Comparison of modern techniques for analyzing *NFRs* in agile: A systematic literature review

Siraj Un Muneer
CS Dept.
BUITEMS

Dr. Muhammad Nadeem
CE Dept.
BUITEMS

Dr. Bakhtiar Kasi
SE Dept.
BUITEMS

Author Note
This is a self-financed research.
Author may be corresponded to sighage@gmail.com

Abstract

The need for focusing security requirements for software projects is of great significance. There are numerous studies conducted on gathering and eliciting security requirements in requirements management process. The objective for this systematic literature review is to make a comparison of some modern requirements management techniques with classic techniques for managing Non-Functional Requirements (NFRs) in agile Software Methods. We identified 25 research articles and pointed out some weaknesses found in classic requirements management techniques used with agile Methodologies. We examined the weaknesses posed by some requirements management techniques and mapped them with the strengths offered by modern ones. This research contributes in identifying solutions discussed in the literature to overcome the weaknesses found in classic requirements management techniques.

Keywords: agile methodologies, agile, Scrum, NRFs, Software security, software engineering, software architecture, security requirements, SLR

INTRODUCTION

Agile methods for developing software is a particularly new idea, this approach gained attention in past years. Majority of the organizations are adopting agile methodologies. So, this tendency is due the unending necessity of producing state-of-the-art solutions, rapid development, and software with good Return on Investment (ROI). It is preconceived in agile Methods that changing requirements are certain, and thus the development cycle must adapt to this fact. Furthermore, software development teams are required to deliver product to the customer quickly with fewer concerns on extensive planning and documentation (Knauss, 2017).

Some of the characteristics mentioned below have been especially useful in projects using agile methodologies: short deadlines, smaller teams, consistent change in requirements, systems incorporating new technologies. Adopting agile successfully leads to achievement of higher quality software at cheaper rates and enhance developer's morale. (Sachdeva, 2017).

Nevertheless, most agile methodologies have been a light and lucid process, the adoption still often is difficult, which is due the reason that they are not clear themselves, so it is challenging to introduce agile methodologies altering the culture of a company. Special challenges and fundamental organizational changes arise with agile adoption which are necessary for successful outcome (Rindell, 2016) (Camacho, 582-589).

FUNDAMENTALS OF AGILE METHODOLOGIES

Since the dawn of software development, for any specific project development methodologies have been the main focus of life cycle approaches. Since1940, in approaches like structured programming, aspect or object-oriented programming or more recently XP (extreme

programming) there have been noticeable changes on software development paradigm. With each evolution, a new change is introduced and new ways of thinking and analyzing the problem which brings about strength in development cycles. To efficiently use methodologies, the formulated defined process is important to be followed. In conventional software engineering methodologies, there were debilities and shortcomings and agile methods were developed as an effort for improvement. Decades of hard work have resulted in improved agile methodologies which have resulted in improvements to projects; however, they have not proven to be best for all projects or situations. In context of industry, people's work habits have been considerably changed by introducing agile methodologies; these methodologies impart an opposite approach in contrast with classical software development approaches. Many proposed agile methodologies have been in use in industry until today; e.g. XP (Extreme Programming), ASD (Adaptive Software Development), Scrum, and others.

LITERATURE SURVEY

Just-in-Time (JIT) approaches are one of the very famous techniques for managing NFRs in agile projects. A knowledge management framework is proposed in the literature to allow JIT RE to properly meet non-functional requirements on spot. (Knauss, 2017). Topnotch methodologies for example XP (Extreme Programming) aid in developing systems that a system with all functional requirements. However, non-functional requirements are usually missed-out or reminded later and made part of the project quite late in the development (Sachdeva, 2017). Projects with gigantic infrastructure for example cloud and big data NFRs are quite crucial, for which AC (Acceptance Criteria) has to be defined visibly and security and performance in the DoD (Definition of Done) for all related user stories and releases is especially noteworthy. (Sachdeva, 2017). In another study, VAHTI security instructions are proposed for developing a secure ID management system and all the management processes (Rindell, 2016). A study has empirically analyzed factors such as cost, priority and pressure of time were reinforced since the traditional waterfall development models were used (Camacho, 582-589). Technical issues may be resolved by adopting development philosophies like DevOps integration (Camacho, 582-589). Yet another study has analyzed Microsoft Software Development Lifecycle for agile (Ch'oliz, 2015)They have shown a successful synchronization among the independent security team and agile software engineering teams (Ch'oliz, 2015).

Traceability Process Model (TPM) has been introduced to trace the NFRs (Non-Functional Requirements) such as security and performance (Arbain, 228-233). Big up-front design (BUFD) is yet another technique which is needed for security related engineering (Raschke, 2014). On the other hand, to facilitate agile security evaluation process to a high degree in agile security evaluation method for the common criteria standard has been introduced in another study. (Raschke, 2014). A similar study introduced agility into a safety-critical development process (Stephenson, 2006). A Non-Functional Requirements modeling for agile Processes (NORMAP) has been proposed in a study that identifies and links agile Loose cases with agile Use Cases and agile Choose Cases (Farid, 2012). Extensive customer involvement and developer's security awareness and expertise is focused in a study to improve the development process for security (Bartsch, 2011). Security principles integrated in development phases is also proposed which can result in a good analysis and implementation of security features in agile processes
(Azham, 2011). Another integration of a lightweight agile approaches to security risk management in the agile development life cycle is discussed (Franqueira, 2011).

Cross-functional prioritization of processes can be achieved using a replacement of traditional Market Requirements Documents (MRDs) with Fast Track concept to attain agility (Hodgkins, 2007). A selection framework for business has been introduced after comparing several traditional development processes with agile methodologies (Chen, 2007). The aforementioned selection framework proposes to successfully incorporate security requirements and volatile system requirements in agile methodologies.

A position paper has tried to shed light on the way agile practitioners include NFRs such as security requirements in agile projects and came up with a conclusion of the study having some implications for practice and research (Terpstra, 2017). A focus with a view of captured data on investigation of quality requirements' influence on architectural decisions is demonstrated in a study (Kassab, 2017). CEP (Capture, Elicit and Prioritize) methodology is proposed that baselines NORMAP and prioritizes security requirements (Maiti, 2017). A profiling methodology has been proposed in a study that takes personas as a tool to collect data for requirements. The study realizes the concept of anonymized and crowd-sourced personas (Alvertis, 2016).

CEPP (Capturing, Eliciting, Predicting and Prioritizing), a focused study to effectively gather metadata on NFRs from requirements documents (Maiti, 2017). Also, historical trending is used for predicting overlooked additional NFRs in early stages of agile processes.

The use of security assurances cases to maintain a generalized view of security claims as features are being developed. A study that enables the incremental development of security features as well as ensuring security requirements of features are fulfilled (Ben Othmane, 2014). Domain specific modeling languages (DSMLs) are introduced for the domain of security engineering (Eichler, 2012). The languages and implementation requirements have been sketched to report pitfalls and remaining issues with regard to development.

A lightweight quality evaluation method which reflects quality attributes to enhance Non-functional features of an agile approach (Um, 2011). rCOS is adopted in agile software development through convincing examples for improving software trustworthiness (Farroha, 2011). SQUARE (Security Quality Requirements Engineering) is used as an example methodology to consider quality issues like security in the early phases of software development Lifecycle (Zuo, 2010).

RESEARCH METHODOLOGY

An SLR (Systematic Literature Review) helps in discovery and analysis of extensive research available with the pertinent domain of a particular study. The existing research is empirically evaluated and well established within the research community in accordance with predefined criteria. After working on the review results provide scientific evidence by categorizing and classifying relevant studies. The main steps described below were performed:

A. Formulating research questions

The primary focused was on identification of problems in analyzing software security requirements in agile methodologies. Following research questions have been formulated to accomplish the review:

RQ1: What are the weaknesses of requirements gathering techniques used in agile methodologies?

RQ2: How do modern techniques for analyzing NFRs overcome the weaknesses identified in RQ1?

The first question deals with discovering weaknesses of requirement gathering techniques used in agile methodologies.
The second question emphasizes on exploring modern requirements gathering techniques to overcome the weaknesses identified.

B. Search strategy

We considered the most popular scientific research library, IEEE Xplore. The intent was to review the most recent publications from 2008 to 2017, approaching specialized journals and conferences.

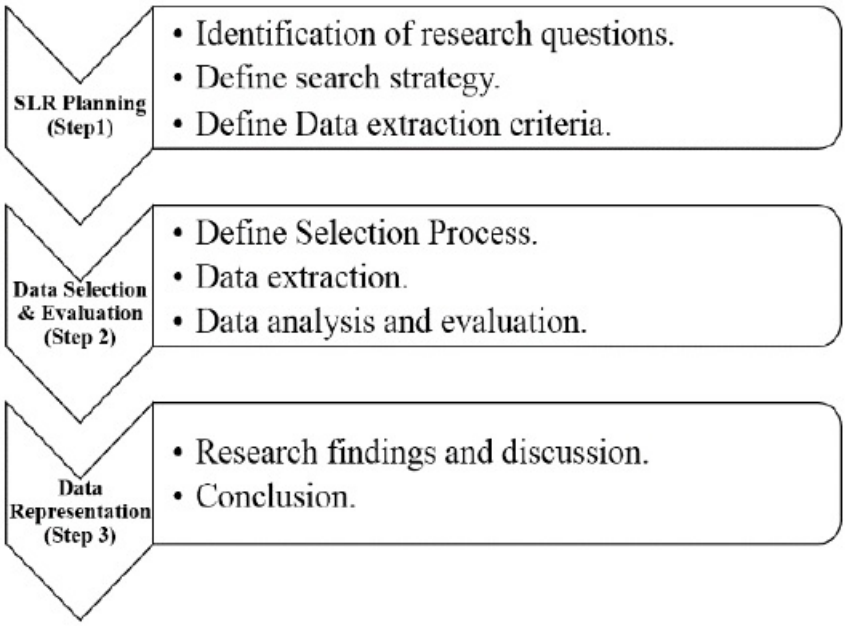

Figure 1

Systematic Literature Review Process

Search structure based on keywords "software security requirements in agile". The search string was looked up in three parts of a full text (body of the paper), keywords, and abstract. Following procedure was used to identify the most relevant studies for this SLR and taking a fast view of the rest of the sections in a paper:

- Choice of the keyword for research
- Look up in the digital library, based upon Inclusion/Exclusion criteria trying with keywords.
- Analyzing every single paper based on its title & abstract.
- Collecting papers fulfilling search criteria.
- Rereading introductions & conclusions.

C. Search based on Title

Firstly, duplicates and irrelevant papers are manually excluded based on titles. After the first stage, only 33 papers remained for further refinement in the next phase.

D. Search based on Abstract

Secondly, analysis of information in abstracts is performed and the papers are categorized along with approach for the analysis of agile methodologies. Research approaches include experiments, case studies, surveys, and reviews. At this stage, the quality of empirical data is not judged. After this stage, 28 papers remained in the list.

RESULTS FROM THE REVIEW

A. Exclusion Criteria

Table 1 includes selected number of papers form IEEE database. We searched for keywords "Software Security Requirements in agile", finding 79 papers in the most famous research library. These papers were reviewed at the level of "Title", reducing the amount to 33. After that, we reviewed the "Abstract", reducing the amount to 28. Finally, looking at the overall view of complete papers, reducing them to 25, which are pertinent for our research.

	Excluded	Included
Initial Search Results		79
Duplicate Studies	7	71
Exclusion based on title	38	33
Exclusion based on Abstract	5	28
Exclusion after viewing full paper	3	25

Table 1
Selected Papers

B. Weaknesses of Requirements Management Techniques

Table 2 shows some weaknesses of requirement management techniques used in agile methods, discussed in literature.

Techniques	Weaknesses
Just-in-time (JIT) (Knauss, 2017)	• NFRs not met on spot • Not all aspects of requirements are covered
Big up-front design (BUFD) (Raschke, 2014)	• It can take up to months • Skips most of NFRs
Rapid feedback (Stephenson, 2006)	• Ignores mostly security requirements • Focuses only FRs
RE in scrum (Azham, 2011)	• Ignores the security risk management activity
Top-down approach based on plando-checkact cycle (Franqueira, 2011)	• Time consuming • Human resource requirement
Upfront, fix planning which drives remaining risk management activities (Franqueira, 2011)	• Extensive feedback required
Documentation-centric approach which relies on documented knowledge (Franqueira, 2011)	• No proper prioritization of requirements

Techniques	Weaknesses
Assumes complete and correct information and consensus about criteria used (Franqueira, 2011)	• Uncertainty of knowledge

Table 2

C. Modern Techniques for Requirements Management

Requirements management is the process of finding, discovering, elaborating, and acquiring requirements for computer software systems. It is generally understood that requirements are managed rather than just captured or collected. Some modern techniques and agile philosophies discussed in the literature are mentioned in Table 3. These techniques were proposed to overcome the weaknesses discussed earlier.

Techniques	Strengths
SecVolution (Knauss, 2017)	• Detects as many know suspicious requirements an submit this much smaller list of requirements to the rare expert for final resolution • In order to extract knowledge from human-made natural language requirements, NLP techniques • Uses an ontology to represent the knowledge
Knowledge Management Framework (Knauss, 2017)	• An ontology plays a central role of managing knowledge • External published warnings (of vulnerabilities) • insights of security experts are encoded in the ontology
Secure Identity Management System based on VAHTI (Rindell, 2016)	• Faster reaction to changes in the requirements and directness of the client feedback • Direct channels to the client were viewed to be very valuable during the implementation
Software Security Testing Process (Ch'oliz, 2015)	• Enforced the Security Team to apply a new methodology • Achieving the desired synchronization of the Security Team with the sprints of the SE Team • Detection of security findings occurs earlier in the lifecycle instead of at the end of the project timeline

Techniques	Strengths
Traceability Process Model (TPM) (Arbain, 228-233)	• Combines both agile and FR and NFR model traceability metamodel traceability • Helps the development team to trace NFRs such as Security and Performance in the system
Capture Elicit Prioritize (Maiti, 2017)	• Retrieves Non-Functional Requirements from requirements documents and diagram/images contained in the docs
Crowdsourced and anonymized Personas (Alvertis, 2016)	• Document requirements and drive the development process through real user profiles of connected services like Facebook, Twitter, Instagram etc.
Capturing, Eliciting, Predicting and Prioritizing (CEPP) (Maiti, 2017)	• Prioritization of NFRs • Improve upon prior studies of NFRs in order to provide effective techniques to prioritize and predict NFRs
Domain Specific Modeling Languages (DSMLs) (Eichler, 2012)	• Supports the development and application of adequate DSMLs, agile approaches and frameworks to provide appropriate tooling are needed • Sketch the language and implementation requirements for our modeling tools, design and implementation consideration • Exports pitfalls and issues pertaining to development of tools for modeling.
Security Quality Requirements Engineering (SQUARE) (Zuo, 2010)	• Emphasis on implementation in initial phases of methodology. • In the business study phase of DSDM, all the steps of SQUARE fit.

Table 3

D. Modern Techniques as solution

We have mapped some identified weaknesses of the classic requirements management techniques with the strengths exhibited by modern techniques in Table 4. Following table shows that the weaknesses posed by classic techniques may be avoided by adapting modern techniques.

Weaknesses of Classic Techniques	Solutions by Modern Techniques
All aspects of requirements are not covered	• Detects as many know suspicious SecVolution detects as suspicious requirements

Weaknesses of Classic Techniques	Solutions by Modern Techniques
Time consuming and extensive feedback required	• Natural-language processing techniques • Ontology to represent the knowledge • Crowdsourced and anonymized Personas technique offers automated collection of requirements using social media profiles • Very fast requirements collection method
NFRs are ignored	• Traceability Process Model (TPM) helps trace NFRs such as Security and Performance in the design
Extensive Human resource involvement	• Retrieves NFRs from requirements documents or images and diagrams using OCR
Uncertainty of knowledge	• KM (Knowledge Management) Framework, an ontology acts as a central role of knowledge management • Externally published threats or vulnerabilities' warnings • security experts' insights encoded in ontology
No proper prioritization of NFRs	• CEPP provides prioritization of NFRs • improves effectiveness of techniques for prioritization and prediction of NFRs

Table 4

CONCLUSION

This paper presented a comprehensive comparison on some state-of-the-art requirements management methodologies. Some of the methodologies discussed in the literature were being used with agile methodologies for years. The weaknesses of such methodologies were also mentioned in the literature and this research has incorporated the discussed weaknesses.

Most of the researchers have proposed many new methodologies for requirements management and they have discussed their strengths. Mostly, proposed methodologies have been tested on projects and applied to some scenarios to test their efficacy. Some of them have no evidence of their effectiveness.

We have mapped the weaknesses with the strengths and identified that modern techniques can serve as a solution in overcoming the discussed weaknesses. The proposed requirements management techniques have a big potential and can result in effective quality requirement management. Security requirements are usually skipped or omitted due to many factors, but

methodologies discussed in this research takes quality requirements in perspective and makes it significant to be incorporated at design time.

ACKNOWLEDGEMENT

Special thanks to Dr. Muhammad Nadeem for the inspiration and undivided attention which helped a lot in the completion of this study. We are especially thankful to one of the authors of an article which has been cited, for providing us with the full version of the paper which was yet to be published.

REFERENCES

Alvertis, I. a. (2016). Using crowdsourced and anonymized Personas in the requirements elicitation and software development phases of software engineering. *2016 11th International Conference on Availability, Reliability and Security (ARES)*, 851--856.

Arbain, A. F. (228-233). agile non functional requiremnents (NFR) traceability metamodel. *8th Malaysian Software Engineering Conference (MySEC), 2014*, 2014.

Azham, Z. a. (2011). Security backlog in Scrum security practices. *2011 5th Malaysian Conference in Software Engineering (MySEC)*, 414-417.

Bartsch, S. (2011). Practitioners' perspectives on security in agile development. *Sixth International Conference on Availability, Reliability and Security (ARES), 2011*, 479-484.

Ben Othmane, L. a. (2014). Using assurance cases to develop iteratively security features using scrum. *2014 Ninth International Conference on Availability, Reliability and Security (ARES)*, 490--497.

Camacho, C. R. (582-589). agile team members perceptions on non-functional testing: influencing factors from an empirical study. *11th International Conference on Availability, Reliability and Security (ARES), 2016, IEEE*, 2016.

Chen, J. Q. (2007). Light-weight development method: a case study. *2007 International Conference on Service Systems and Service Management*, 1-6.

Ch'oliz, J. a. (2015). Independent security testing on agile software development: a case study in a software company. *10th International Conference on Availability, Reliability and Security (ARES), 2015*, 522-531.

Eichler, J. a. (2012). Supporting security engineering at design time with adequate tooling. *2012 IEEE 15th International Conference on Computational Science and Engineering (CSE)*, 194--201.

Farid, W. M. (2012). The Normap methodology: Lightweight engineering of non-functional requirements for agile processes. *2012 19th Asia-Pacific Software Engineering Conference (APSEC)*, 322-325.

Farroha, D. L. (2011). agile development for system of systems: Cyber security integration into information repositories architecture. *Systems Conference (SysCon), 2011 IEEE International*, 182--188.

Franqueira, V. N. (2011). Towards agile security risk management in RE and beyond. *2011 First International Workshop on Empirical Requirements Engineering (EmpiRE)*, 33-36.

Hodgkins, P. a. (2007). agile program management: Lessons learned from the verisign managed security services team. *agile Conference (AGILE), 2007*, 194-199.

Kassab, M. (2017). A Contemporary View on Software Quality Requirements in agile and Software Architecture Practices. *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, 260--267.

Kitchenham, B. a. (2009). Systematic literature reviews in software engineering--a systematic literature review. *Information and software technology*, 7-15.

Knauss, E. a. (2017). Quality Requirements in agile as a Knowledge Management Problem: More than Just-in-Time.

Maiti, R. R. (2017). Capturing, eliciting, and prioritizing (CEP) NFRs in agile software engineering. *SoutheastCon, 2017*, 1-7.

Raschke, W. a. (2014). Supporting evolving security models for an agile security evaluation. *2014 IEEE 1st Workshop on Evolving Security and Privacy Requirements Engineering (ESPRE)*, 31-36.

Richard Rabin Maiti, A. K. (2018). agile Software Engineering & The Future of Non-Functional Requirements. *Journal of Software Engineering Practice*, 1-8.

Rindell, K. a. (2016). Case study of security development in an agile environment: building identity management for a government agency. *IEEE*, 556-563.

Sachdeva, V. a. (2017). Handling non-functional requirements for big data and IOT projects in Scrum. *7th International Conference on Cloud Computing, Data Science & Engineering-Confluence, 2017*, 216-221.

Stephenson, Z. a. (2006). Health modelling for agility in safety-critical systems development. *IET*.

Terpstra, E. a. (2017). agile Practitioners' Understanding of Security Requirements: Insights from a Grounded Theory Analysis. *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, 439-442.

Um, T. a. (2011). A quality attributes evaluation method for an agile approach. *2011 First ACIS/JNU International Conference onComputers, Networks, Systems and Industrial Engineering (CNSI)*, 460--461.

Zuo, A. a. (2010). Research of agile software development based on formal methods. *2010 International Conference on Multimedia Information Networking and Security (MINES)*, 762--766.