

Managing Non-functional Requirements in Agile Software Development

Ezeldin Sherif(✉), Waleed Helmy, and Galal Hassan Galal-Edeen

Information Systems Department, Faculty of Computers and AI, Cairo University, Cairo, Egypt
ezeldinsherif@gmail.com, {w.helmy,galal}@fci-cu.edu.eg

Abstract. Agile software development is an iterative software development methodology that aims at maximizing productivity, effectiveness, and speed of delivery. There are a lot of benefits of Agile Software Development. However, there are still some challenges. For example, non-functional requirements are not treated as first-class artifacts during the development lifecycle, which causes many problems such as customer dissatisfaction and much rework which therefore affects time and cost. This paper explains different solutions that have handled non-functional requirements issues in Agile. The paper shows the strength and weakness of each solution, however, there is no single solution that handles all main activities of requirements engineering such as elicitation, analysis, validation and management in regards of non-functional requirements.

Keywords: Agile · Non-functional requirements · Scrum

1 Introduction

The term Agile implies agility, adaptability, and flexibility [1]. Agile software development entails breaking down a software project into smaller pieces. First, the development team works on the most important functions that are selected by the users. The team uses an iterative incremental development method for all selected functions and accordingly, makes each iteration's results into a running system. Agile software development (ASD) has become increasingly popular in the past decade due to its benefits such as an increase in team productivity, motivation and software quality. Due to changing market demands and the rapid speed of technological change, Agile is becoming an increasingly important [2]. Agile focuses more on working software, however less on non-functional requirements [3]. This causes increased cost or longer time-to-market in case of not properly dealing with non-functional requirements [4]. Therefore, this is our focus; as non-functional requirements must be treated as important as functional requirements in order to produce working quality software. The paper presents different solutions that manage non-functional requirements in ASD. This is how

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
O. Gervasi et al. (Eds.): ICCSA 2022, LNCS 13376, pp. 205–216, 2022.
https://doi.org/10.1007/978-3-031-10450-3_16

the paper is structured: Sect. 2 presents requirements engineering challenges in Agile, Sect. 3 presents solutions to handle those challenges and Sect. 4 provides the conclusion.

2 Requirements Engineering Challenges in Agile

There are many challenges in Agile requirements engineering, as shown in Table 1. The table shows 15 different research papers along with 16 challenges. The challenge labeled with "Yes" means that this challenge is mentioned in the given paper. The challenge that is labeled with "-" means that this challenge is not mentioned in the given paper. As noticed from Table 1, negligence of non-functional requirements and minimal documentation are two significant challenges in requirements engineering of Agile software development. Problems in neglecting non-functional requirements include too-minimal documentation of non-functional requirements, insufficient techniques to deal with non-functional requirements in Agile and focus on functionality [3,4]. According to Cao and Ramesh's empirical investigation, [6,7], Non-functional requirements (NFRs) are given less emphasis in the early stages of Agile since the client is more concerned with core operations. Non-functional requirements are frequently overlooked in Agile testing for a variety of reasons, including lack of awareness, experience, priority, culture, and time and expense constraints [20]. There are many different solutions that handled the negligence of non-functional requirements in Agile. Each solution handled the negligence in different perspective and each solution is described in section three.

Table 1. Challenges in agile requirements engineering

No	Challenge	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[19]
1	**Neglect of NFR**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**-**	**Yes**	**Yes**	**Yes**
2	Minimal documentation	Yes	Yes	-	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
3	Inadequate/inappropriate architecture	Yes	-	Yes	-	Yes	-	-	-	Yes	Yes	Yes	Yes	Yes	Yes	Yes
4	Prioritization of NFR	Yes	-	-	-	-	-	-	Yes	-	-	Yes	-	-	Yes	Yes
5	NFR infeasibility	-	-	Yes	-	-	Yes	-	-	-	-	-	-	-	-	-
6	Teams interaction	-	-	Yes	-	-	-	-	-	-	-	-	Yes	-	-	-
7	Inadequate NFR verification	-	-	Yes	Yes	-	-	-	-	-	-	-	Yes	-	Yes	-
8	NFRs identification	-	-	Yes	-	-	Yes	-	-	-	-	Yes	-	-	-	-
9	Customer unavailability	-	-	-	Yes	Yes	-	-	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
10	Project budget and time estimation	-	-	-	-	Yes	Yes	-	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
11	Change of requirements	-	-	-	-	Yes	Yes	-	-	Yes	Yes	Yes	-	-	-	Yes
12	Problems related to prototyping	-	-	-	-	-	-	Yes	-	-	-	-	-	-	-	-
13	Not understanding the big picture	-	-	-	-	-	-	Yes	-	-	-	-	-	-	-	-
14	Lack of requirements traceability	-	-	-	-	-	-	-	Yes	-	-	-	-	-	-	-
15	Contractual limitations	-	-	-	-	-	-	-	Yes	Yes	Yes	-	-	-	-	Yes
16	Customer inability and agreement	-	-	-	-	-	-	-	-	Yes	Yes	Yes	-	-	-	-

3 Solutions to Handle the Challenges

This section presents related solutions that can handle non-functional requirements in Agile software development. As shown in Table 2, it consists of 7 different methods and methodologies to handle the non-functional issue(s) in Agile software development. As shown in Table 3, it consists of 10 different frameworks, tools, processes, and guidelines to handle the non-functional issue(s) in Agile software development. In each table, it shows the strength of each of the proposed solutions. The keyword "yes" means that the item does apply to that specific solution/research paper, and the keyword "no" means that the item doesn't apply to that specific solution/research paper. In general, requirements engineering phase consists of four main activities: requirements elicitation, analysis, validation and management. As showing in Tables 2 and 3, each solution tackles different perspectives of those activities, but there is no single solution that handles the four main activities of requirements engineering. There is a need for a solution that can handle those activities of requirements engineering in Agile along with functional requirements but without affecting the principles and values of Agile.

3.1 Methods/Methodologies to Handle Non-functional Requirements in Agile

This section explains 7 different methods and methodologies that handle non-functional requirements in Agile software development as it can be seen in Table 2.

Table 2. Different methods/methodologies to handle non-functional requirements in agile

Item/Paper	1	2	3	4	5	6	7
	NORMAP [21]	NERV [25]	MEDoV [27]	CEP [28]	Decision Tree [30]	ACRUM [33]	SENSoR [34]
Type of solution	Methodology	Methodology	Method	Methodology	Methodology	Method	Method
Inform client meaning of NFR	No	No	No	No	No	No	Yes
Handles functional requirements	Yes	Yes	No	Yes	No	Yes	No
Elicit NFR	No	Yes	Yes	Yes	No	No	Yes
Analyze NFR	No	Yes	Yes	No	No	Yes	Yes
Link NFR with functional requirements	Yes	Yes	No	No	No	Yes	No
Model NFR	Yes	No	Yes	No	No	No	No
Recommend NFR	No	No	No	No	No	No	No
Plan (prioritized) NFR	No	No	No	Yes	No	No	No
Schedule NFR	No	No	No	No	No	No	No
Predict NFR	No	No	No	No	Yes	No	No
Document NFR	Yes	Yes	Yes	Yes	No	Yes	Yes
Validate NFR	No	No	Yes	No	No	Yes	Yes

NORMAP. NORMAP is a simple framework for identifying, linking, and modeling non-functional requirements [21]. To support the goals, three areas

of improvement were proposed. The first enhancement is the W^8 user story card model. It is basically an enhancement of the Agile index card (user story) technique in order to gather requirements in ASD. Each "W" in W^8 represents a question to be answered. It includes eight questions to capture functional and non-functional requirements. The first "W" is "who" is the actor, second "W" is "what" the user wants, the third "W" is "why" that captures business justification, fourth "W" is "within" estimated time frame, fifth "W" is "while it is nice to have" or in other words optional non-functional requirements, sixth "W" is "without ignoring" meaning required non-functional requirements, seventh "W" is "with a priority of" and last "W" is "which may impact" that captures a list of impacted requirements. The second enhancement is that NORMAP proposes an Agile requirements taxonomy, which divides requirements into functional (Agile Use Case) and non-functional (Agile Loose Case) categories. The third enhancement to link functional and non-functional requirements, "Before, after, override, and wrap" are examples of Aspect-Oriented "pointcut" operators used by NORMAP. The term "pointcut" refers to a point in time where an ACC is performed and linked to an AUC in order to avoid code tangling and crosscutting issues. In other words, it combines functional and non-functional requirements and links ACC to AUC. The NORMAP framework can be utilized semi-automatically or manually. NORMANUAL is the name of the manual framework, and NORMATIC is the name of the semi-automated framework. The proposed framework contains many steps, which increases the complexity of the framework. Therefore, it may contradict the principles and values of ASD. In addition, the framework was not validated in real-world Agile projects; however, it is validated using data-sets. The first dataset is a collection of unrelated requirements from several projects in a DePaul University software engineering course. The second dataset represents the European Union eProcurement Online System's requirements model. Also, the framework only focuses on identifying, modeling and linking non-functional requirements with functional requirements in Agile.

NERV. NERV is lightweight methodology to address NFRs in the early stages of Agile [25]. NERV handles NFRS elicitation, reasoning and validation in ASD. It is an improvement of NORMAP methodology. NERV uses Agile user story cards to capture functional requirements (including users, actions performed and reason for doing action), an NFRusCOM card for handling quality requirements information and capturing quality requirements' aspects, an NFR trigger card for eliciting quality requirements, and a non-functional requirements reasoning taxonomy for classifying quality requirements. It also employs NFR's quantification taxonomy to quantify the quality requirements' validation criteria, as well as the NERV Agility Index (NAI) to measure the degree of agility of each non-functional and functional need. It is validated using two case studies (datasets) and not real-world projects. Furthermore, NERV added six additional artifacts (NFR Elicitation Taxonomy, NFRusCOM Card, NFR Reasoning Taxonomy, NFR Quantification Taxonomy, NFR Trigger Card, and NAI), which

may lead to Agile becoming a heavily documented methodology, which is contrary to the Agile Manifesto's objectives.

MEDoV. MEDoV consists of three phases. Phase one is setting project objectives with business goals by a deep understanding of customer values and expectations [27]. Phase one is carried out only once at the start of the project, however phases II and III are carried out again in each iteration. Eliciting and documenting requirements is part of phase two. Phase three performs requirements analysis, specification and validation. Much formal documentation is needed in order to specify requirements. There is the possibility of wasted efforts in understanding the KPI for each requirement.

CEP. CEP methodology is an extension of NERV and NORMAP [28]. The objective of this methodology is to develop an automated framework that captures, elicits and prioritizes non-functional requirements from requirements documents that contain images. The capture part was done using OCR, converting images into readable textual format. The elicitation part uses NFRL+, which elicits the non-functional requirements from digitized documents and images. The prioritization part is done by prioritizing the non-functional requirements by using weights. For example, the non-functional requirement of the weight of one has a higher priority than a non-functional requirement of five. The methodology is simple enough, but there are few details to explain how obtaining a solution is approached. For example, what if the document is written poorly or the requirements are not complete. In addition, the methodology was not validated in real-world Agile projects; however, it was validated using data-sets.

Decision Tree. This research uses the prediction model to extend the Capture Elicit Prioritize (CEP) methodology by using a simple decision tree to make predictions using historical non-functional requirements data [30]. By applying the decision tree, non-functional requirements that appeared three times or more are the ones that were predicted in the future iterations. This research only focuses on the prediction by using simple decision trees. The scope of the study is focused on prediction only. The proposed solution wasn't applied using real-life projects.

ACRUM. It's an Agile development method based on quality attributes [33]. There are three major activities in ACRUM along with Scrum activities: first activity is the AQUA (analysis of quality attributes), the second activity is the preparation of the correlation mapping table (Requirements Association Matrix) between requirements and its analysis, the third activity is failure or success of the quality attributes is verified through the Validation of Quality Attribute (VQA). It was validated using only two case studies.

SENoR. It is a method for eliciting NFRs, consisting of three steps: preparation, workshop and follow-up. The first step is preparation, which is the business case presentation [34]. According to ISO25010, the second step is a workshop, which consists of a series of short brainstorming sessions. Each session dedicated to one of the ISO25010's qualities sub-characteristic. The purpose of the second step is to elicit NFRs. The third step is follow-up which consists of analyzing non-functional requirements and introducing improvements, such as identifying contradictory or duplicated information. The output of this step is non-functional requirements and lessons learned. There are four roles in SENoR: moderator, presenter, recorder and experts. The Moderator is the person who leads the whole process. The presenter is the individual who explains the project's concept to the participants, the Recorder is the person who records the work of participants, and the Expert is the person who is an expert on specific content and defining its requirements. It only focuses on elicitation of the non-functional requirements.

3.2 Other Solutions to Handle Non-functional Requirements in Agile

This section explains 10 other solutions (frameworks, processes, tools and guidelines) that handle non-functional requirements Agile software development as it can be seen in Table 3.

Table 3. Other solutions to handle non-functional requirements in agile

Item/Paper	1	2	3	4	5	6	7	8	9	10
	NORMATIC [22]	NORPLAN [23]	NORVIEW [24]	CEPP [26]	NFRec Tool [29]	NFR recommendation System [31]	AgileRE [32]	Elicitation Guideline [35]	Framework for requirements elicitation [36]	Modeling NFR [37]
Type of solution	Tool	Metric	Framework	Study	Tool	Process	Tool	Guidelines	Framework	New artifacts
Inform client meaning of NFR	No	No	No	No	No	No	No	No	No	No
Handles functional requirements	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Elicit NFR	No	No	No	Yes	Yes	No	No	Yes	Yes	No
Analyze NFR	No	No	No	No	No	No	No	No	No	No
Link NFR with functional requirements	No	No	No	No	Yes	No	Yes	Yes	No	Yes
Model NFR	Yes	No	No	No	No	No	No	No	No	Yes
Recommend NFR	No	No	No	No	Yes	Yes	No	No	No	No
Plan (prioritized) NFR	No	Yes	No	Yes	No	No	No	No	No	No
Schedule NFR	No	No	Yes	No	No	No	No	No	No	No
Predict NFR	No	No	No	Yes	No	No	No	No	No	No
Document NFR	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No
Validate NFR	No	No	No	No	No	Yes	Yes	No	No	No

NORMATIC. NORMATIC is a Java-based visual modeling simulation tool that is part of the NORMAP framework [22]. It supports the three building blocks of the NORMAP framework, which includes identifying, linking and modeling non-functional requirements with functional requirements. The tool is designed to model functional and non-functional requirements in a visual environment that may be utilized in Agile methods like Scrum. The three NORMAP building components were used to create NORMATIC: Agile Use Case (AUC), Agile Loose Case (ALC), and Agile Choose Case (ACC). AUC includes requirement quality attributes; ALC identifies to address non-functional as a story, and ACC represents instances of a potential solution for identified ALC.

The proposed tool was validated using datasets, specifically two case studies only. The European Union Electronic Procurement System is the first dataset, and the PROMISE dataset is the second. So it wasn't used in real-world Agile projects to measure the tool's effectiveness, and the number of case studies is minimal. In addition, since this tool is part of the NORMAP framework, it has the same complexity as the framework, which may affect the concept of Agile software development by containing too many complicated steps. Also, NORMATIC is not web-enabled, and therefore it doesn't support the mobile computing environment, and today's trend is shifting toward web applications and mobile applications.

NORPLAN. NORPLAN is one of the components of the NORMAP framework [23]. NORPLAN's key components for calculating the plan include requirements quality metrics, risk-driven algorithms and project management metrics. NORPLAN's goal is to generate a potentially better requirements implementation sequence. This implementation sequence calculates technical and project management risks of all requirements (non-functional and functional), which will aid in planning which requirements will be implemented in future sprints/releases. In addition, riskiest-requirements-first and riskiest-requirements-last are the two priority techniques used by NORPLAN, instead of the normal business value prioritization scheme used in Agile software development methodologies.

First, NORPLAN was not implemented in real-world Agile software projects. Second, a high-business-value-first prioritization scheme should be considered as a first class artifact as it may cause conflict with client/customer business needs and therefore cause unnecessary issues such as losing a segment of the market. In other words, it disregards the Product Owner's requested priority of requirements. Third, since NORPLAN is part of NORMAP, it has a similar complexity with NORMAP which also contradicts the concept of Agile.

NORVIEW. NORVIEW is the last component of NORMAP framework [24]. The next step is to schedule and visualize the plan using different project management metrics after it has been developed in NORPLAN. This will aid Scrum teams in planning, scheduling and visualizing NFRs. NORVIEW is a tree-like direct visual representation of NORPLAN. NORVIEW is the last component of NORMAP and its purpose is to support planning and scheduling. This component could be a part of NORPLAN in order to reduce the number of components and therefore reduce complexity. In addition, same as for other components of NORMAP, it is not validated in the real-world Agile projects.

CEPP. This methodology gathers non-functional requirements metadata from documents and images [26]. It uses historical trending to predict non-functional requirements that could be overlooked in the early stages of ASD. Also, it prioritizes non-functional requirements using α β γ- framework where process α is used to subjectively prioritize needs in order to subjectively minimize the number of

options. The objective is to provide effective techniques to predict and prioritize non-functional requirements. This methodology wasn't validated by either case studies or data sets.

NFRec Tool. The tool aids in the elicitation of non-functional needs by performing two tasks: project information structuring and non-functional requirement recommendation [29]. A developer can visualize suggestions of non-functional requirements for each user story in the current or future sprints, and therefore non-functional requirements will be considered early in the development. The tool should be analyzed and validated using different projects from distinct software development organizations.

NFR Recommendation System. It is a non-functional requirements recommendation system based on collaborative filtering to support Scrum practitioners [31]. It introduced one new role, Scrum Architect, and a new artifact called "semi-structured user story". This may affect the concept of agility and the Agile manifesto by getting close to traditional software development methodology. The dataset was insufficient to test recommendation systems in various domains.

AgileRe Tool. It is a tool that provides basic requirements management features such as persona creation, epic and user story management, role-based access for product owners, customer feedback, dashboard and reports [32]. This tool is accessible through the web browser interface. The tool covers many different features in regard to requirements; however, there is less focus on NFRs. In the tool, NFRs are included in the acceptance criteria.

Elicitation Guide. It contains elicitation guidelines for non-functional requirements in Agile methods, and it helps developers and users [35]. This guideline helps in two perspectives: the first is that developers interact with the customer or user for requirements elicitation. The second is that the team has to extract or elicit non-functional requirements from the requirements. More case studies are needed, especially in the software industry, and preferably a tool should be developed to help with the objectives of this solution.

Framework for Requirements Elicitation. Pre-elicitation, mid-elicitation, and post-elicitation are the three phases of the elicitation framework [36]. It combines mind mapping, Scrum, and the Joint Requirement Document (JRD). There is a lack of focus on non-functional requirements. The main parts of the proposed framework are:

- Phase I (pre-elicitation) is JRD to create an initial scope document.
- Phase II (mid-elicitation) is mind mapping that focuses on detailed requirements gathering.
- Phase III (post-elicitation): The requirements are sent to the Scrum process's product backlog as user stories.

Modeling NFR. It presents three building blocks for identifying, linking and modeling non-functional requirements with the functional requirement in order to improve software quality [37]. The first block is where functional requirements are modeled through AUC (Agile Use Cases). Second, non-functional requirements are modeled using ALC (Agile Loose Cases). The third is non-functional requirements; Potential solutions are modeling through ACC (Agile Choose Cases). Three new artifacts were introduced, which may compromise the principles of Agile. In addition, it was validated using datasets, and not real-life projects.

4 Conclusion

Agile software development is a one of the hot topics in the software engineering. It comes with its own challenges such as challenges in requirements engineering in Agile software development. There are many challenges within requirements engineering of Agile such as ignoring non-functional requirements and minimal documentation. In the past decade, there were many solutions that handled non-functional requirements in Agile in different ways. Each of those solutions had its own weakness and strength. However, there is no single solution that covers non-functional requirements elicitation, analysis, documentation, validation, recommendation, along with functional requirements. This can be a potential research point to reduce the issues of ignoring non-functional requirements in Agile software development.

References

1. Jarzębowicz, A., Weichbroth, P.: A systematic literature review on implementing non-functional requirements in agile software development: issues and facilitating practices. In: Przybyłek, A., Miler, J., Poth, A., Riel, A. (eds.) LASD 2021. LNBIP, vol. 408, pp. 91–110. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-67084-9_6
2. Lu, H.-K., Lin, P.-C., Huang, P.-C., Yuan, A.: Deployment and evaluation of a continues integration process in agile development. J. Adv. Inf. Technol. **8**(4), 204–209 (2017). https://doi.org/10.12720/jait.8.4.204-209
3. Jarzebowicz, A., Weichbroth, P.: A qualitative study on non-functional requirements in agile software development. IEEE Access **9**, 40458–40475 (2021). https://doi.org/10.1109/ACCESS.2021.3064424
4. Behutiye, W., et al.: Management of quality requirements in agile and rapid software development: a systematic mapping study. Inf. Softw. Technol. **123**, 106225 (2020). https://doi.org/10.1016/j.infsof.2019.106225
5. Schön, E.-M., Winter, D., Escalona, M.J., Thomaschewski, J.: Key challenges in agile requirements engineering. In: Baumeister, H., Lichter, H., Riebisch, M. (eds.) XP 2017. LNBIP, vol. 283, pp. 37–51. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57633-6_3
6. Behutiye, W., Karhapää, P., Costal, D., Oivo, M., Franch, X.: Non-functional requirements documentation in agile software development: Challenges and solution proposal. arXiv, pp. 515–522 (2017). https://doi.org/10.1007/978-3-319-69926-4

7. Alsaqaf, W., Daneva, M., Wieringa, R.: Agile quality requirements engineering challenges: first results from a case study. Int. Symp. Empir. Softw. Eng. Meas. **2017**, 454–459 (2017). https://doi.org/10.1109/ESEM.2017.61
8. Gaikwad, V., Joeg, P.: A case study in requirements engineering in context of agile. Int. J. Appl. Eng. Res. **12**(8), 1697–1702 (2017)
9. Elghariani, K., Kama, N.: Review on agile requirements engineering challenges. In: 2016 3rd International Conference on Computer Information Science ICCOINS 2016 - Proceedings, pp. 507–512 (2016). https://doi.org/10.1109/ICCOINS.2016.7783267
10. Alam, S., Nazir, S., Asim, S., Amr, D.: Impact and challenges of requirement engineering in agile methodologies: a systematic review. Int. J. Adv. Comput. Sci. Appl. **8**(4), 411–420 (2017). https://doi.org/10.14569/ijacsa.2017.080455
11. Käpyaho, M., Kauppinen, M.: Agile requirements engineering with prototyping: a case study. In: 2015 IEEE International Requirements Engineering Conference RE 2015 - Proceedings, pp. 334–343 (2015). https://doi.org/10.1109/RE.2015.7320450
12. Inayat, I., Moraes, L., Daneva, M., Salim, S.S.: A reflection on agile requirements engineering: solutions brought and challenges posed. In: ACM International Conference Proceeding Series, 29 May 2015, vol. 25 (2015). https://doi.org/10.1145/2764979.2764985
13. Inayat, I., Salim, S.S., Marczak, S., Daneva, M., Shamshirband, S.: A systematic literature review on agile requirements engineering practices and challenges. Comput. Human Behav. **51**, 915–929 (2015). https://doi.org/10.1016/j.chb.2014.10.046
14. Sunner, D., Bajaj, H.: Classification of functional and non-functional requirements in agile by cluster neuro-genetic approach. Int. J. Softw. Eng. Its Appl. **10**(10), 129–138 (2016). https://doi.org/10.14257/ijseia.2016.10.10.13
15. Saleh, M., Baharom, F., Farvin, S., Mohamed, P., Ahmad, M.: A systematic literature review of challenges and critical success factors in agile requirement engineering. In: International Conference on Information and Knowledge Management, pp. 248–254, July 2018
16. Batra, M., Bhatnagar, A: A research study on critical challenges in agile requirements engineering. Int. Res. J. Eng. Technol. **6**, 1214–1219 (2019)
17. Telesko, R.: Road to agile requirements engineering: Lessons learned from a web app project. Stud. Syst. Decis. Control **141**, 65–78 (2018). https://doi.org/10.1007/978-3-319-74322-6_5
18. Ramesh, B., Cao, L., Baskerville, R.: Agile requirements engineering practices and challenges: an empirical study. Inf. Syst. J. **20**(5), 449–480 (2010). https://doi.org/10.1111/j.1365-2575.2007.00259.x
19. Saleh, M., Baharom, F., Mohamed, S.F.P.: Critical success factors and challenges in agile requirements engineering. Turkish J. Comput. Math. Educ. **12**(3), 1670–1682 (2021). https://doi.org/10.17762/turcomat.v12i3.989
20. Cruzes, D.S., Felderer, M., Oyetoyan, T.D., Gander, M., Pekaric, I.: How is security testing done in agile teams? A cross-case analysis of four software teams. Lect. Notes Informatics (LNI). In: Proceedings - Series of the Gesellschaft fur Informatik, vol. 292, pp. 133–134 (2019). https://doi.org/10.18420/se2019-40
21. Farid, W.M.: The Normap methodology: lightweight engineering of non-functional requirements for agile processes. In: Proceedings of the Asia-Pacific Conference on Software Engineering, APSEC, vol. 1, pp. 322–325 (2012). https://doi.org/10.1109/APSEC.2012.23

22. Farid, W.M., Mitropoulos, F.J.: NORMATIC: a visual tool for modeling non-functional requirements in agile processes. In: Conference Proceedings on IEEE SOUTHEASTCON, no. 978 (2012). https://doi.org/10.1109/SECon.2012.6196989
23. Farid, W.M., Mitropoulos, F.J.: NORPLAN: non-functional requirements planning for agile processes. In: Conference Proceedings on IEEE SOUTHEASTCON (2013). https://doi.org/10.1109/SECON.2013.6567463
24. Farid, W.M., Mitropoulos, F.J.: Visualization and scheduling of non-functional requirements for agile processes. In: Conference Proceedings on IEEE SOUTHEASTCON (2013). https://doi.org/10.1109/SECON.2013.6567413
25. Domah, D., Mitropoulos, F.J.: The NERV methodology: a lightweight process for addressing non-functional requirements in agile software development. In: Conference Proceedings on IEEE SOUTHEASTCON, vol. 2015-June (2015). https://doi.org/10.1109/SECON.2015.7133028
26. Maiti, R.R., Mitropoulos, F.J.: Capturing, eliciting, predicting and prioritizing (CEPP) non-functional requirements metadata during the early stages of agile software development. In: Conference Proceedings on IEEE SOUTHEASTCON, vol. 2015-June (2015). https://doi.org/10.1109/SECON.2015.7133007
27. Dragicevic, S., Celar, S., Novak, L.: Use of method for elicitation, documentation, and validation of software user requirements (MEDoV) in agile software development projects. In: Proceedings of the 6th International Conference on Computational Intelligence, Communication Systems and Networks, CICSyN 2014, pp. 65–70 (2014). https://doi.org/10.1109/CICSyN.2014.27
28. Maiti, R.R., Mitropoulos, F.J.: Capturing, eliciting, and prioritizing (CEP) NFRs in agile software engineering. In: Conference Proceedings on IEEE SOUTHEASTCON (2017). https://doi.org/10.1109/SECON.2017.7925365
29. Ramos, F., et al.: Evaluating software developers' acceptance of a tool for supporting agile non-functional requirement elicitation. In: Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE, vol. 2019-July, August, pp. 26–31 (2019). https://doi.org/10.18293/SEKE2019-107
30. Maiti, R.R., Krasnov, A., Wilborne, D.M.: Agile software engineering & the future of non-functional requirements. J. Softw. Eng. Pract. **2**(1), 1–8 (2018)
31. Ramos, F., Costa, A., Perkusich, M., Almeida, H., Perkusich, A.: A non-functional requirements recommendation system for scrum-based projects. In: Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE, vol. 2018-July, pp. 149–154, July 2018. https://doi.org/10.18293/SEKE2018-107
32. Gaikwad, V., Joeg, P., Joshi, S.: AgileRE: agile requirements management tool. Adv. Intell. Syst. Comput. **661**, 236–249 (2018). https://doi.org/10.1007/978-3-319-67618-0_22
33. Jeon, S., Han, M., Lee, E., Lee, K.: Quality attribute driven agile development. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, SERA 2011, pp. 203–210 (2011). https://doi.org/10.1109/SERA.2011.24
34. Kopczyńska, S., Nawrocki, J.: Using non-functional requirements templates for elicitation: a case study. In: 2014 IEEE 4th International Workshop on Requirement Patterns, RePa 2014 - Proceedings, pp. 47–54 (2014). https://doi.org/10.1109/RePa.2014.6894844
35. Younas, M., Jawawi, D.N.A., Ghani, I., Kazmi, R.: Non-functional requirements elicitation guideline for agile methods. J. Telecommun. Electron. Comput. Eng. **9**(3–4 Special Issue), pp. 137–142 (2017)

36. Saeeda, H., Dong, J., Wang, Y., Abid, M.A.: A proposed framework for improved software requirements elicitation process in SCRUM: implementation by a real-life Norway-based IT project. J. Softw. Evol. Process, **32** (2020). https://doi.org/10.1002/smr.2247
37. Farid, W.M., Mitropoulos, F.J.: Novel lightweight engineering artifacts for modeling non-functional requirements in agile processes. In: Conference Proceeding on IEEE SOUTHEASTCON (2012). https://doi.org/10.1109/SECon.2012.6196988