service requests resolved within 2 working days."), whether the generation and use of other types of models are possible could be explored (e.g., the NFR Framework for goal modeling and goal-oriented requirements engineering [82] may help address the challenge *incomplete nonfunctional requirements*).

Considering the conditions for using conceptual models in the Agile methodology for software development that we mentioned before, and as we now have demonstrated that the information captured by a set of related user stories and BDD scenarios is sufficient to create different types of conceptual models that are used in other methods to develop software (e.g., RUP), a natural direction for future research is to recommend the automatic extraction of the conceptual models from the set of user stories. For this purpose, appropriate algorithms and tools need to be developed. Natural language processing (NLP) techniques could be a good fit for this purpose. Using this support, any time user stories change, the extraction and model generation could easily be repeated to update the conceptual models. This way, the members of the team could focus on writing user stories, while the conceptual models would be available to them to support requirements engineering. The models could not only provide a basic documentation of the requirements to foster communication and shared domain understanding but could also help improve the completeness and consistency of the user stories and help verifying them. An early elaboration of these ideas to demonstrate their feasibility is found in [83].

We note that some tools have already been developed to generate conceptual models from textual descriptions of requirements (e.g., [84]). There are also a couple of tools that automatically extract specific types of conceptual models from user stories (e.g., the visual narrator shows the concepts and relationships extracted from user stories [85]). A recent systematic literature review analyzed 38 different studies on the application of NLP techniques to user stories, including research on generating models from user stories [85, 86]. To the best of our knowledge, current NLP-based solutions for generating conceptual models from user stories apply the original version of the user story template and not the version with BDD scenarios. We

believe that the information provided by the pre- and postconditions as captured in the BDD scenarios is essential for identifying, understanding, and analyzing the dependencies between user stories, as we showed with our demonstration. We have yet to come across research on generating process models or state machines from user stories.

VIII. CONCLUSION

In this paper, we have explored the use of conceptual models to address the challenges to requirements engineering and management in software development. We started with a literature review to update the current understanding of the challenges to requirements engineering for the Agile methodology. We also interviewed 16 seasoned practitioners of this methodology to validate and possibly extend the challenges documented in the literature. In total, we identified 25 different challenges, which we discussed in the paper. This up-to-date overview is more extensive and detailed than the challenges to requirements engineering discussed in other studies [11-14], which is the first contribution of our paper.

Next, to investigate our main research question, *how can conceptual models address the challenges of requirements engineering in the Agile methodology for software development without conflicting with its values?*, we performed a thematic analysis of the challenges grouping 22 of them first into 5 categories (i.e., project team, customer involvement, requirements quality, user stories, testing) and next in one of two higher order themes: challenges related to *human communication and collaboration* (i.e., project team and customer involvement categories) and challenges related to *understanding and clarifying requirements* (i.e., requirements quality and user stories categories) that covered a total of 20 of the 25 identified challenges. For both types of challenges, the literature suggests that conceptual models can be helpful as they promote both communication and collaboration, and shared domain understanding.

The potential benefits of using conceptual models in the Agile methodology are no guarantee that they will be adopted by practitioners as the effort involved in creating models may contradict Agile values and principles. Therefore, we continued outlining the conditions for adoption of models – the creation of models should fit within current requirements engineering and management related activities in Agile projects, should be automated, and models should be updated whenever requirements change.

To investigate how these conditions could be fulfilled, we focused on a second research question, *how can conceptual models address the challenges of requirements engineering in the Agile methodology for software development that are related to user stories?*, considering that the user story is the main artifact used in the Agile methodology and that the literature has shown the problems with using and managing user stories (i.e., our challenges in the user stories category). By means of a demonstration experiment, we showed that four types of conceptual model (i.e., use case model, domain model, state machine, process model) can be constructed solely based on the information captured by a set of related user stories (e.g., epic or theme in Scrum) provided that the user stories are extended with BDD scenarios that document pre- and postconditions for the actions described in the user stories. This demonstration of the feasibility of generating conceptual models from user stories, particularly for models that allow understanding and analyzing dependencies between user stories, is another contribution of this paper. To the best of our knowledge, the generation of models using information of BDD scenarios is novel.

To automate now the generation of models, we suggest relying on NLP techniques. The application of NLP techniques to user stories is not new (see [86] for a recently published exhaustive review), and we experimented ourselves with the idea in [83]. We suggest the further elaboration and exploration of that approach that could be guided by the insights provided in our paper as a valuable and viable avenue for further research on requirements engineering within the Agile context for software development.

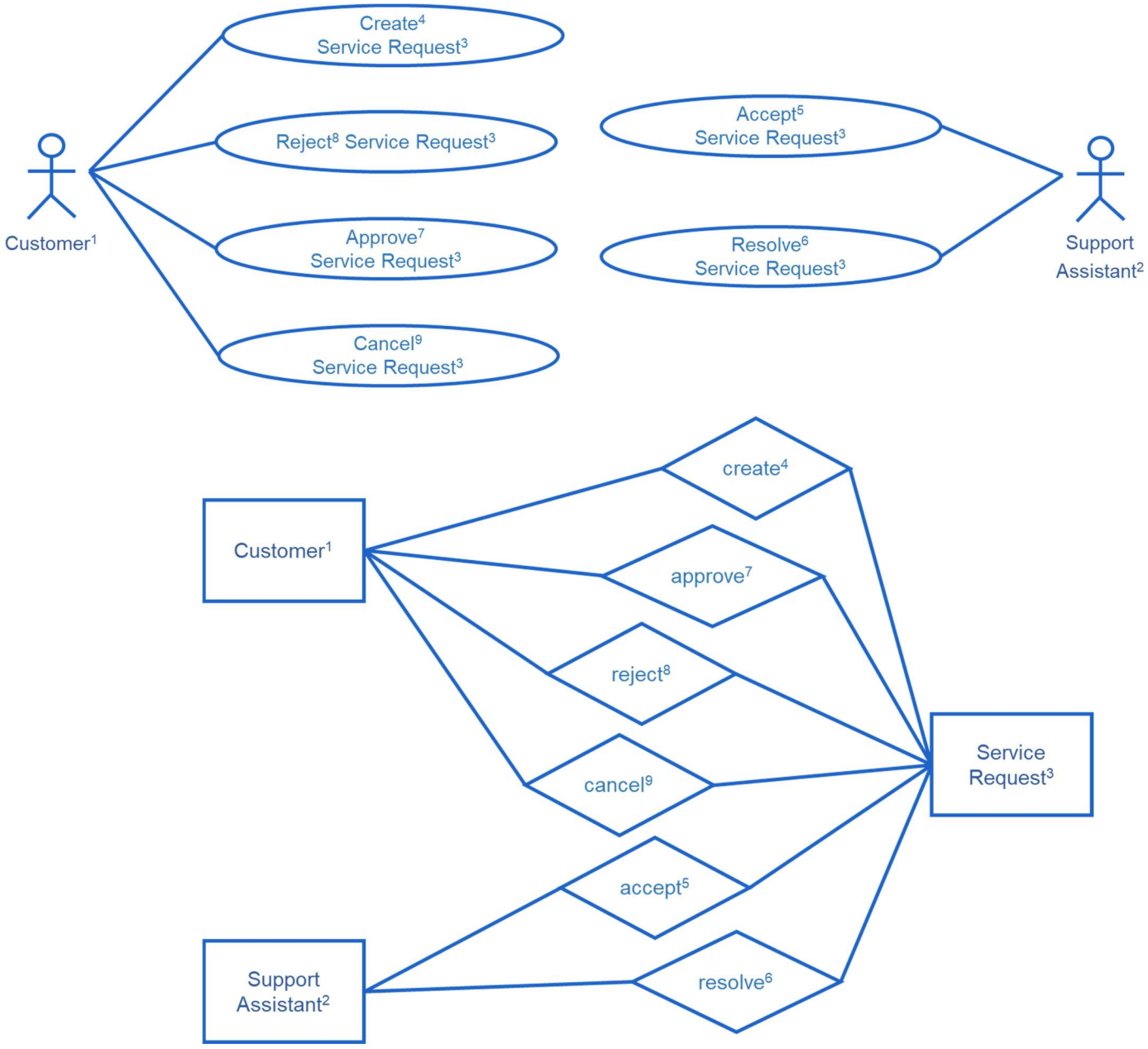
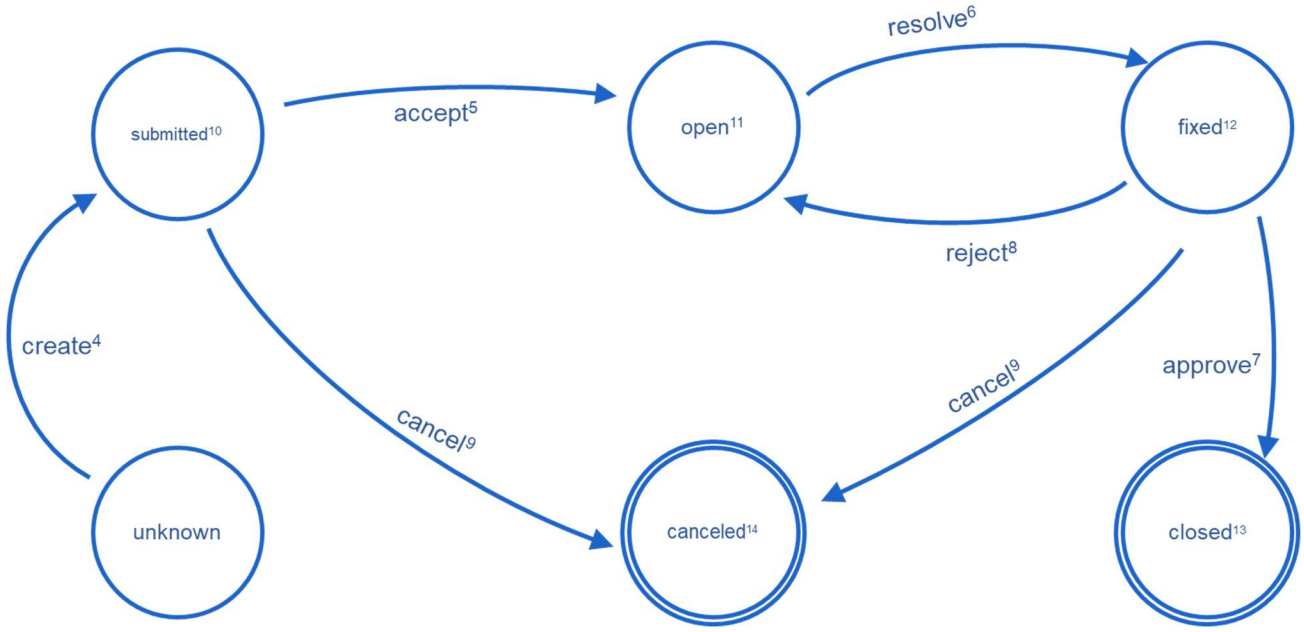


FIGURE 1. Use Case Model and Domain Model based on the user stories.



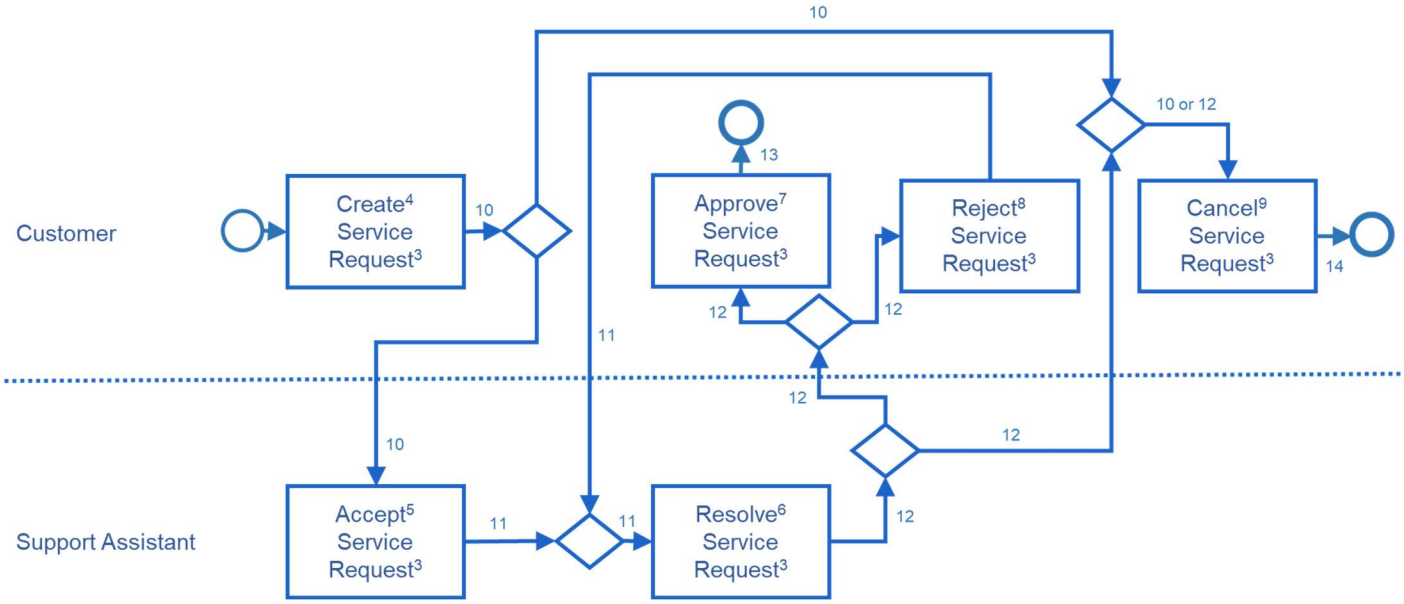


FIGURE 2. State Machine and Process Model based on the user stories.

REFERENCES

[1] W. Cunningham. (2001). *Manifesto for Agile Software Development*. [Online] Available: http://agilemanifesto.org/

[2] L. Cao and I. Ramesh, "Agile Requirements. Engineering Practices: An Empirical Study," *IEEE Software,* Article vol. 25, no. 1, pp. 60-67, 2008.

[3] K. Conboy, "Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development," *Information Systems Research,* Article vol. 20, no. 3, pp. 329-354, 2009.

[4] I. Inayat and S. S. Salim, "A framework to study requirements-driven collaboration among agile teams: Findings from two case studies," *Computers in Human Behavior,* Article vol. 51, no. Part B, pp. 1367-1379, 10/1/October 2015 2015.

[5] D. Leffingwell, *Agile Software Requirements: lean requirements practices for teams, programs, and the enterprise* (Agile Software Development Series). Boston: Addision-Wesley, 2011.

[6] M. Cohn, *User Stories Applied: For Agile Software Development*. Boston: Addison-Wesley, 2004.

[7] M. Murtazina and T. V. Avdeenko, "An Ontology-based Approach to Support for Requirements Traceability in Agile Development " *Procedia Computer Science* vol. 50, pp. 628-635, 2019.

[8] A. R. Amna and G. Poels, "Ambiguity in user stories: A systematic literature review.," *Information and Software Technology,* vol. 145, 2022.

[9] J. A. Hoffer, J. F. George, and J. S. Valacich, *Modern Systems analysis and design*, 6 ed. Pearson, 2011.

[10] Y. Wand and R. Weber, "Information Systems and Conceptual Modeling: A Research Agenda," *Information Systems Research,* vol. 13, no. 4, pp. 363-376, 2002.

[11] B. Ramesh, C. Lan, and R. Baskerville, "Agile requirements engineering practices and challenges: an empirical study," *Information Systems Journal,* Article vol. 20, no. 5, pp. 449-480, 2010.

[12] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband, "Review: A systematic literature review on agile requirements engineering practices and challenges," *Computers in Human Behavior,* Review Article vol. 51, no. Part B, pp. 915-929, 10/1/October 2015 2015.

[13] H. Elshandidy and S. Mazen, "Agile and Traditional Requirements Engineering: A Survey," *International Journal of Scientific and Engineering Research,* vol. 4, no. 9, pp. 473-482, 2013.

[14] S. Alam, S. A. Shah, S. N. Bhatti, and A. M. Jadi, "Impact and Challenges of Requirement Engineering in Agile Methodologies: A Systematic Review," *International Journal of Advanced Computer Science and Applications,* vol. 8, no. 4, pp. 411-420, 2017.

[15] E.-M. Schön, J. Thomaschewski, and M. J. Escalona, "Agile Requirements Engineering: A systematic literature review," *Computer Standards & Interfaces,* Article vol. 49, pp. 79-91, 1/1/January 2017 2017.

[16] V. N. Vithana, "Scrum Requirements Engineering Practices and Challenges in Offshore Software Development," *International Journal of Computer Applications,* vol. 116, no. 22, pp. 43-49, 2015.

[17] L. Williams, " Agile Software Development Methodologies and Practices," (Advances in Computers. Amsterdam, Netherlands: Elsevier, 2010, p.^pp. Pages.

[18] F. Anwer, S. Aftab, S. M. Shah, and U. Waheed, "Comparative analysis of two popular agile process models: Extreme Programming and Scrum," *International Journal of Computer Science and Telecommunications,* vol. 8, no. 2, pp. 1-7, 2017.

[19] J. F. Smart, *BDD in action: Behavior-Driven development for the whole software lifecycle*. New York: Manning Publications Company, 2014.

[20] J. Recker and P. Green, "How do Individuals Interpret Multiple Conceptual Models? A Theory of Combined Ontological Completeness and Overlap " *Journal of the Association for Information Systems,* vol. 20, no. 8, 2019.

[21] S. Sundararajan, M. Bhasi, and P. K. Vijayaraghavan, "Case study on risk management practice in large offshore-outsourced Agile software projects," *IET Software,* Article vol. 8, no. 6, pp. 245-257, 2014.

[22] M. Daneva *et al.*, "Agile requirements prioritization in large-scale outsourced system projects: An empirical study," *The Journal of Systems & Software,* Article vol. 86, pp. 1333-1353, 5/1/May 2013 2013.

[23] V. Kannan *et al.*, "User Stories as Lightweight Requirements for Agile Clinical Decision Support Development," *Journal of the American Medical Informatics Association,* vol. 26, no. 11, pp. 1344–1354, 2019.

[24] W. Helmy, A. Kamel, and O. Hegazy, "Requirements Engineering Methodology in Agile Environment," *International Journal of Computer Science,* vol. 9, no. 5, pp. 293-300, 2012.

[25] M. Trkman, J. Mendling, and M. Krisper, "Using business process models to better understand the dependencies among user stories," *Information and Software Technology,* Article vol. 71, pp. 58-76, 3/1/March 2016 2016.

[26] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative Research in Psychology,* vol. 3, no. 2, pp. 77-101, 2006.

[27] A. L. Strauss and J. Corbin, *Basics of qualitative research: Grounded theory procedures and techniques*. CA: Sage: Thousand Oaks, 1990.

[28] J. Recker, R. Holten, M. Hummel, and C. Rosenkranz, "How Agile Practices Impact Customer Responsiveness and Development Success: A Field Study," *Project Management Journal,* Article vol. 48, no. 2, pp. 99-121, 2017.

[29] K. Conboy, S. Coyle, W. Xiaofeng, and M. Pikkarainen, "People over Process: Key Challenges in Agile Development," *IEEE Software,* Article vol. 28, no. 4, pp. 48-57, 2011.

[30] F. Fagerholm, M. Ikonen, P. Kettunen, J. Münch, V. Roto, and P. Abrahamsson, "Performance Alignment Work: How software developers experience the continuous adaptation of team performance in Lean and Agile environments," *Information & Software Technology,* Article vol. 64, pp. 132-147, 2015.

[31] G. Alaa and G. Fitzgerald, "Reconceptualizing Agile Information Systems Development Using Complex Adaptive Systems Theory," *Emergence: Complexity & Organization,* Article vol. 15, no. 3, pp. 1-23, 2013.

[32] M. Drury, K. Conboy, and K. Power, "Obstacles to decision making in Agile software development teams," *The Journal of Systems & Software,* Article vol. 85, pp. 1239-1254, 6/1/June 2012 2012.

[33] F. K. Y. Chan and J. Y. L. Thong, "Acceptance of agile methodologies: A critical review and conceptual framework," *Decision Support Systems,* Article vol. 46, no. 4, pp. 803-814, 2009.

[34] B. Tessem, "Individual empowerment of agile and non-agile software developers in small teams," *Information and Software Technology,* Article vol. 56, pp. 873-889, 8/1/August 2014 2014.

[35] O. McHugh, Conboy, K., Lang, M., "Agile Practices: The Impact on Trust in Software Project Teams," *IEEE Software,* vol. 29, no. 3, pp. 71-76, 2012.

[36] S. Adolph, P. Kruchten, and W. Hall, "Reconciling perspectives: A grounded theory of how people manage the

process of software development," *The Journal of Systems & Software,* Article vol. 85, pp. 1269-1286, 6/1/June 2012 2012.
[37] K. Petersen and C. Wohlin, "A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case," *The Journal of Systems & Software,* Article vol. 82, pp. 1479-1490, 1/1/2009 2009.
[38] Y. Lindsjørn, D. I. K. Sjøberg, T. Dingsøyr, G. R. Bergersen, and T. Dybå, "Teamwork quality and project success in software development: A survey of agile development teams," *The Journal of Systems & Software,* Article vol. 122, pp. 274-286, 12/1/December 2016 2016.
[39] S. Kudaravalli, S. Faraj, and S. L. Johnson, "A Configural Approach to Coordinating Expertise in Software Development Teams," *MIS Quarterly,* Article vol. 41, no. 1, pp. 43-64, 2017.
[40] S. Sarker and S. Sarker, "Exploring Agility in Distributed Information Systems Development Teams: An Interpretive Study in an Offshoring Context," *Information Systems Research,* Article vol. 20, no. 3, pp. 440-461, 2009.
[41] J. Iqbal *et al.*, "Requirements engineering issues causing software development outsourcing failure," *PloS one,* vol. 15, no. 4, 2020.
[42] P. Heck and A. Zaidman, "A Systematic Literature Review on Quality Criteria for Agile Requirements Specifications," *Software Quality Journal,* vol. 26, pp. 127-160, 2018.
[43] S. V. Shrivastava and U. Rathod, "Risks in Distributed Agile Development: A Review," *Procedia - Social and Behavioral Sciences,* Article vol. 133, pp. 417-424, 5/15/15 May 2014 2014.
[44] S. C. Misra, V. Kumar, and U. Kumar, "Identifying some important success factors in adopting agile software development practices," *The Journal of Systems & Software,* Article vol. 82, pp. 1869-1890, 1/1/2009 2009.
[45] M. Brhel, H. Meth, A. Maedche, and K. Werder, "Exploring principles of user-centered agile software development: A literature review," *Information and Software Technology,* Review Article vol. 61, pp. 163-181, 5/1/May 2015 2015.
[46] W. Alsaqaf, M. Daneva, and R. Wieringa, "Quality requirements challenges in the context of large-scale distributed agile: An empirical study," *Information and Software Technology,* vol. 110, pp. 39-55, 2019.
[47] D. Llyod, R. Moawad, and M. Kadry, "A Supporting Tool for Requirements Change Management in Distributed Agile Development," *Future Computing and Informatics Journal,* vol. 2, pp. 1-9, 2017.
[48] K. Conboy and L. Morgan, "Beyond the customer: Opening the agile systems development process," *Information and Software Technology,* Article vol. 53, pp. 535-542, 5/1/May 2011 2011.
[49] R. P. Ghozalia, H. Saputraa, M. A. Nuriawana, Suharjitoa, D. N. Utamaa, and A. Nugrohoa, "Systematic Literature Review on Decision-Making of Requirement Engineering from Agile Software Development " *Procedia Computer Science,* vol. 157 pp. 274–281, 2019.
[50] P. Serrador and J. K. Pinto, "Does Agile work? — A quantitative analysis of agile project success," *International Journal of Project Management,* Article vol. 33, pp. 1040-1051, 7/1/July 2015 2015.
[51] C. Yang, P. Liang, and P. Avgeriou, "A Systematic Mapping Study on the Combination of Software Architecture and Agile Development," *The Journal of Systems and Software,* vol. 11, pp. 157-184, 2016.
[52] G. K. Hanssen, "A longitudinal case study of an emerging software ecosystem: Implications for practice and theory," *Journal of Systems & Software,* Article vol. 85, no. 7, pp. 1455-1466, 2012.
[53] K. Dikert, M. Paasivaara, and C. Lassenius, "Challenges and success factors for large-scale agile transformations: A systematic literature review," *Journal of Systems & Software,* Article vol. 119, pp. 87-108, 2016.
[54] P. Gregory, L. Barroca, H. Sharp, A. Deshpande, and K. Taylor, "The challenges that challenge: Engaging with agile practitioners' concern," *Information and Software Technology,* vol. 77, pp. 92-104, 2016.
[55] M. Senapathi and M. L. Drury-Grogan, "Refining a model for sustained usage of agile methodologies," *The Journal of Systems & Software,* Article vol. 132, pp. 298-316, 10/1/October 2017 2017.
[56] L. Vijayasarathy and D. Turk, "Drivers of agile software development use: Dialectic interplay between benefits and hindrances," *Information and Software Technology,* Article vol. 54, pp. 137-148, 1/1/2012 2012.
[57] A. Hess, P. Diebold, and N. Seyff, "Understanding Information Needs of Agile Teams to Improve Requirements Communication," *Journal of Industrial Information Integration,* vol. 14, pp. 3-15, 2019.
[58] S. Jayatilleke and R. Lai, "A systematic review of requirements change management," *Information and Software Technology,* Review Article vol. 93, pp. 163-185, 1/1/January 2018 2018.
[59] D. Dönmez and G. Grote, "Two sides of the same coin – how agile software development teams approach uncertainty as threats and opportunities," *Information and Software Technology,* Article vol. 93, pp. 94-111, 1/1/January 2018 2018.
[60] N. Ramasubbu, A. Bharadwaj, and G. Kumar Tayi, "Software Process Diversity: Conceptualization, Measurement, and Analysis of Impact on Project Perfromance," *MIS Quarterly,* Article vol. 39, no. 4, pp. 787-807, 2015.
[61] A. S. Campanelli and F. S. Parreiras, "Agile methods tailoring – A systematic literature review," *The Journal of Systems & Software,* Article vol. 110, pp. 85-100, 12/1/December 2015 2015.
[62] R. Chapman, N. White, and J. Woodcock, "What Can Agile Methods Bring to High-Integrity Software Development? Considering the issues and opportunities raised by Agile practices in the development of high-integrity software," *Communications of the ACM,* Article vol. 60, no. 10, pp. 38-41, 2017.
[63] M. L. Drury-Grogan, K. Conboy, and T. Acton, "Examining decision characteristics & challenges for agile software development," *The Journal of Systems & Software,* Article vol. 131, pp. 248-265, 9/1/September 2017 2017.
[64] S. Saito, Y. Iimura, A. K. Massey, and A. Antón, "Discovering Undocumented Knowledge Through Visualization of Agile Software Development Activities," *Requirements Engineering,* vol. 23, pp. 381–399 2018.
[65] C. J. Torrecilla-Salinas, J. Sedeño, M. J. Escalona, and M. Mejías, "Estimating, planning and managing Agile Web development projects under a value-based perspective," *Information and Software Technology,* Article vol. 61, pp. 124-144, 5/1/May 2015 2015.
[66] A. De Lucia and A. Qusef, "Requirements Engineering in Agile Software Development," *Journal of Emerging Technologies in Web Intelligence,* vol. 2, no. 3, pp. 212-220, 2010.
[67] E. Knauss, "The Missing Requirements Perspective in Large-Scale Agile System Development," *IEEE Software,* vol. 36, no. 3, pp. 9-13, 2019.
[68] T. Kamal, Q. Zhang, and M. Azeem, "Toward Successful Agile Requirements Change Management Process in Global Software Development: A Client–Vendor Analysis," *IET Software,* vol. 14, no. 3, pp. 265-274, 2020.
[69] 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, Article vol. 54, pp. 1107-1124, 10/1/October 2012 2012.
[70] A. Henriksen and S. A. R. Pedersen, "A qualitative case study on Agile Practice and Project Success in agile software projects," *Journal of Modern Project Management,* Article pp. 62-73, 2017.
[71] I. Nurdiani, J. Börstler, and S. A. Fricker, "The impacts of agile and lean practices on project constraints: A tertiary study," *Journal of Systems and Software,* vol. 119, no. Supplement C, pp. 162-183, 2016/09/01/ 2016.
[72] N. B. Moe, T. Dingsøyr, and T. Dybå, "A teamwork model for understanding an agile team: A case study of a Scrum project," *Information & Software Technology,* Article vol. 52, no. 5, pp. 480-491, 2010.
[73] C. H. Kung and A. Solvberg, "Activity modelling and behavior modelling of information systems," in *Information Systems Design Methodologies: Improving the Practice*, T. W. Olle, H. G. Sol, and A. A. Verrijn-Stuart, Eds. Amsterdam: North-Holland, 1986, pp. 145-171.
[74] D. v. d. Linden, I. Hadar, and A. Zamansky, "What Practitioners Really Want: Requirements for Visual Notations in Conceptual Modeling," *Software and Systems Modeling,* vol. 18, pp. 1813-1831, 2019.
[75] G. Garousi, V. Garousi-Yusifoğlu, G. Ruhe, J. Zhi, M. Moussavi, and B. Smith, "Usage and usefulness of technical software documentation: An industrial case study," *Information and Software Technology,* Article vol. 57, pp. 664-682, 1/1/January 2015 2015.
[76] S. Jabbari, A. Mohammad, and J. Recker, "Combined use of conceptual models in practice: An exploratory study," *Journal of Database Management,* vol. 28, no. 2, pp. 56-88, 2017.
[77] K. E. Kendall and J. E. Kendall, *Systems Analysis and Design*, 10 ed. Pearson, 2019.
[78] K. Markham, Mintzes, J., Jones, M., "The concept map as a research and evaluation tool: Further evidence of validity," *Journal of Research in Science & Teaching,* vol. 31, no. 1, pp. 91-101, 1994.
[79] A. Dennis, B. H. Wixom, and D. Tegarden, A. Dennis, Ed. *Systems Analysis and Design: An Object-Oriented Approach with UML*, 5 ed. Wiley, 2015.
[80] J. Mendling and J. Recker, "Towards systematic usage of labels and icons in business process models," in *12th International Workshop on Exploring Modeling Methods in Systems Analysis and Design*, CEUR, Montpellier, France, 2008, pp. 1-13.
[81] S. Heng, M. Snoeck, and K. Tsilionis, "Generating a Software Architecture out of User Stories and BDD Scenarios: Research Agenda," in *1st International Workshop on Agile Methods for Information Systems Engineering*, 2022, Leuven, Belgium: CEUR.
[82] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, *Non-functional Requirements in Software Engineering*. Kluwer Academic Publishing, 2000.
[83] A. Gupta, G. Poels, and P. Bera, "Creation of Multiple Models from User Stories- A Natural Language Processing Approach," in *Entity Relationship (ER) Conference*, Salvador, Bahia, Brazil, 2019, pp. 47-57, vol. 11787: Lecture Notes in Computer Science.
[84] R. Mesquita, A. Jacqueira, C. Agra, M. Lucena, and F. Alencar, "US2StarTool: generation i* models from user stories.," in *International i* Workshop (iStar)*, 2015.
[85] G. Lucassen, F. Dalpiaz, M. van der Werf, and S. Brinkkemper, "Visualizing User Story Requirements at Multiple Granularity Levels via Semantic Relatedness," in *Conceptual Modeling ER*, 2016, vol. 9974: Springer.
[86] I. K. Raharjana, D. Siahaan, and C. Fatichah, "User Stories and Natural Language Processing: A Systematic Literature Review," *IEEE Access,* vol. 9, pp. 53811–53826, 2021.

APPENDIX

TABLE A1. List of requirements engineering challenges and the IDs of the papers in which they occur (see Table A2).

ID	Category	Challenge Description	Ref1	Ref2	Ref3	Ref4	Ref5	Ref6	Ref7	Ref8	Ref9	Ref10	Ref11
1	Project Team	Team's lack of involvement and motivation	6	29	35	26	45	14	9	16	2	31	
2	Project Team	Breakdown of team communication and coordination	28	22	30	1	19	34	23	51	52		
3	Project Team	Difficulty in managing distributed teams	33	56	57	52	54						
4	Project Team	Sharing of knowledge	15	10	47	49	8						
5	Project Team	Lack of management involvement	10	12	43	6	38	17	32				
6	Customer	Difficulty in customer interaction	40	28	25	46	21	33	27	13			
7	Customer	Customer inability and disagreement	40	20	41	15	28	46					
8	Requirements Quality	Difficulty in estimating time and costs	40	28	5	7	45	13					
9	Requirements Quality	Minimal documentation	28	11	43	15	15	46	31	20	55		
10	Requirements Quality	Incomplete non-functional requirements	39	28	40	44	4						
11	Requirements Quality	Incomplete and missing requirements	39	12	52								
12	Requirements Quality	Ambiguous requirements	12	36									
13	Requirements Quality	Requirements volatility	39	20	24	15	42	53	54	11	20	46	50
14	Requirements Quality	Requirements prioritization	25	18	43	3							
15	Requirements Quality	Inadequate requirements verification	28										
16	User Stories	Detailed user stories not created	7	41									
17	User Stories	User stories are not integrated	37	15									
18	User Stories	Difficulty in decomposing user stories	12	36									
19	Testing	Availability of testing resources	7	12	48								
20	Testing	Reduction of testing and coverage	25										

21		Inadequate or inappropriate architecture and interfaces	28	25	41	44							
22		External visibility on project tasks	45	16									

TABLE A2. List of references used in Table A1.

Paper ID	Papers
1	Adolph, S., P. Kruchten and W. Hall (2012). "Reconciling perspectives: A grounded theory of how people manage the process of software development." The Journal of Systems & Software 85: 1269-1286.
2	Alaa, G. and G. Fitzgerald (2013). "Re-conceptualizing agile information systems development using complex adaptive systems theory." Emergence: Complexity & Organization 15(3): 1-23.
3	Bjarnason, E., K. Wnuk and B. Regnell (2012). "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: 1107-1124.
4	Brhel, M., H. Meth, A. Maedche and K. Werder (2015). "Exploring principles of user-centered Agile software development: A literature review." Information and Software Technology 61: 163-181.
5	Campanelli, A. S. and F. S. Parreiras (2015). "Agile methods tailoring – A systematic literature review." The Journal of Systems & Software 110: 85-100.
6	Chan, F. K. Y. and J. Y. L. Thong (2009). "Acceptance of Agile methodologies: A critical review and conceptual framework." Decision Support Systems 46(4): 803-814.
7	Chapman, R., N. White and J. Woodcock (2017). "What Can Agile Methods Bring to High-Integrity Software Development? Considering the issues and opportunities raised by Agile practices in the development of high-integrity software." Communications of the ACM 60(10): 38-41.
8	Conboy, K. (2009). "Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development." Information Systems Research 20(3): 329-354.
9	Conboy, K., S. Coyle, W. Xiaofeng and M. Pikkarainen (2011). "People over Process: Key Challenges in Agile Development." IEEE Software 28(4): 48-57.
10	Conboy, K. and L. Morgan (2011). "Beyond the customer: Opening the Agile systems development process." Information and Software Technology 53: 535-542.
11	Daneva, M., E. van der Veen, C. Amrit, S. Ghaisas, K. Sikkel, R. Kumar, N. Ajmeri, U. Ramteerthkar and R. Wieringa (2013). "Agile requirements prioritization in large-scale outsourced system projects: An empirical study." The Journal of Systems & Software 86: 1333-1353.
12	Dikert, K., M. Paasivaara and C. Lassenius (2016). "Challenges and success factors for large-scale Agile transformations: A systematic literature review." Journal of Systems & Software 119: 87-108.
13	Dönmez, D. and G. Grote (2018). "Two sides of the same coin – how Agile software development teams approach uncertainty as threats and opportunities." Information and Software Technology 93: 94-111.
14	Drury, M., K. Conboy and K. Power (2012). "Obstacles to decision making in Agile software development teams." The Journal of Systems & Software 85: 1239-1254.
15	Drury-Grogan, M. L., K. Conboy and T. Acton (2017). "Examining decision characteristics & challenges for Agile software development." Journal of Systems & Software 131: 248-265.
16	Fagerholm, F., M. Ikonen, P. Kettunen, J. Münch, V. Roto and P. Abrahamsson (2015). "Performance Alignment Work: How software developers experience the continuous adaptation of team performance in Lean and Agile environments." Information & Software Technology 64: 132-147.
17	Hanssen, G. K. (2012). "A longitudinal case study of an emerging software ecosystem: Implications for practice and theory." Journal of Systems & Software 85(7): 1455-1466.
18	Henriksen, A. and S. A. R. Pedersen (2017). "A qualitative case study on agile practices and project success in Agile software projects." Journal of Modern Project Management: 62-73.
19	Inayat, I. and S. S. Salim (2015). "A framework to study requirements-driven collaboration among Agile teams: Findings from two case studies." Computers in Human Behavior 51(Part B): 1367-1379.
20	Inayat, I., S. S. Salim, S. Marczak, M. Daneva and S. Shamshirband (2015). "Review: A systematic literature review on Agile requirements engineering practices and challenges." Computers in Human Behavior 51(Part B): 915-929.
21	Jayatilleke, S. and R. Lai (2018). "A systematic review of requirements change management." Information and Software Technology 93: 163-185.
22	Kudaravalli, S., S. Faraj and S. L. Johnson (2017). "A configural approach to coordinating expertise in software development teams." MIS Quarterly 41(1): 43-64.

23	Lindsjørn, Y., D. I. K. Sjøberg, T. Dingsøyr, G. R. Bergersen and T. Dybå (2016). "Teamwork quality and project success in software development: A survey of Agile development teams." The Journal of Systems & Software 122: 274-286.
24	Maruping, L. M., V. Venkatesh and R. Agarwal (2009). "A Control Theory Perspective on Agile Methodology Use and Changing User Requirements." Information Systems Research 20(3): 377-399.
25	Petersen, K. and C. Wohlin (2009). "A comparison of issues and advantages in Agile and incremental development between state of the art and an industrial case." The Journal of Systems & Software 82: 1479-1490.
26	Qumer, A. and B. Henderson-Sellers (2008). "A framework to support the evaluation, adoption and improvement of Agile methods in practice." The Journal of Systems & Software 81: 1899-1919.
27	Ramasubbu, N., A. Bharadwaj and G. Kumar Tayi (2015). "Software process diversity: conceptualization, measurement, and analysis of impact on project performance." MIS Quarterly 39(4): 787-807.
28	Ramesh, B., C. Lan and R. Baskerville (2010). "Agile requirements engineering practices and challenges: an empirical study." Information Systems Journal 20(5): 449-480.
29	Recker, J., R. Holten, M. Hummel and C. Rosenkranz (2017). "How Agile Practices Impact Customer Responsiveness and Development Success: A Field Study." Project Management Journal 48(2): 99-121.
30	Sarker, S. and S. Sarker (2009). "Exploring Agility in Distributed Information Systems Development Teams: An Interpretive Study in an Offshoring Context." Information Systems Research 20(3): 440-461.
31	Schön, E.-M., J. Thomaschewski and M. J. Escalona (2017). "Agile Requirements Engineering: A systematic literature review." Computer Standards & Interfaces 49: 79-91.
32	Senapathi, M. and M. L. Drury-Grogan (2017). "Refining a model for sustained usage of Agile methodologies." The Journal of Systems & Software 132: 298-316.
33	Shrivastava, S. V. and U. Rathod (2014). "Risks in Distributed Agile Development: A Review." Procedia - Social and Behavioral Sciences 133: 417-424.
34	Sundararajan, S., M. Bhasi and P. K. Vijayaraghavan (2014). "Case study on risk management practice in large offshore-outsourced Agile software projects." IET Software 8(6): 245-257.
35	Tessem, B. (2014). "Individual empowerment of Agile and non-Agile software developers in small teams." Information and Software Technology 56: 873-889.
36	Torrecilla-Salinas, C. J., J. Sedeño, M. J. Escalona and M. Mejías (2015). "Estimating, planning and managing Agile Web development projects under a value-based perspective." Information and Software Technology 61: 124-144.
37	Trkman, M., J. Mendling and M. Krisper (2016). "Using business process models to better understand the dependencies among user stories." Information and Software Technology 71: 58-76.
38	Vijayasarathy, L. and D. Turk (2012). "Drivers of Agile software development use: Dialectic interplay between benefits and hindrances." Information and Software Technology 54: 137-148.
39	Alam, S., Shah, S. A., Bhatti, S. N., & Jadi, A. M. (2017). Impact and Challenges of Requirement Engineering in Agile Methodologies: A Systematic Review. International Journal of Advanced Computer Science and Applications, 8(4), 411-420
40	Elshandidy, H., & Mazen, S. (2013). Agile and Traditional Requirements Engineering: A Survey. International Journal of Scientific and Engineering Research, 4(9), 473-482.
41	Vithana, V. N. (2015). Scrum Requirements Engineering Practices and Challenges in Offshore Software Development. International Journal of Computer Applications, 116(22), 43-49.
42	De Lucia, A., & Qusef, A. (2010). Requirements Engineering in Agile Software Development. Journal of Emerging Technologies in Web Intelligence, 2(3), 212-220.
43	Gregory, P., Barroca, L., Sharp, H., Deshpande, A., & Taylor, K. (2016). The challenges that challenge: Engaging with Agile practitioners' concern. Information and Software Technology, 77, 92-104.
44	Helmy, W., Kamel, A., & Hegazy, O. (2012). Requirements Engineering Methodology in Agile Environment. International Journal of Computer Science, 9(5), 293-300.
45	McHugh, O., Conboy, K., Lang, M. (2012). Agile Practices: The Impact on Trust in Software Project Teams. IEEE Software, 29(3), 71-76.
46	Hess, A., Diebold, P., & Seyff, N. (2019). Understanding Information Needs of Agile Teams to Improve Requirements Communication. Journal of Industrial Information Integration, 14, 3-15
47	Yang, C., Liang, P., & Avgeriou, P. (2016). A Systematic Mapping Study on the Combination of Software Architecture and Agile Development. The Journal of Systems and Software, 11, 157-184.
48	Nurdiani, I., Börstler, J., & Fricker, S. A. (2016). The impacts of Agile and lean practices on project constraints: A tertiary study. Journal of Systems and Software, 119(Supplement C), 162-183
49	Ghozalia, R. P., Saputraa, H., Nuriawana, M. A., Suharjitoa, Utamaa, D. N., & Nugrohoa, A. (2019). Systematic Literature Review on Decision-Making of Requirement Engineering from Agile Software Development Procedia Computer Science, 157 274–281.
50	Knauss, E. (2019). The Missing Requirements Perspective in Large-Scale Agile System Development. IEEE Software, 36(3), 9-13.

51	Heck, P., & Zaidman, A. (2018). A Systematic Literature Review on Quality Criteria for Agile Requirements Specifications. Software Quality Journal, 26, 127-160.
52	Iqbal, J., Ahmad, R. B., Khan, M., Fazal-E-Amin, A., S., N. N., M. H., A., & Shoaib, M. (2020). Requirements engineering issues causing software development outsourcing failure. PloS one, 15(4). doi:https://doi.org/10.1371/journal.pone.0229785
53	Kamal, T., Zhang, Q., & Azeem, M. (2020). Toward Successful Agile Requirements Change Management Process in Global Software Development: A Client–Vendor Analysis. IET Software, 14(3), 265-274.
54	Llyod, D., Moawad, R., & Kadry, M. (2017). A Supporting Tool for Requirements Change Management in Distributed Agile Development. Future Computing and Informatics Journal, 2, 1-9.
55	Saito, S., Iimura, Y., Massey, A. K., & Antón, A. (2018). Discovering Undocumented Knowledge Through Visualization of Agile Software Development Activities. Requirements Engineering, 23, 381–399
56	Misra, S. C., Kumar, V., & Kumar, U. (2009). Identifying some important success factors in adopting Agile software development practices. The Journal of Systems & Software, 82, 1869-1890. doi:10.1016/j.jss.2009.05.052
57	Alsaqaf, W., Daneva, M., & Wieringa, R. (2019). Quality requirements challenges in the context of large-scale distributed Agile: An empirical study. Information and Software Technology, 110, 39-55.