Evaluating Software Developers' Acceptance of a Tool for Supporting Agile Non-Functional Requirement Elicitation

Felipe Ramos[§], Antônio Pedro[¶], Marcos Cesar[¶], Alexandre Costa[§],
Mirko Perkusich[¶], Hyggo Almeida[‡] and Angelo Perkusich[‡‡]
Intelligent Software Engineering Group, Federal University of Campina Grande,
Campina Grande - PB, Brazil, Zip Code 58429-140
[§]CAPES Foundation, Ministry of Education of Brazil, Brasilia - DF, Zip Code 70.040-020
[§]{feliperamos, antonioalexandre}@copin.ufcg.edu.br,
[¶]{antonio.abreu, marcos.cesar, mirko.perkusich}@embedded.ufcg.edu.br
[‡]hyggo@dsc.ufcg.edu.br, [‡‡]perkusic@dee.ufcg.edu.br

***Abstract*—Due to the need for flexibility to requirements changes, agile software development methods have been attracting the attention of academic and industrial domains. Unlike traditional approaches, agile methods focus on the rapid delivery of business value to customers through empirical and incremental development processes. Despite being effective in delivering quality functional requirements, agile practices generally neglect non-functional requirements until the later stages of software development. However, neglecting non-functional requirements during requirements analysis can lead to project failures. In this paper, we present the NFRec tool, which aims to support software developers in the elicitation of non-functional requirements in the context of agile software development. Additionally, we report the results from a case study to evaluate the acceptance of the NFRec tool from the point of view of software developers of four projects from a Brazilian software company. To gather information about the tool acceptance, we applied a questionnaire based on the indicators from the Technology Acceptance Model. Overall, the four teams considered the NFRec tool useful and easy to use for supporting the management of non-functional requirements in agile projects.**

Keywords*—Non-functional requirements; agile requirement engineering; supporting tool; empirical study; technology acceptance.

I. Introduction

Since the declaration of the Agile Manifesto in 2001, academic [8] and industrial [16] communities have devoted considerable attention to agile software development (ASD) methods, such as Scrum and eXtreme Programming. One of the key aspects of the ASD is to rapidly adapt to volatile requirements [8], following a development process that places a great emphasis on frequent delivery of business value for customers [8].

Although agile practices are considered effective in the delivering of valuable functional requirements (FRs) [14], non-functional requirements (NFRs) are commonly neglected until the later stages of software development [14]. However, neglecting NFRs can lead to software failure [2], since NFRs are often more critical to determine the perceived success or failure of a software product than FRs [9].

By addressing this problem, some studies proposed techniques to support the NFR elicitation in the agile context [10], [11], [12], [13]. Maiti and Mitropoulos [12], [13] presented a methodology to collect NFRs metadata from software requirements artifacts such as documents and images available in the initial stages of projects. Farid et al. [10], [11] proposed a solution based on decision trees to predict NFRs from future iterations of agile projects based on the metadata collected with the methodology proposed in Maiti and Mitropoulos [12], [13]. However, to the best of our knowledge, none of these works proposed the use of historical data of previous projects to provide suggestions of NFRs for ongoing ones. Additionally, there is a lack of empirical evaluations of proposed solutions in real industrial environments.

In our previous work [15], we proposed a non-functional requirement recommendation system (RS) for Scrum-based projects, which is based on the analysis of information from past projects. As a result, our solution achieved a recall rate of up to 81%, showing the feasibility of automating the definition of NFRs through historical data of Scrum-based projects. To allow the applicability of the proposed recommendation system on industrial environments, we developed a tool based on an improved version of the solution presented in [15], called NFRec.

In the current paper, we focus on the empirical assessment of the NFRec tool and report the results obtained through a case study to evaluate its acceptance in a real industrial environment. For matters of validation, our work focus on Scrum, which is the most popular agile method [17]. As a result, the subjects of the case study considered the NFRec tool useful and easy to use for supporting the management of non-functional requirements in agile projects.

This paper is organized as follows. Section II presents a background and a motivation to use a supporting tool for NFRs elicitation in ASD context. Section III shows the NFRec tool. Section IV presents details about the empirical evaluation. Section V discusses the threats to validity. Finally, Section VI details our conclusions and perspectives for future works.

DOI reference number: 10.18293/SEKE2019-107

II. Background and Motivation

Non-functional requirements play a key role in the successful development of software products [2]. Moreover, they are considered as a factor of differentiation among software products that present similar FRs [1].

In this sense, Bourimi et al. [3] proposed a framework that aims to conceptually impose the early consideration of NFRs, by considering the adoption of the stakeholder of NFRs in the Scrum team. In contrast, Farid [7] proposed a framework for modeling NFRs in the context of ASD.

Although the presented techniques provide a means for identifying NFRs on agile projects, there is still room for improvement. For example, solutions such as the ones proposed by Bourimi et al. [3] and Farid [7] present only a conceptual reinforcement or/and the addition of artifacts and roles in the agile process. Therefore, no solutions are proposed to support automated NFR elicitation. In contrast, solutions presented by Farid et al. [10], [11] and Maiti and Mitropoulos [12], [13] require that project documents and images are available on early stages of the software development, which is not common when dealing with ASD. The presented needs have been considered in the NFRec tool.

III. NFRec Tool

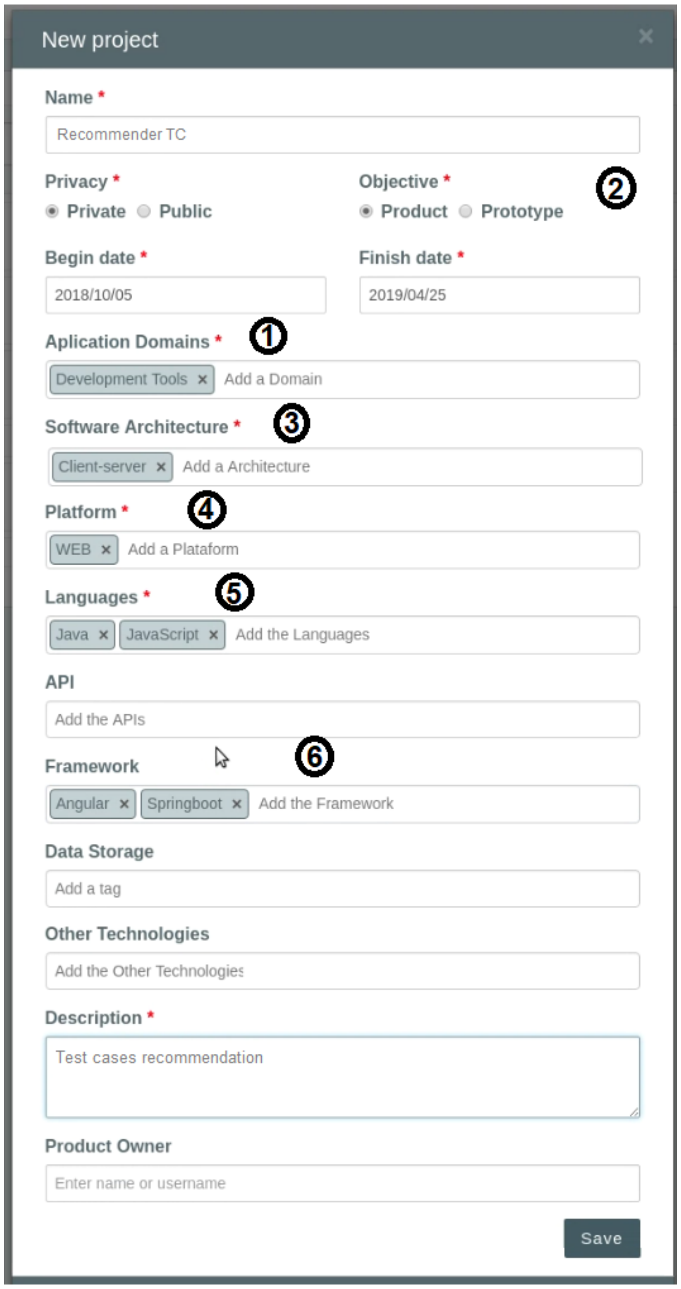
In this section, we present the NFRec tool, which aims to support developers in the early elicitation of NFRs during the Sprint Planning Meetings[1]. NFRec comprises the following two activities: (i) Structuring of Project Information; and (ii) NFR Recommendation.

A. Structuring of Project Information

To enable the generation of the NFRs recommendations, it is necessary to structure the projects' information to guarantee the retrieval of information by the recommendation system. In the NFRec tool, user stories and project profiles are structured based on the assignment of categories and tags.

The first activity accomplished in the NFRec tool by Scrum teams is to create project profiles based on the assignment of tags referring to the five factors that directly affect the definition of NFRs [15], i.e., platform (e.g., web, mobile, embedded, desktop, etc.), project domain (e.g., health, banking, etc.), project objective (e.g., product or prototype), software architecture (e.g., client-server, MVC, multilayered, etc.), and technology tags, which represent basic technologies for the development of a software product (e.g., programming language, database, etc.). In Figure 1, we present an example of a structured project profile created in the NFRec tool. The project is from the domain *Development Tools* (1), its objective is to develop a *Product* (2), the software architecture is *Client-server* (3), the platform is *Web* (4) and the technologies used are the programming languages *Java* and *JavaScript* (5) and the frameworks *Angular* and *Spring Boot* (6). We highlight that previously stored tags must be reused by Scrum teams to avoid data inconsistencies.

Besides project profiles structuring, all user stories (USs) specified in the NFRec tool must be categorized to enable the retrieval of information by the RS. Therefore, during the Sprint



New project

Name *
Recommender TC

Privacy *
● Private ○ Public

Objective *
● Product ○ Prototype

Begin date *
2018/10/05

Finish date *
2019/04/25

Aplication Domains *
Development Tools × Add a Domain

Software Architecture *
Client-server × Add a Architecture

Platform *
WEB × Add a Plataform

Languages *
Java × JavaScript × Add the Languages

API
Add the APIs

Framework
Angular × Springboot × Add the Framework

Data Storage
Add a tag

Other Technologies
Add the Other Technologies

Description *
Test cases recommendation

Product Owner
Enter name or username

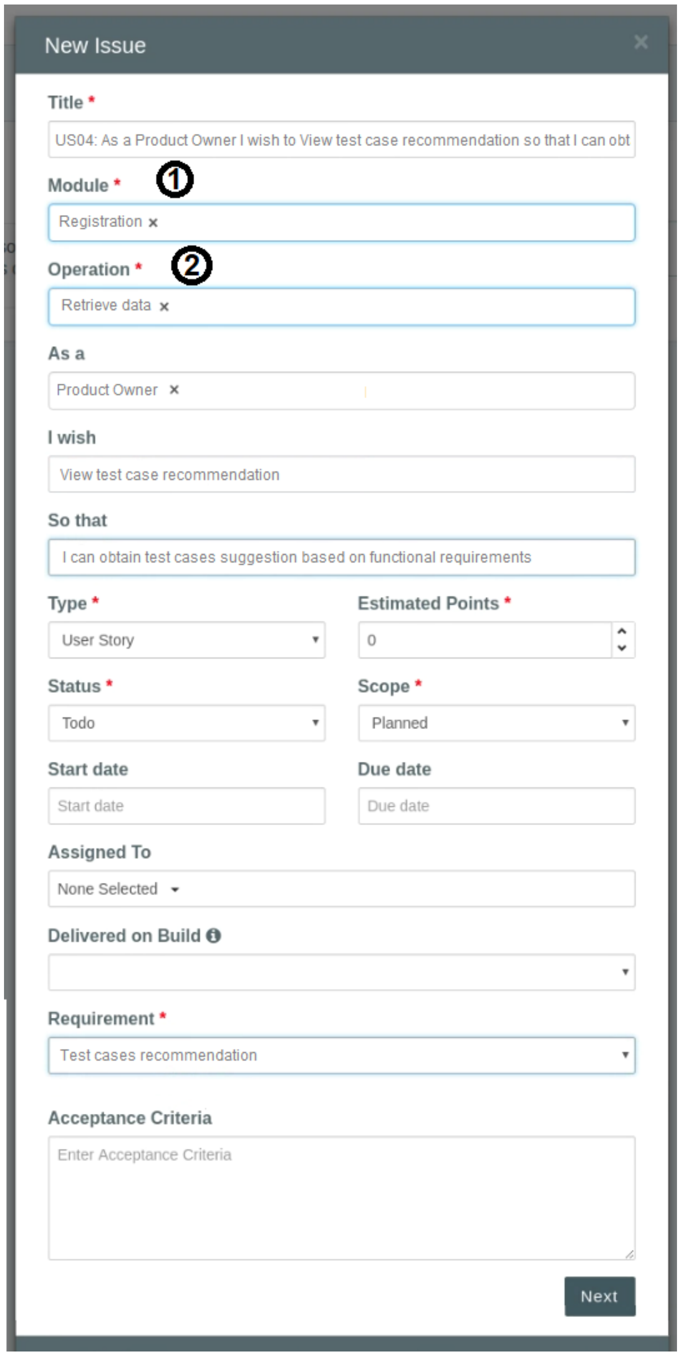
Fig. 1: Example of a structured project profile created in the NFRec tool

Planning Meetings, developers use the NFRec tool to create and store USs, which are classified by a category (module) and a subcategory (operation) that indicate the purpose of the FR specified by the US. In Figure 2, we present an example of a structured user story created in the NFRec tool. The US is classified with the module *Registration* (1) and the operation *Retrieve data* (2). We highlight that previously stored categories (i.e., modules and operations) must be reused by development teams to avoid data inconsistencies.

B. NFR Recommendation

For each US defined by the development team, the NFRec tool recommends a list of NFRs based on historical data

[1]Sprint Planning Meeting is an event of the Scrum in which the Scrum team plans the work to be performed in the Sprint.



New Issue

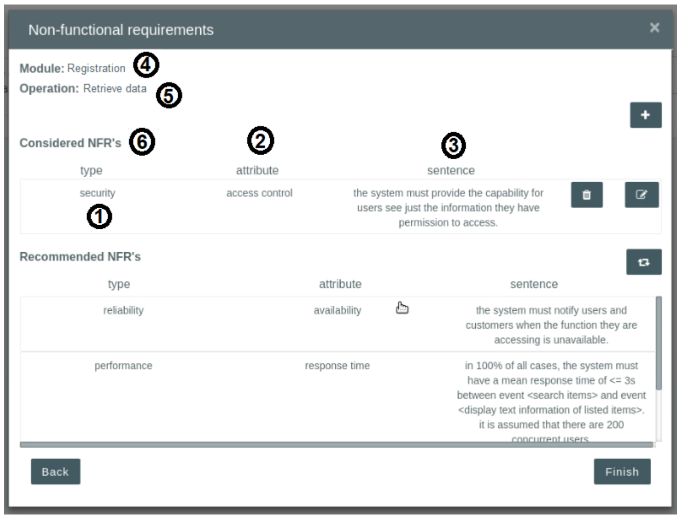
Title *	US04: As a Product Owner I wish to View test case recommendation so that I can obt
Module *	Registration ×
Operation *	Retrieve data ×
As a	Product Owner ×
I wish	View test case recommendation
So that	I can obtain test cases suggestion based on functional requirements
Type *	User Story
Estimated Points *	0
Status *	Todo
Scope *	Planned
Start date	Start date
Due date	Due date
Assigned To	None Selected
Delivered on Build	
Requirement *	Test cases recommendation
Acceptance Criteria	Enter Acceptance Criteria

Next

Fig. 2: Example of a structured user story created in the NFRec tool

analysis. NFRs are represented in the tool by a type, an attribute, and a sentence. The classification with type and attribute follows the indications presented by Mairiza et al. [9]. Additionally, NFR sentences are written following the indications presented by Eckhardt et al. [5], [6].

In Figure 3, we present an example of NFR recommendations presented in the NFRec tool, in which three NFRs are suggested for a US of the module *Registration* (4) and the operation *Retrieve data* (5). In the example, the developers have already accepted/considered one of the three recommendations



Non-functional requirements

Module: Registration
Operation: Retrieve data

Considered NFR's

type	attribute	sentence
security	access control	the system must provide the capability for users see just the information they have permission to access.

Recommended NFR's

type	attribute	sentence
reliability	availability	the system must notify users and customers when the function they are accessing is unavailable.
performance	response time	in 100% of all cases, the system must have a mean response time of <= 3s between event <search items> and event <display text information of listed items>. it is assumed that there are 200 concurrent users

Back   Finish

Fig. 3: Example of a NFR recommendation presented in the NFRec tool

(6), a NFR of type *security* (1), attribute *access control* (2), and sentence "*The system must provide the capability for users to see just the information they have permission to access.*" (3).

By using the tool, developers can visualize suggestions of NFRs for each US of current or future Sprints. Thus, NFRs can be considered early in the software development process, mitigating the risk of negligence with non-functional requirements resulted from agile practices.

IV. Empirical Evaluation

To evaluate the practical use of the NFRec tool, we conducted a case study in four ongoing Scrum-based projects from a Brazilian software company. We intend to: (i) assess the acceptance of the NFRec tool by agile software developers; (ii) evaluate the precision of generated recommendations.

A. *Case Study Design*

We performed an embedded case study [18] in which each project was considered a unit of analysis and each development team the subject of study. This step of the research lasted a month and each project executed two Sprints of 15 days during this period. Meanwhile, we collect information from different recommendation scenarios, and hence, we consider a sufficient period to answer the research questions.

The overall objective of the case study is to evaluate the cost-benefit of the NFRec tool from the perspective of development teams, regarding the support of non-functional requirements management. To accomplish that, we formulate the following research questions (RQs):

- **RQ$_1$**: is the NFRec tool useful to assist in eliciting non-functional requirements in Scrum-based software projects?
- **RQ$_2$**: is the cost to use the NFRec tool in Scrum-based software projects acceptable?
- **RQ$_3$**: what is the precision of the recommendations of the NFRec tool?

Therefore, we consider the cost-benefit of the NFRec tool worthwhile if the questions are positively answered.

As previously stated, we consider four software projects as study analysis units, referred as Project A, Project B, Project C and Project D. All of them consisting of Web information systems with the following scopes:

- **Project A**: development of a system with a cloud service and a Web client to enable independent or shared writing of poetry. It is composed of two developers;
- **Project B**: development of a Web client for generating graphics resources such as badges and business cards. It is composed of three developers;
- **Project C**: development of a system with a cloud service and a Web client to support the management of training projects through the management of students' activities and schedules, selections of participants to projects, etc. It is composed of five developers;
- **Project D**: development of a system with a cloud service and a Web client to assist the building and executing of Bayesian Networks. It is composed of five developers.

Prior to the case study, all projects performed requirements management during the Sprint Planning Meetings using a tool without NFR recommendation. Therefore, they did not perform any direct activity to define NFRs through the tool. They described them as acceptance criteria, DoD items, functional requirements, failures identified by test cases, etc.

To gather data for evaluating the acceptance of the NFRec tool, we apply a questionnaire based on the Technology Acceptance Model [4]. TAM aims to explain why individuals choose to adopt or not adopt a specific technology when accomplishing a task and it is based on two variables: Perceived Usefulness (PU) and Perceived Ease of Use (PEU). PU is related to the degree to which an individual believes that the use of a certain technology would increase his/her performance in the work. In contrast, PEU refers to the degree to which an individual believes that the use of a certain technology would be free of mental and physical effort.

Therefore, by adapting the TAM to the context of this work, we formulate 14 questions to evaluate the PU of the NFRec tool ($\mathbf{RQ}_1$) and 14 to assess its PEU ($\mathbf{RQ}_2$). For each question, we use a five-level Likert scale to collect participants' responses. The scale adopted the values: (1) Strongly Disagree, (2) Disagree, (3) Neither agree nor disagree, (4) Agree, and (5) Strongly Agree. In addition, we calculate the precision of the recommendations processed during the case study ($\mathbf{RQ}_3$).

Overall, we collected 4 answers for the questionnaire, i.e., one for each development team. On average, the teams of projects B and C have one year of experience in Scrum-based software projects. On the other hand, the teams of projects A and D had a mean of three years of experience in the same type of project.

B. Case Study Execution

To run the case study, we make the NFRec tool available to the teams of each project via a web link. The execution of the study comprised two stages: (i) training, and (ii) execution.

During the training phase, we presented concepts about the structuring of project information and the non-functional requirements recommendation, followed by a demonstration of the NFRec tool. The training lasted 1 hour. After this phase, the teams started using the NFRec tool at their projects' Sprint Planning Meetings, i.e., a real evaluation scenario in the industry.

First, they used the tool to create the profiles of their respective projects based on the assignment of tags referring to the five factors that affect the definition of NFRs (i.e., application domain, platform, project objective, software architecture, and technologies [15]). None of the teams had reported any troubles at this step.

Next, for each current Sprint, the teams used the NFRec tool to support the requirement management process during the Sprint Planning Meetings, in which they created structured USs. For each created US, the teams received NFRs recommendations and they could freely interact with the tool, accepting or rejecting suggestions. The duration of the meetings did not change in any of the projects, i.e., they continued within the planned interval of 1 hour. In the following, we present the activities carried out in the first moment of the case study.

In the first observed Sprint of Project A, the team selected one US, which received two NFR recommendations. The team accepted only one of them. Therefore, the RS achieved a precision rate of 50% for the corresponding Sprint from Project A.

For the first observed Sprint of Project B, the team selected two USs. For the first described US, the NFRec tool generated six NFR recommendations. The team accepted five of them. In contrast, for the second US, the tool generated two NFR recommendations, but the team accepted only one of them. Therefore, the RS achieved a precision rate of 75% for the first evaluated Sprint from Project B.

In the first observed Sprint of Project C, the team selected three USs and the NFRec tool generated three NFR recommendations for each one them. For two of the USs, the team accepted all the suggestions. On the other hand, for the remaining one, the team accepted only two of the recommendations. Therefore, the RS achieved a precision rate of 88.88% for the respective Sprint from Project C.

Finally, in the first observed Sprint of Project D, the team selected three USs. Two of them received three NFR recommendations, which were accepted by the team. In contrast, for the remaining one, the tool presented two suggestions of NFRs, of which only one was accepted by the team. Therefore, the RS achieved a precision rate of 87.5% in the first evaluated Sprint from Project D.

We highlight that the Product Owners of each project validated the recommendations accepted by the development teams. However, we did not evaluate whether the constraints specified by the recommended NFRs were considered in the development of the USs within the current Sprints since this issue is not part of the scope of this work. In the following, we present the activities carried out in the second moment of the case study.

In the second observed Sprint of Project A, the team selected two USs during the Sprint Planning Meeting. The NFRec tool presented one recommendation for one of the US and three for the other. The team accepted all the suggestions. Therefore, the RS achieved a precision rate of 100% in the second evaluated Sprint from Project A.

In the second observed Sprint of Project B, the team selected two USs during the Sprint Planning Meeting. The NFRec tool recommended one NFR for two of them. For the other one, the tool presented two recommendations. Overall, the team accepted two of the four suggestions. Therefore, the RS achieved a precision rate of 50% in the second first Sprint evaluated from Project B.

In the second observed Sprint of Project C, the team selected two USs. The NFRec tool recommended one NFR for the first described US and two for the later. The team accepted all the suggestions. Therefore, the RS achieved a precision rate of 100% for the corresponding Sprint.

Finally, for the second observed Sprint of project D, the development team selected three USs and the NFRec tool generated two NFR recommendations for each one of them. The team accepted all the suggestions for two of the USs but rejected one of the recommendations for the remaining one. Therefore, the RS achieved a precision rate of 83.33% for the corresponding Sprint from Project D.

At the end of the case study, all subjects filled in the questionnaire regarding their acceptance of the NFRec tool.

C. Results and Discussion

After the case study execution, we calculated the precision of the recommendations generated during the observed period. In Table I, we present the results of the overall precision and the precision for each project. Overall, the NFRec tool recommended 44 NFRs, of which 36 were accepted by the developers, resulting in a precision rate of 81.8% of the recommendations for the 20 considered USs. Therefore, we considered the results promising, concluding that $\mathbf{RQ}_3$ was positively answered. Among the four evaluated projects, we observed a higher precision rate in Project C (91.7 %), which is composed of professionals with previous experience in the use of the structured model of USs. On the other hand, we observed the lowest precision rate in Project B (66.7 %), which is the only one of the four projects that it is not developing a cloud service (*back-end*). Therefore, some recommendations were rejected for suggesting back-end assumptions/constraints such as validating the integrity of data sent to the server. Additionally, the team of Project B reported difficulties during the classification of the USs with modules and operations.

TABLE I: Results of precision calculated from the data obtained in the case study

Project	Num. **USs**	**Num. accep. NFRs**	**Num. rec. NFRs**	**Precis.**
Project A	3	5	6	83,3%
Project B	5	8	12	66,7%
Project C	5	11	12	91,7%
Project D	7	12	14	85,7%
Total	20	36	44	81,8%

As mentioned before, at the end of the case study, each development team answered a questionnaire to assess the acceptance of the solution regarding the perceived usefulness and perceived ease of use. To evaluate the responses, we summarized them as follows: Likert scale values represented by (1) and (2) were considered as indicative of disagreement (*Disagreement*); (3) as indicative of neutrality (*Neutral*); and (4) and (5) as indicative of agreement (*Agreement*).

In Tables II and III, we present the data collected for PU and PFU, respectively. Positive responses to variables are highlighted in bold. The maximum number of answers per question is four since we considered four subjects in the study. Therefore, as we formulate 14 questions for each TAM variable, the maximum number of responses per variable is 56. The final result for each analyzed variable is given by the sum of the answers per positive indicative (i.e., *Agreement*) divided by the maximum number of responses to the corresponding variable (i.e., 56).

By analyzing the data of the Table II, we verify that the research participants showed good acceptance for 13 of the 14 analyzed items of PU. Only items PU9 and PU12 did not receive mostly positive evaluations. For PU9, participants gave three neutral answers and one positive. In contrast, for PU12, they gave two positive responses and two neutral ones. Overall, the PU was positively assessed in 48 of 56 possible responses. The evaluation demonstrated an acceptance of the perceived usefulness of 85.7%. Therefore, it is possible to state that the research participants considered the tool useful ($\mathbf{RQ}_1$).

By analyzing the data of the Table III, we verify that the respondents showed good acceptance for 12 of the 14 analyzed items of PFU. Only the items PFU1, PFU11, PFU12, PFU13, and PFU14 did not receive mostly positive evaluations. For PFU1 and PFU13, the respondents gave two positives and two negative responses. In contrast, for PFU11, PFU12, and PFU14, they gave two positives and two neutral responses. Overall, the respondents positively assessed PFU in 41 of 56 answers, i.e., an acceptance of the perceived ease of use of 73.2%. Initially, some developers encountered difficulties in structuring USs and defining modules and operations. This fact may explain why PFU acceptance values were lower than those of PU. Even so, it is possible to state that the research participants considered the tool easy to use ($\mathbf{RQ}_2$).

TABLE II: Results for perceived usefulness from TAM

ID	**Item**	**Agreement**	**Neutral**	**Disagreement**
PU1	Job Difficult Without	**4**	0	0
PU2	Control Over Work	**4**	0	0
PU3	Job Performance	**4**	0	0
PU4	Addresses My Needs	**4**	0	0
PU5	Saves Me Time	**3**	1	0
PU6	Work More Quickly	**4**	0	0
PU7	Critical to My Job	**4**	0	0
PU8	Accomplish More Work	**3**	1	0
PU9	Cut Unproductive Time	**1**	3	0
PU10	Effectiveness	**4**	0	0
PU11	Quality of Work	**3**	1	0
PU12	Increase Productivity	**2**	2	0
PU13	Makes Job Easier	**4**	0	0
PU14	Useful	**4**	0	0

V. Threats to Validity

In this work, we consider the classification of validity threats proposed by Wohlin et al. in [18]. In what follows, we present the identified threats to validity.

Conclusion validity threats. We conducted the case study for two 15-day Sprints only, which represents a conclusion validity threats since we could get different answers with a longer period. However, to mitigate this threat, we evaluated four teams simultaneously, which returned similar results. In

TABLE III: Results for perceived ease of use from TAM

ID	Item	Agreement	Neutral	Disagreement
PFU1	Confusing	2	0	**2**
PFU2	Error Prone	0	1	**3**
PFU3	Frustrating	0	0	**4**
PFU4	Dependence on Manual	0	1	**3**
PFU5	Mental Effort	0	0	**4**
PFU6	Error Recovery	**4**	0	0
PFU7	Rigid and Inflexible	1	1	**2**
PFU8	Controllable	**4**	0	0
PFU9	Unexpected Behavior	0	0	**4**
PFU10	Cumbersome	0	1	**3**
PFU11	Understandable	**2**	2	0
PFU12	Ease of Remembering	**2**	2	0
PFU13	Provides Guidance	**2**	0	2
PFU14	Easy to Use	**2**	2	0

addition, we collected data regarding the different types of USs, i.e., we could observe different recommendation scenarios during the case study.

Internal validity threats. On average, developers from two of the four projects had just one year of experience. This fact can lead to a threat to internal validity, since they may not be mature enough to properly answer the questionnaire. To mitigate this threat, developers of each project answered the questionnaire together, and hence, they had the opportunity to discuss their responses with each other, aggregating their knowledge. Another threat to internal validity refers to the construction of the questionnaire. However, to mitigate this problem, we generated it based on the TAM, which is a validated and extensively used model in the literature to evaluate new technologies.

Construct validity threats. Factors related to the type of project (e.g, scope, complexity, familiarity with technology, team experience, etc.) might affect the acceptability of the tool, which can lead to a threat to construct validity. To mitigate this problem, we tried to select projects with different scopes and domains. However, we reinforce that further research is needed to investigate this threat to validity.

External validity threats. The case study was carried out with four projects from the same company and, consequently, it is not possible to generalize the obtained results to other companies that use Scrum. However, as future work, we intend to continue the empirical evaluation of the NFRec tool in more projects from different companies.

VI. Conclusion and Future Work

In this paper, we presented the NFRec tool for supporting agile non-functional requirement elicitation. We reported the results of a case study to evaluate the acceptance of the tool from the point of view of development teams of four software projects from a Brazilian software company.

The empirical evaluation indicated that the NFRec tool seems to be able to accurately recommend NFRs, responding positively to the research question $\mathbf{RQ}_3$. Furthermore, the results of the case study obtained through the questionnaire showed that most of the subjects considered the NFRec tool useful and easy to use for supporting the elicitation of non-functional requirements, responding positively to the research questions $\mathbf{RQ}_1$ and $\mathbf{RQ}_2$.

For future work, we intend to replicate this study with a greater number of subjects, analyzing different projects from distinct software companies to mitigate the validity threats.

Acknowledgment

The authors would like to thank CAPES for supporting this work.

References

[1] B. M. Aljallabi and A. Mansour. Enhancement approach for non-functional requirements analysis in agile environment. In *2015 International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE)*, pages 428–433, Sept 2015.

[2] V. Bajpai and R. P. Gorthi. On non-functional requirements: A survey. In *Electrical, Electronics and Computer Science (SCEECS), 2012 IEEE Students' Conference on*, pages 1–4, March 2012.

[3] M. Bourimi, T. Barth, J. M. Haake, B. Ueberschär, and D. Kesdogan. AFFINE for enforcing earlier consideration of NFRs and human factors when building socio-technical systems following agile methodologies. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6409 LNCS:182–189, 2010.

[4] F. D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, pages 319–340, 1989.

[5] J. Eckhardt, A. Vogelsang, and H. Femmer. An approach for creating sentence patterns for quality requirements. In *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, pages 308–315, Sep. 2016.

[6] J. Eckhardt, A. Vogelsang, H. Femmer, and P. Mager. Challenging incompleteness of performance requirements by sentence patterns. In *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pages 46–55, Sep. 2016.

[7] W. M. Farid. The normap methodology: Lightweight engineering of non-functional requirements for agile processes. In *Proceedings of the 2012 19th Asia-Pacific Software Engineering Conference - Volume 01*, APSEC '12, pages 322–325, Washington, DC, USA, 2012. IEEE Computer Society.

[8] R. Hoda, N. Salleh, J. Grundy, and H. M. Tee. Systematic literature reviews in agile software development: A tertiary study. *Information and Software Technology*, 85:60 – 70, 2017.

[9] D. Mairiza, D. Zowghi, and N. Nurmuliani. An investigation into the notion of non-functional requirements. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 311–317, New York, NY, USA, 2010. ACM.

[10] R. R. Maiti, A. Krasnov, and D. M. Wilborne. Agile software engineering & the future of non-functional requirements. *Journal of Software Engineering Practice*, 2(1):1–8, december 2018.

[11] R. R. Maiti, A. Krasnov, and M. Wilborne. Predicting nfrs in agile software engineering. In *Proceedings of the 19th Annual SIG Conference on Information Technology Education*, SIGITE '18, pages 161–161, New York, NY, USA, october 2018. ACM.

[12] R. R. Maiti and F. J. Mitropoulos. Capturing, eliciting, predicting and prioritizing (cepp) non-functional requirements metadata during the early stages of agile software development. In *SoutheastCon 2015*, pages 1–8, April 2015.

[13] R. R. Maiti and F. J. Mitropoulos. Capturing, eliciting, and prioritizing (cep) nfrs in agile software engineering. In *SoutheastCon 2017*, pages 1–7, March 2017.

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

[15] F. Ramos, A. Costa, M. Perkusich, H. Almeida, and A. Perkusich. A non-functional requirements recommendation system for scrum-based projects. *In The 30th International Conference on Software Engineering and Knowledge Engineering*, 2018.

[16] S. Stavru. A critical examination of recent industrial surveys on agile method usage. *Journal of Systems and Software*, 94:87 – 97, 2014.

[17] VersionOne. *12th Annual State of Agile Survey*, 2018. Acessado em: 19 de dezembro de 2018. https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report.

[18] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012.