UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

JOÃO MAURÍCIO ALVES VALVERDE CARVALHO

REQUIREMENTS SPECIFICATION FOR DEVELOPERS IN AGILE PROJECTS:

EVALUATION BY ONE INDUSTRIAL CASE

CAMPINA GRANDE - PB
2021

JOÃO MAURÍCIO ALVES VALVERDE CARVALHO

REQUIREMENTS SPECIFICATION FOR DEVELOPERS IN AGILE PROJECTS: EVALUATION BY ONE INDUSTRIAL CASE

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Dr. Tiago Lima Massoni.

CAMPINA GRANDE - PB
2021

C331r Carvalho, João Maurício Alves Valverde.
Requirements specification for developers in agile projects: evaluation by one industrial case. / João Maurício Alves Valverde Carvalho. – 2021.

11 f.

Orientador: Prof. Dr. Tiago Lima Massoni.
Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

1. Software requirements specifications. 2. Agile software development. 3. Requirement enginer. 4. Qualitative research. 5. Grounded theory. 6. Brazilian developers. I. Massoni, Tiago Lima. II. Título.

CDU:004(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

JOÃO MAURÍCIO ALVES VALVERDE CARVALHO

REQUIREMENTS SPECIFICATION FOR DEVELOPERS IN AGILE PROJECTS: EVALUATION BY ONE INDUSTRIAL CASE

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Tiago Lima Massoni

Orientador – UASC/CEEI/UFCG

Professor Dr. Hyggo Oliveira de Almeida

Examinador – UASC/CEEI/UFCG

Professor Tiago Lima Massoni

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 20 de Outubro de 2021.

CAMPINA GRANDE - PB

RESUMO

A adoção de métodos de desenvolvimento ágil cresceu nos últimos anos. Os métodos ágeis tratam a Especificação de Requisitos de Software (SRS) de forma diferente dos métodos de desenvolvimento tradicionais. As 'User Stories' são uma das abordagens mais amplamente utilizadas para especificar requisitos em projetos ágeis. No entanto, estudos empíricos na indústria apontam que as USs são direcionadas aos clientes, cobrem apenas requisitos simples e funcionais visíveis para os usuários e não abordam requisitos de sistema e não funcionais. A abordagem de Especificação de Requisitos para Desenvolvedores (RSD) visa fornecer informações mais próximas das necessidades de desenvolvimento. Este artigo apresenta um estudo empírico avaliando a abordagem RSD em um caso industrial.

Requirements specification for developers in agile projects: Evaluation by one industrial case

João Maurício Carvalho
Orientador: Tiago Massoni
Departamento de Sistemas e Computação
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
joao.carvalho@ccc.ufcg.edu.br

ABSTRACT

Agile Software Development (ASD) adoption has grown in recent years. The agile methods treat the Software Requirements Specification (SRS) differently from the traditional development methods. User stories are one of the most widely used approaches to specifying requirements in agile projects. However, empirical studies in the industry point out that user stories are targeted to customers, only cover simple, functional requirements visible to the users, and do not address system and non-functional requirements. The Requirements Specification for Developers (RSD) approach aims to provide information closer to development needs. This paper presents an empirical study evaluating the RSD approach in an industrial case.

Keywords

Software requirements specifications, Agile software development, Requirement engineer

1. Introduction

The 15th Annual State of Agile Report, conducted by VersionOne[1], showed that software development teams' adoption of Agile methods has increased from 37% in 2020 to 86% in 2021. However, 46% of respondents pointed out inconsistencies in processes and practices are the most significant Agile adoption barrier. In particular, some studies have identified problems related to the requirements engineering activities, such as low availability of the customer, ambiguity of requirements artifacts, and the management of requirements due to frequent changes [2-5].

User stories have become the most commonly used requirements notation in agile projects[6-7]. By definition, a user story is an informal, general explanation of a software feature written from the end user's perspective. User stories' purpose is to articulate how a software feature will provide value to the customer. However, it does not address system and non-functional requirements, which are essential for an engineer to properly code, test, and maintain certain features.

Consequently, developers consider that software requirement specifications based on User Stories are brief, vague, ambiguous, and insufficient for capturing the complexities of the up-front design.[4,8]. The Requirements Specifications for Developers (RSD) seeks to provide an integrated view of the requirements linking the benefits of identifying the problem domain concepts (conceptual modeling), the visual representation of interface requirements (mockups), the business rules, nonfunctional requirements, and technical constraints (acceptance criteria). Motivated by this scenario, we conducted an empirical study, evaluating the RSD approach [9] content and how its uses affect teamwork.

Empirical studies in the context of requirements specification activities have been developed for a long time. However, most studies were not conducted in the context of Agile Software Development. According to Schön et al. [12], agile practices in Requirements Engineering need additional attention as a research theme. More empirical studies are required to understand the impact of agile methods in Requirements Engineering.

This study aimed to evaluate the impact of the RSD in practice and identify its strengths and limitations. Results showed the engineers believe the approach provides more effectiveness in visualizing and understanding requirements. However, a proprietary platform for using the method may be necessary to use its capabilities thoroughly.

The remainder of this paper was organized as follows: Section 2 presents an overview of the RSD approach. Section 3 describes the methodology used. Section 4 presents the results of the case study. Some related work is discussed in Section 5. Finally, Section 6 presents our conclusions and direction for future work.

2. RSD Approach overview

The RSD approach[9] replaces User Stories or other SRS methods in ASD. The approach structures customer needs and system requirements using a single view that integrates three perspectives. The first perspective models the business concepts (entities, attributes, and relationships). The second describes the acceptance criteria representing the business rules and technical requirements, NFR, or other constraints. The third describes the visual interface elements between the system and the user (mockups). Thus, it provides a more comprehensive requirements coverage when compared to User Stories, which only addresses user requirements.

The approach does not impose any particular tool to create and maintain artifacts and acceptance criteria items. RSD

can be used with XP, Scrum, or any other agile method where the client validates the requirements through working software, as established in the Agile Manifesto[11]. RSD approach focuses on the development team, and the RSR produced is not intended to be used as a mechanism for requirements validation with the customer.

2.1 Design and structure of the RSD approach

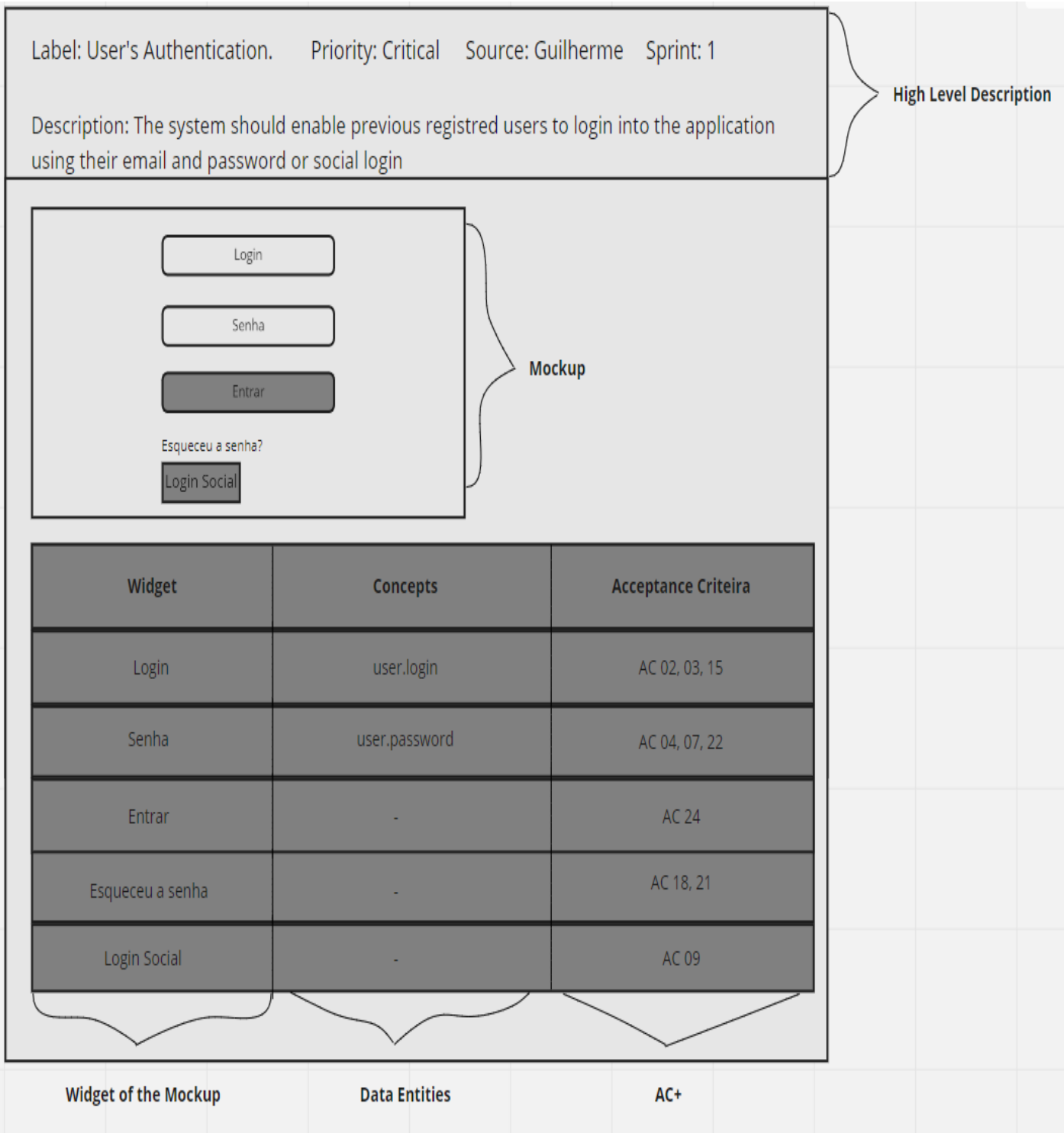

Fig 1. Structure of RSD artifact.

To illustrate the use of the RSD approach, Fig. 1 shows the RSD artifact related to the sign-in of a web application system. We can see that in addition to the application description and other data, we also have a mockup that defines an outline of how the page should be implemented and describes the acceptance criteria associated not only with the application but specifically for each widget. The RSD approach proposes the adoption of three well-established design practices that make the SRS targeted to the software engineer: Conceptual Modeling, Mockups Modeling and Specification of Acceptance Criteria (AC+, an extension of the Acceptance Criteria to be further explained in this section).

We can break the RSD artifact into five parts:

- High-Level Description
  - Identifies the requirement. This section includes Label, High-Level description, priority, requestor stakeholder, and sprint in which it will be implemented.
- Mockup
  - If applicable, the mockup allows visualization of the data and how they will be presented in the system, facilitating teamwork development.
- Widget
  - Presents the widget present in the mockup model.
- Data Entities
  - Presents the data entities and attributes extracted from the conceptual model, which are related to each widget.
- Acceptance Criteria
  - Presents the ID of the Acceptance Criteria. Acceptance Criteria are supposed to be re-used in the RSD Approach. They are written in another table and referenced by ids in the artifacts.

2.2 Acceptance Criteria Plus

An essential aspect of the RSD approach is Acceptance Criteria + (AC+), an extension of the Acceptance Criteria concept. AC+ defines not only business rules but also validation rules, interface, technical, or any other type of constraint necessary for the system coding. They can be classified into six types:

- **Business** - Represents a restriction related to the nature of the business.
- **Validation** - Represents some validation the application needs to perform but is not directly related to the business.
- **Interface** - Represents any restriction related to the user interface.
- **Technical** - Represents a technical restriction on how the solution should be implemented.
- **Non-Functional** - Represents concerns about tracking quality.
- **Other** - When it does not fit in any of the previous types.

Table 1
Acceptance Criteria Examples.

Id	Description	type
09	Login using google account must be enabled on Keycloak.	T
18	Email field must be auto-completed if previously done in the login screen	I
02	Username and email should be considered as login options	B
07	Before sending a request, frontend should validate that all fields were filled out.	V
24	Authentication must be available 99.99% of the time	N
22	'Password' field must hide what is being typed	I
25	To save, it is necessary that all required fields (*) are filled	V
26	The dropdown list must only display active clients	B
40	Failed webhooks notifications must be placed in an SQL table so that a new attempt can be made.	T
41	Notification retries must be made every 1 hour within the 24 hour period and discarded thereafter.	B

A requirement may have many AC+s with different priorities than can also be allocated to different sprints. New AC+s can be identified at any moment throughout the

development process. Some examples of AC+s are presented in **Table 1.**

Acceptance criteria are associated with id's so that reuse is possible. The type of language used in each varies according to the kind of AC+. For example, the AC+ of ID 40 is of the technical category and talks about handling errors in webhook notifications. Therefore, in the description, we can find a much more technical language, such as database technologies. Meanwhile, in business-related AC+, the language is more informal and does not go into technical implementation details. Like in AC+ #26 where the content of a dropdown is discussed.

Different requirements may reuse AC+s. Reuse also can occur in the same RSD. Two or more widgets may share the same AC+, for example. Some AC+ has nothing to do with widgets, such as AC+ regarding the implementation of algorithms or non-functional requirements, for example. In that case, they must be added in a row with the column 'widget' blank. Some Widgets are also not related to any data entities and should also be added in a row with the column ‘concept’ blank.

3. Methodology

The goal of the study was to qualitatively assess how the RSD approach works in practice. The studies followed a protocol that describes the procedures to collect and analyze the data. An interview script was built to guide the interviews with the software engineers. We intend to answer the following Research Questions:

- RQ1: How does the team evaluate the SRS produced using the RSD approach?
- RQ2: How does the RSD approach affect teamwork?

To properly evaluate the effectiveness of the RSD approach, we should consider the following aspects:

- The interaction between the development team and the customer. Collaboration with the customer is essential in every step of a development cycle. Furthermore, it is necessary to observe if the RSD approach decreases stakeholder dependence during the development stage.
- How the use of the approach influences the understanding of the demands. The objective of the RSD approach is, among many, to promote less ambiguity in requirements, promote reuse and avoid rework. To assess these factors, it is necessary to look at the effects of the approach within a real context.

Since there are some real-world variables to take into account, the reductionism of a controlled experiment would not be appropriate in our case. Consequently, we chose case study as the research method to evaluate the RSD approach. Thus, we conducted one empirical qualitative case study to assess the effectiveness of the approach in practice.

3.1 Participants

The case study was conducted over 1 month in the development of a customer service system for a digital marketing company. The scope of the project ranges from registering new customers to generating customer follow-up presentations. The development team consisted of four software engineers and the company uses traditional Scrum practices. The method used to specify requirements was User Stories from the beginning of the company's software projects.

3.2 Study Procedure

The team adopted the RSD approach in a new project for the period of two sprints. Each sprint lasted for two weeks. Several Scrum and XP practices were well established in the project, such as backlog, frequent releases, continuous integration, and retrospective. The developer team was composed of four engineers where one of them also played the software analyst role. All of them had at least two years of experience with ASD practices. The analyst elaborated the RSD in collaboration with the other engineers and the internal product owner during the planning session before the beginning of each sprint. Artifacts were updated throughout the sprint when discoveries were made regarding the requirement. The team managed the product backlog in the Jira software. All the RSD were written in Atlassian Jira. Javascript language was used to code the requirements. The analyst used the Pencil tool for modeling the mockups.

There was a conversation with the development team to introduce the approach. Examples were presented to demonstrate the differences between the RSD approach and user stories. A demo planning was also made to clarify doubts and make the team familiar with the approach. In the demo planning, we produced 6 RSD artifacts regarding a fictional game statics software. The paper's author is part of the engineering team and conducted the sprint planning where artifacts were made together with the entire team.

Observations were made during the daily meetings where the team was induced to comment on the impressions they were having using the RSD approach so far. Throughout two sprints of 2 weeks each, the team created 14 RSD documents. There was no comparison with another group not using the approach. Each document was evaluated by the software engineer in charge of coding it. They were asked to evaluate with a score from 1 to 5 in the following items: Objectivity, Clearness, and completeness of scope. This assessment was based on a list of factors that affect ASD requirements engineering[13].

Interviews were conducted individually with each of the developers involved at the end of the second sprint. Before the start, it was explained to the interviewee that the purpose of the interview was to assess the approach and suggest improvements. The questions were prepared sequentially, but the interviews were conducted freely. If the respondent showed interest in commenting on a specific point, the interviewer would skip to the related question and return to the script afterward. The following script was prepared to conduct the interviews:

- Q1: What roles did you play in the project?
- Q2: How were the requirements specified in the projects you worked on previously?
- Q3: Can you describe what you think is the ideal SRS for the developer?
- Q4: On a scale of one (inadequate) to five (very adequate), how do you assess the structure of RSD?

- Q5: Could you point out the main differences between the RSD approach and the methodologies you've used in the past?
- Q6: How do you assess the reuse of requirements in the project?
- Q7: Do you consider that the use of the approach reduced the frequency of interactions with the customer to resolve doubts during the development process?
- Q8: Did you notice anything specific about the approach that improved your performance?
- Q9: Did you notice anything specific about the approach that worsened your performance?
- Q10: Using a discrete scale (lower, equal, higher), how do you evaluate the effort required to specify using the RSD approach compared to others approaches you have used before? Do you think it is worth it?
- Q11: What changes would you like to make in the RSD approach?

The interviews were recorded and then transcribed. Relevant excerpts from the interviews were highlighted and given tags describing what was being said to group similar responses. The data collected in the interviews were then triangulated with the data obtained in the evaluations of the RSD artifacts and the observations made during the study to increase the credibility of the data.

4. Results

In this section, we present the evaluation results by organizing them by research questions.

4.1 How does the team evaluate the SRS produced?

The developer team evaluated each RSD artifact they worked on, considering two aspects: content and structure. During the two sprints, 14 artifacts were created with 55 AC+s in total.

- Content:

  All RSD artifacts were evaluated in compliance with all quality factors. In the Q8 interview script, mockups were presented as a performance improvement factor by 75% of respondents. 93% (13) of the artifacts were described without ambiguity. The artifact that was considered ambiguous did not clarify how the system should technically handle the deletion of a specific entity. It was not clear whether the system should do a hard delete or a soft delete.

  The team considered 86% of the RSD artifacts sufficient to be implemented without consulting complementary sources. One of the artifacts judged as insufficient to implement by the responsible developer had a lack of business AC+ related to an alternative use flow of the feature. The other one lacked technical AC+, as some complexities related to implementation had not been taken into account. Considering this, we can conclude that the reported problems are not related to the content of the approach but rather because the team neglected some aspects of certain features during the planning phase.

  We asked the engineers to compare the SRS produced by the RSD approach and the methodologies they have used in the past in Q5 and Q10. Old artifacts are accessible in the project management tool, but respondents were not asked to review them prior to the interview. All participants claimed that they used User Stories in past experiences to produce SRS. The fact that RSD is designed for the developer was mentioned in two of the interviews. Mockups were also mentioned in two interviews as a pro RSD factor. Only one respondent evaluated that the effort required to produce an RSD artifact is more significant than other methodologies, as highlighted by Interviewee #2: “The effort is higher because it is necessary to draw the mockup. However, I think it is worth it.”.

  When asked whether they thought that the SRS produced with the RSD approach was better than others they had already used, all responded positively. However, one of the engineers mentioned that he believed that the same result could be achieved using User Stories. He believes that although the structure of the approach provides a more aligned view of what developers need, the determining factor for projects with incomplete SRSs is because teams neglect the planning phases in developing a feature.

  “I believe the same result can be achieved using User Stories, but teams tend to neglect the effort required to produce good SRS" **Interviewee #4, Q5.**

- Structure

  Some questions were focused on how the RSD approach structures data. The approach was evaluated as very adequate by most respondents (3) and adequate by the other. In Q6, most respondents pointed out that it is hard to keep AC+ organized and reuse them without a proper tool to catalog and search them. Traceability was compromised because the team made the relationship between AC+ and requirements manually on sprint planning. The approach does not define any specific tools for organizing the requirements and acceptance criteria. One of the interviewees mentioned in Q11 that he would like the AC+ to be embedded within the artifact to avoid checking two different documents. On the other hand, the team did not report any problem regarding using the Pencil tool for elaboration of mockups.

  "I think the RSD structure itself is fine, but a tool of its own would be beneficial since it is difficult to catalog and reuse AC+ manually. Also, it is painful to have to check the content of AC+ in another document. So it would be better if they were embedded in the requirements". **Interviewee #4, Q11.**

  In Q8, mockups were mentioned as a factor that increases engineers' productivity, as they have a simple and objective view of what should be implemented.

4.2 How does the RSD approach affect teamwork?

Only one engineer reported the effort to create an RSD artifact as higher than other approaches. The effort was considered higher because it is necessary to draw the mockups. However, the engineer pointed out it was worth it because mockups help visualize the feature. He pointed out that despite believing that more significant effort is needed in the planning phase, the effort is reduced in the development phase.

In Q8, mockups were mentioned as a factor that increases engineers' productivity, as they have a simple and objective view of what should be implemented. When asked if they thought the approach reduced the frequency of interaction with stakeholders during the development process, almost all respondents indicated that using the approach decreased the frequency. Only one engineer pointed out that he did not notice any differences in this aspect. He mentioned that although mockups are good for visualizing features, interactions with stakeholders are still necessary for detail alignment. Some details are only observed when the developer is going to code the requirement, and therefore, it is necessary to consult stakeholders and add new information to the RSD artefact in the middle of the development process.

Acceptance tests were done by the internal Product Owner using the RSD as a reference. Some interviewees mentioned that this helped the feature validation process and the understanding of what was wrong or missing. Previously, acceptance tests were done arbitrarily, only considering how the internal Product Owner imagined the final solution. Respondents pointed out that the RSD artifact helped them have a more transparent and accurate idea of what was expected and reduced rework. When a feature was not accepted, the product owner indicated to the developer which AC+ was missing or wrongly implemented.

5. Related Work

In this section, we discuss some other Empirical Studies to evaluate the effectiveness of the RSD approach. One work assessing the RSD approach in two industrial cases was identified. In a work published in 2019[10], the approach was conducted for 12 months in one of the projects and 3 in the other. The engineer responsible for coding the feature evaluated the artifact individually, and one interview was conducted with each engineer at the end of the study. The study interviewed 14 engineers in total, and the result shows that in the long term, the lack of a specific tool to use the approach hinders its use. On the other hand, as in our study, the results regarding the content of the RSD artifacts were positive. One of the case studies pointed out that the team tends to neglect AC+s related to non-functional requirements. Therefore, it was suggested that the approach have an initial catalog of non-functional requirements.

6. Conclusions

The study aimed to evaluate the use of the RSD approach in practice and identify its strengths and limitations. The details of the study procedures have been detailed and can be used in other contexts to develop new studies.

For RQ1, the results show that the approach met the engineers' expectations and provided a more suitable view of what should be coded. The interviewed engineers considered that artifacts produced using the RSD approach are more suitable for implementation than the previously used approaches.

For RQ2, the result suggests that using the method adds little or no additional effort to the specification activity and can reduce the effort to code, test, and maintain software. It was also suggested that mockups might lessen the frequency of iterations with stakeholders during the development process. On the other hand, the team found it difficult to reuse AC+ completely. The fact that there is no specific tool for using the approach makes the process of cataloging and searching for acceptance criteria costly. The fact that AC+ was not embedded in the artifact page emerged as a factor in decreasing productivity, as it is necessary to check two separate documents.

This paper and related works show the RSD approach promotes a promising technique for specifying requirements in agile software development. However, a tool must be explicitly created for the application of the methodology. For example, an open-source application could be developed initially just as a repository to store and search AC+ and evolve into a complete platform where artifacts are created and stored in one place.

More evaluations of the RSD approach are still required to assess, for example, its impact on knowledge transfer between team members, how it works in different contexts like open-source projects, and how it performs with large distributed teams. In addition, an experiment directly comparing the approach with other methodologies already consolidated in the market would be interesting to understand the advantages of its use better.

7. References

[1] Version One. The 15th Annual State of Agile Report, Technical Report, Version One, 2021 accessed on November 10, 2021.,

[2] R. Kasauli, G. Liebel, E. Knauss, S. Gopakumar and B. Kanagwa, "Requirements Engineering Challenges in Large-Scale Agile System Development," 2017 IEEE 25th International Requirements Engineering Conference (RE), 2017, pp. 352-361, doi: 10.1109/RE.2017.60.

[3] M. Daneva, E. Van Der Veen, C. Amrit, S. Ghaisas, K. Sikkel, R. Kumar, N. Ajmeri, U. Ramteerthkar, R. Wieringa, Agile requirements prioritization in large-scale outsourced system projects: an empirical study, J. Syst. Softw. 86 (5) (2013) 1333–1353, doi:10.1016/j.jss.2012.12.046.

[4] A. Read and R. O. Briggs, "The Many Lives of an Agile Story: Design Processes, Design Products, and Understandings in a Large-Scale Agile Development Project," 2012 45th Hawaii International Conference on System Sciences, 2012, pp. 5319-5328, doi: 10.1109/HICSS.2012.684.

[5] S. Wagner, D. Fernández, M. Felderer, M. Kalinowski, Requirements engineering practice and problems in agile projects: Results from an international survey, in: Proceedings of the XX Iberoamerican Conference on Software Engineering, Buenos Aires, Argentina., 2017, pp. 389–402.

[6] Kassab M (2015) The changing landscape of requirements engineering practices over the past decade. In: Proceedings of the IEEE international workshop on empirical requirements engineering (EmpiRE). IEEE, pp 1–8.

[7] Wang X, Zhao L, Wang Y, Sun J (2014) The role of requirements engineering practices in agile development: an empirical study. In: Proceedings of the Asia Pacific requirements engineering symposium (APRES), CCIS, vol 432, pp 195–209

[8] Nik Nailah Binti Abdullah, Shinichi Honiden, Helen Sharp, Bashar Nuseibeh, and David Notkin. 2011. Communication patterns of agile requirements engineering. In Proceedings of the 1st Workshop on Agile Requirements Engineering (AREW '11). Association for Computing Machinery, New York, NY, USA, Article 1, 1–4. DOI:https://doi.org/10.1145/2068783.2068784

[9] Medeiros, Juliana & Vasconcelos, Alexandre & Goulão, Miguel & Silva, Carla & Araújo, João. (2017). An approach based on design practices to specify requirements in agile projects. 10.1145/3019612.3019753.

[10] Medeiros, Juliana & Vasconcelos, Alexandre & Silva, Carla & Goulão, Miguel. (2019). Requirements Specification for Developers in Agile Projects: Evaluation by two Industrial Case Studies. Information and Software Technology. 117. 106194. 10.1016/j.infsof.2019.106194.

[11] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. (2001). Manifesto for Agile Software Development Manifesto for Agile Software Development.

[12] E.-M. Schön, D. Winter, M.J. Escalona, J. Thomaschewski, Key challenges in agile requirements engineering, in: International Conference on Agile Software Development, Springer, 2017, pp. 37–51, doi:10.1007/978-3-319-57633-6_3.

[13] Medeiros, Juliana & Vasconcelos, Alexandre & Silva, Carla & Goulão, Miguel. (2018). Quality of software requirements specification in agile projects: A cross-case analysis of six companies. Journal of Systems and Software. 142. 10.1016/j.jss.2018.04.064.