PUCRS

ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

LEANDRO RIPOLL SALDANHA

MISUSER STORIES: AN EXTENDED AGILE FRAMEWORK FOR HANDLING SECURITY REQUIREMENTS IN AGILE SOFTWARE DEVELOPMENT

Porto Alegre

2019

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF TECHNOLOGY
COMPUTER SCIENCE GRADUATE PROGRAM

MISUSER STORIES: AN EXTENDED AGILE FRAMEWORK FOR HANDLING SECURITY REQUIREMENTS IN AGILE SOFTWARE DEVELOPMENT

LEANDRO RIPOLL SALDANHA

Dissertation submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fullfillment of the requirements for the degree of Master in Computer Science.

Advisor: Prof. Dr. Avelino Franciso Zorzo

Porto Alegre
2019

Ficha Catalográfica

S162m Saldanha, Leandro Ripoll

Misuser Stories : an Extended Agile Framework for Handling Security Requirements in Agile Software Development / Leandro Ripoll Saldanha . – 2019.
81.
Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Avelino Francisco Zorzo.

1. Security requirements. 2. Agile development. 3. Scrum. 4. Field study. I. Zorzo, Avelino Francisco. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecária responsável: Salete Maria Sartori CRB-10/1363

Leandro Ripoll Saldanha

Misuser Stories: an Extended Agile Framework for Handling Security Requirements in Agile Software Development

This Dissertation has been submitted in partial fulfillment of the requirements for the degree of Master of Computer Science, of the Graduate Program in Computer Science, School of Technology of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on August 23, 2019.

COMMITTEE MEMBERS:

Prof. Dr. Kleinner Silva Farias de Oliveira (PPGCA/UNISINOS)

Prof. Dr. Sabrina dos Santos Marczak (PPGCC/PUCRS)

Prof. Dr. Avelino Francisco Zorzo (PPGCC/PUCRS - Advisor)

HISTÓRIAS DE MAU USUÁRIO: UM FRAMEWORK ESTENDIDO PARA LIDAR COM REQUISITOS DE SEGURANÇA NO DESENVOLVIMENTO ÁGIL DE SOFTWARE

RESUMO

Equipes de desenvolvimento ágil são multidisciplinares, tendo como foco a entrega contínua de produto funcional e testável em intervalos previamente definidos. Para atender às expectativas dos usuários, a equipe de desenvolvedores deve levar em consideração requisitos funcionais e não funcionais. Os requisitos funcionais estão diretamente ligados às regras de negócio e são mais visíveis do que os requisitos não funcionais, os quais podem ser subjetivos e demandar conhecimentos mais amplos. Desta forma, requisitos de segurança, em muitos casos, são classificados como não funcionais, tendo origem em fontes diversas e demandando conhecimentos que vão além do negócio e das técnicas de desenvolvimento. Neste sentido, existem críticas sobre a falta de cuidado do ágil com relação aos requisitos não funcionais, em especial requisitos de segurança. Esta pesquisa visa justamente a intersecção entre as áreas de desenvolvimento ágil de software e segurança da informação, propondo algumas práticas para lidar com requisitos de segurança em equipes ágeis de desenvolvimento. A principal contribuição deste trabalho consiste em introduzir um novo artefato denominado história de mau usuário, com suas regras e práticas recomendadas. Este novo artefato visa definir um framework ágil estendido para ampliar o foco dado aos requisitos de segurança, tornando tais requisitos mais explícitos.

Palavras-Chave: Requisitos de segurança, Desenvolvimento ágil, Scrum, Estudo de campo, Histórias de mau usuário.

MISUSER STORIES: AN EXTENDED AGILE FRAMEWORK FOR HANDLING SECURITY REQUIREMENTS IN AGILE SOFTWARE DEVELOPMENT

ABSTRACT

Agile teams are multidisciplinary and focus on delivering working and verifiable software within predefined periods in a continuous way. In order to satisfy users' expectations, the agile team has to deal with functional and non-functional requirements. The functional requirements are directly linked to the business rules, being more noticeable than the non-functional ones, which may be subtle and might require a wider range of knowledge. Thus, in many situations, security requirements are classified as non-functional requirements. These can derive from a variety of sources, requiring expertise beyond business or even development techniques. Therefore, there are some critics about the way agile deals with non-functional requirements specially the security ones. This research aimed to put together agile software development and information security areas, by proposing a set of practices to cope with security requirements in agile development teams. The main contribution of the present work is introducing a new artifact named misuser story, composed of related rules and recommended practices. That new artifact intends to define an extended agile framework for promoting the consideration on security requirements, making these requirements more explicit.

Keywords: Security requirements, Agile development, Scrum, Field study, Misuser stories.

LIST OF FIGURES

CONTENTS

1. INTRODUCTION

The Agile Manifesto has influenced Software Engineering since it was unveiled at the beginning of the 2000's. According to the Manifesto, working software is more important than thorough documentation. Furthermore, the agile framework welcomes changes at any point of the project development [15]. That means a drastic turning point when compared with former methodologies adopted so far [82].

Although the agile methodologies have been applied in software development, several issues are still not dealt with appropriately by those methodologies, as software security requirements for example. On one hand, software security is an increasing demand and extremely relevant considering information is a worthy asset for individuals and organizations [62]. On the other hand, considering security within agile software development, it has been said that agile focus on delivering system features may interfere negatively on software security [18] [22]. Agile teams seem to undervalue any kind of effort not directed to the functional requirements [18] [22].

Some approaches have been proposed in order to deal with security requirements in agile software development projects. There are also some practices considered agile-friendly [8] [66]. Despite of what are the security agile approaches, studies have shown that the majority of the security vulnerabilities are found in the application layer [38]. The importance of discussing how to increase the security level within agile environments instead of discussing whether agile and security are compatibility or not is undeniable [13].

In the last years, it is possible to notice many episodes related to the software security issues, including government regulations and laws. That is the case of VATHI [82], the security guideline written by the Finish government in 2014. In the same way, the European Union (EU) has launched the so called General Data Protection Regulation (GDPR), which has come into force in May, 2018 [83] [84] [29].

Given agile tends to undervalue non-functional requirements [18] [22] and also considering the increasing concerning about security and privacy issues [82] [83] [84] [29], the present work aims to research the intersection between Software Engineering and Security Information areas, *i.e.* the agile software development and secure software development. The goal of this work is suggesting a set of agile-friendly practices that will be derived from the theoretical basis together with the practical knowledge generated by some real agile projects as well as professional experience. Such practices are intended to deal with the security requirements in agile development.

As a general goal, the stated research question that will guide the present work is **"General Research Question - GRQ - How to integrate security requirements in agile software development?"**. Three intermediate steps were also stated to help the GRQ

achievement. Such steps represent the three specific goals **(Specific Goals - SG)** that follow:

SG1 – To identify the main sources which generate security requirements in an agile software development project;

SG2 – To identify the recommended practices for agile teams to cope with security requirements;

SG3 – To suggest a set of lightweight practices, which are compatible with the agile philosophy.

The specific goals listed above, as well as the general one, were answered with the help of a prior Systematic Mapping Study (SMS) combined with a field study. The research methodology will be thoroughly explained in Chapter 2.

After presenting the theoretical basis and also after proceeding with the field study based on interviews and document analysis we expect to present a set of proposed practices as a final result of this work. The proposed practices are intended to deal with security requirements in agile environments, being supported by a new agile artifact surrounded by related practices, rules and tools. That represents a new extended agile framework for making security requirements more explicit.

That new artifact named as misuser story was drafted to be introduced in the Scrum flow, but not being restricted to be used just at this agile framework. The set of proposed practices can be adapted to be used at any agile framework, including the new artifact. The misuser stories as well as the related practices will be presented and better discussed at Chapter 6.

In order to put the research subjects into their context, it is important to show the state of the art related to the areas above mentioned, pointing out the links and challenges regarded to security within agile projects. Therefore, the remaining text is organized as it follows. Chapter 2 will demonstrate the methodological procedures and basis. Chapter 3 will briefly describe some agile concepts, the Scrum framework and the related work regarding the security and agile areas.

Chapter 4 will show the security requirements in agile environments, the main security agile approaches and some security frameworks. Chapter 5 brings the interviews and document analysis, followed by Chapter 6 presenting the set of suggested practices. At last, Chapters 7 and 8 present the main conclusions, the limitation and suggestions for further research.

2. METHODOLOGY - FIELD STUDY

Before presenting the adopted methodology for this research, it is important to remind the reader of the posed general goals as well as the specific ones. In compliance with the title of this study, the general goal and research question (GRQ) is **"GRQ - How to integrate security requirements in agile software development?"**.

In order to reach the general goal, three specific goals (SG) were set. The first one is **"SG1 - To identify the main sources which generate security requirements in an agile software development project"**. The second one is **"SG2 - To identify the recommended practices for agile teams to cope with security requirements"**. The last one is **"SG3 - To suggest a set of lightweight practices, which are compatible with the agile philosophy"**.

Prior to this study, a Systematic Mapping Study (SMS) [26] was conducted and published as a technical report [87]. It has helped to respond the specific goals SG1 and SG2. An SMS is a secondary study recommended to obtain a general view of a specific subject [26] [52] [77]. In this case, the subject is security requirements sources and the related agile practices to cope with such requirements.

Following the SMS process [26] [52] [77], it was also adopted the snowballing procedure that consists of using the reference list and the citation list to provide other sources of research [99].

Still considering the SMS, we have answered three proposed questions in order to understand what is being researched in the intersection between agile and security areas. The SMS questions are presented as it follows:

- What is being researched related to agile and security requirements?
- What are the artifacts, tools and methodologies used by agile for dealing with security requirements?
- What is the adherence degree of agile when the subject is to develop secure software?

We have set a search string for finding the papers used in the SMS, following the Kitchenham and Charters guidelines [53]. That consists on the the expected population, intervention and outcomes, given we have intentionally chosen to ignore comparison and context structures. The research string is presented as it follows:

- Population: (Software Engineering OR Software Development)
  AND
- Intervention: (Agile Security OR Security Agile OR Agile Security Requirements OR Security Requirements Agile Development)
  AND

- Outcome: (Practices OR Best Practices OR Framework OR Methodology)

We have focused on IEEE, ACM and Science@Direct as the main research databases. These databases were also selected for supporting our search string. Google academic was also used as a search engine, especially for the snowballing process.

The SMS and snowballing processes have generated a technical report [87] that will be used as one theoretical source for answering the goals of this work, including the set of suggested practices. Combined with the theoretic basis of the present research, we have adopted a field study methodology in order to capture some practical (primary) knowledge to our suggestion.

Software Engineering is a mixture of tools, techniques and human behavior [28]. Such complexity of aspects might require a qualitative research. By adopting qualitative methods, it is possible to obtain richness of details [81] [85]. It can also be considered as an exploratory research because its goal is to understand a subject and to bring it into the light with a number of details [85].

As a study strategy, a field study is a technique where a researcher makes observations of natural ongoing phenomena while interfering or disturbing the studied system as little as possible [60]. The essence of a field study is that the behavior system under study is natural, which means that it would happen even if it was not being observed in a study [60]. Although no study is completely unobtrusive, it is mandatory that a field study remains as unobtrusive as it can possible be [60].

According to McGrath [60], the field study research strategy has parallels in other areas such as anthropology, sociology and organizations. Specifically in organizations area, many case studies could be mentioned as examples of the field study strategy [60]. The field study strategy seems to be appropriate for the present research, given its essence as explained before.

In order to achieve the proposed goals, this research used the field study methodology based on interviews and document analysis. Given the similarities between a field study and a case study (organizational area), authors related to case study will be evoked in order to corroborate the adopted study strategy.

Yin [100] agrees that a case study is indicated to observe current phenomena within their context. Roesch [85] agrees with Yin, adding that a case study is highly recommended when it comes to a research of qualitative nature. Gil [39] believes flexibility is one of the main advantages of a case study, therefore he does not suggest the usage of a strict script.

Some of the data sources indicated by Yin [100] are documents and interviews. The author think that using more than one source is a good practice in a case study. Both interviews and document are adopted in this work as its data source.

As documents to be analyzed, it can be used any reports, manuals, guidelines, electronic sheets and even exchanged e-mails. Documents generated by the company related to the research subject are specially useful to provide relevant information [100].

Semi-structured interviews provide the perspectives of the interviewees about a subject. It is stated a general guideline, but the interviews can be conducted in a spontaneous way by the interviewer. They can also be recorded in order to be used in a future content analysis [100]. These are the adopted procedures in this work.

Marconi and Lakatos [34] believe that a sampling process for choosing the interviewees should privilege the ones who have leadership traces within the company, considering factors such as expertise and experience in the researched area. Gil [39] states that the researcher could choose the interviewees, not randomly but according to their relevance for the research. Such technique is named by the author as intentional sampling [39]. It is exactly how the interviewees were chosen, considering that all Scrum Masters of two segments (business areas) in a company were interviewed.

Based on the methodology and techniques mentioned until now, scrum masters were interviewed in two segments of the studied company. Documents of real projects were also analyzed in order to obtain more information about the researched theme. The chosen organization will be kept anonymous, but it is possible to state that it is a multinational company. It is one of the the biggest software development companies in Latin America, being amongst the six biggest companies of the world in its area (considering revenue and market share).

The company is specialized on enterprise management systems, holding a significant percentage of the market share in South America and also in Latin America. Their solutions are directed to process automation related to companies in a variety of segments such as financial services and health care. Both financial services and health care segments are the target of this study, stating that four agile teams are the focus in the field study.

One team belongs to the financial services segment and the other three are linked to the health care segment. These are all the agile teams and Scrum Masters in that two segments, representing the universe of both segments (related to agile teams). The chosen company, as the unit of study in the present field study, is relevant for what it represents in the global market of software development (mainly in Latin America).

As already mentioned, the present field study is based in interviews, therefore a script for the semi-structured interviews was set. The following script was applied to four Scrum Masters during the interview process. Considering the interviewees, three Scrum Masters are set in the health care segment and one Scrum Master belongs to the financial services segment. As the data collection technique is the semi-structured interview, the questions below are only a guideline, a script to be followed. It can be altered along the interviews if the interviewer feels it will enrich the data collected [85] [39] [100].

Roesch [85] believes the main goal of an interview is to uncover the perception of the interviewees regarded to situations, issues or assumptions brought by the researcher. Gil [39] agrees adding that the questions should provide flexibility. The questions are presented in the list below.

1. What is your age?
2. How much experience (in years) do you have, related to agile development?
3. How much experience (in years) do you have in this company, related to agile development?
4. What kind of training do you have in agile development?
5. What is your understanding about security requirements?
6. In your opinion, what are the main sources which generates security requirements in a project?
7. What practices do you do to deal with security requirements?
8. What would be suggested practices to deal with security requirements, different from the used now?
9. Provide a brief explanation on what is the agile process your team follows as well as the documents produced along this process.

Questions 1 to 4 are used to characterize the interviewed scrum masters related to their agile experience time and agile knowledge. Questions 5 to 8 try to capture the understanding of the scrum masters about security requirements, their sources and the current practices for dealing with such requirements. Question 9 intended to capture the process followed during the sprints and what kind of documents are produced along the project.

In order to analyze the data collected from documents and interviews, the content analysis technique was adopted. Bardin [12] states that content analysis is a methodology to evaluate data in a qualitative way. Such technique is specially helpful when analyzing the interviews results as well as the documents. Bardin [12] defines content analysis as a set of systematically applied techniques for analyzing communication (messages). Such process aims to reach knowledge through inferences derived from indicators built over the analyzed messages [12].

Roesch [85] believes the document analysis complements the interview analysis, providing a second source of data. Yin [100] agrees with Roesch [85], adding the analyzed documents can be both formal and informal. Gil [39] goes forward by summing analyzed data

should be categorized in order to make it easier to link them with the theoretical foundation. Godoy [40] believes that the content analysis as proposed by Bardin [12] is a methodological technique that can be applied in a variety of speeches and communication types. In content analysis, the researcher intends to understand characteristics, structures or models behind the message fragments. The researcher seeks for both explicit and veiled meaning when proceeding the content analysis [40].

Bardin [12] states that content analysis is composed by three phases. The first one is named pre-analysis, consisting on organizing how the work will be done. The adopted procedures should be established here, though they may be flexible. At this phase the sources are chosen following the rules stated by Bardin [12], which means they have to be representative, not conflicting, complete and relevant for the research theme.

In the first phase the researcher has the initial contact with the material, exploring them and reading them in a non-systematic way. The main goal here is grouping subjects according to their similarities [12]. In this research, at this point the themes were created in order to group the excerpts extracted from the analyzed texts.

The second phase is called material exploration, where the counting, grouping and classification rules are defined related to the text excerpts [12]. For the present research, the themes were grouped into categories at this point of the analysis.

The last phase is related to the results treatment, when the analysis gains relevant and valid meaning. Here the researcher adds inferences and interpretation based on the raw data. It goes beyond the explicit content because the researcher has to reach the implicit messages, that ones which are not directly written in the documents and transcriptions [12].

Still considering the content analysis process we have also used the card sorting technique [91], aiming to help us on creating categories and themes based on the data analysis process. We have adopted an open card sorting process, as the categories have emerged freely from the content analysis [91]. The chosen card sorting procedure is also categorized as individual and manual, for categories have been provided by the researcher analysis and for not being used software during this process [91].

Being ready the data collection and the data analysis, a set of suggested practices was elaborated and presented in Chapter 6. Such practices were based on the results of this study, being introduced to the interviewed scrum masters at the end of this research. That presentation has generated the first impressions of the suggested practices provided by the interviewed scrum masters. Such impressions are discussed in Section 6.3. Figure 2.1 presents an overview of the research schema in a timeline perspective.

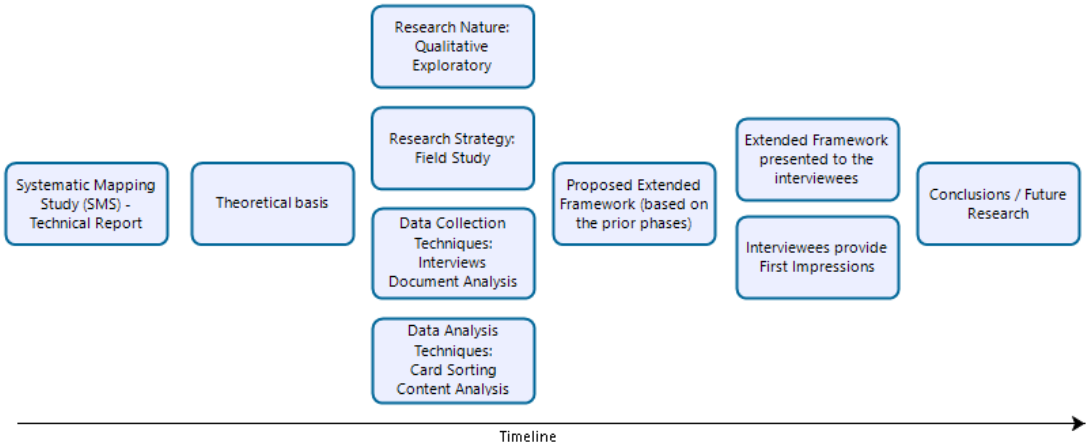

Figure 2.1 – Overview - Research Timeline

After presenting the methodology and the scientific procedures involved, Chapter 3 will approach the concepts related to agility. Together with the security concepts showed at Chapter 4, that will be the theoretic ground of this work.

3. AGILITY

Agility was raised at the beginning of this millennium as a different way to deal with software development. As it is suggested by the name, it was thought to bring agility into software development projects by approximating them to the real world of constant changes [15][79] [82]. The next sections will describe the Agile Manifesto, detail the Scrum methodology and describe some related work that brings agile methodologies and security requirements together.

3.1 Agile Manifesto

The Agile Manifesto [15] has drastically changed the way in which software is developed. It has surfaced in 2001 with the main goal of delivering sooner a working piece of software and adding value to the business during all the software development process. The Manifesto emphatically states that it values communication over process, working software over dense documentation, collaboration over contract, and change response over following a plan. It is important to note that if one values working software over detailed documentation, it does not mean that one does not value documentation at all. It only means that documentation does not deserve the most of one's efforts, which should be directed towards producing working software [15].

The Agile Manifesto is composed by twelve principles that orient the agile philosophy, distinguishing agile from the others. The twelve principles are presented below [15]:

1. Customer satisfaction through early and continuous delivery of valuable software;
2. Welcome change requirements, even late in development;
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale;
4. Business people and developers must work together daily throughout the project;
5. Build projects around motivated individuals, give them the environment and support they need and trust them to get the job done;
6. Face-to-face conversation is the most efficient and effective method of conveying information to and within a development team;
7. Working software is the primary measure of progress;
8. Sustainable development – the sponsors, developers and users should be able to maintain a constant pace indefinitely;

9. Continuous attention to technical excellence and good design enhances agility;
10. Simplicity – the art of maximizing the amount of work not done is essential;
11. Self-organizing teams;
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Compared to the traditional waterfall model [82], agile is drastically different. Agile tries to anticipate scope changes while waterfall tries to reach efficiency by preventing them. Waterfall tries to define the entire scope during the planning phases, before the beginning of the development activities. On the other hand, theoretically speaking, agile deals well with an opened scope where software features and functionalities are defined along the project through agile iterations [79] [82].

Other frameworks along the time have also recognized the agile philosophy. The Project Management Body of Knowledge (PMBOK) is a world-widely used guide for project management, including software development projects [79]. Its most recent edition has also recognized the importance of agility, mentioning it as a methodology for micro management in the project life cycle. Within each chapter of the PMBOK newest edition, there are recommendations for incorporating agile practices along the project (in all knowledge areas of the guide, which is in its sixth edition) [79].

When approaching non-functional requirements, even the traditional waterfall model has already noticed the complexity of managing requirements that are not directly linked to the functional requirements (as the security requirements). Such kind of requirement may be generic and demand a range of additional knowledge beyond the software development or even the business rules [30] [68].

More recently, a new concept related to the agility was tailored. The term DevOps [49] is a blend of the words development and operations. It recognizes the gaps between the agile software development and the operations represented by the professionals who put and maintain the software in a productive environment [49].

According to this new concept, there is a conflict between the development team, who wants to deliver new functionalities as soon as possible, and the operational team, who wants to stabilize the system as a whole. DevOps deals with this kind of problem considering there is a DevOps integration, where development influences operation and operation influences development. Therefore, operations may also demand requirements to the development team, including the security requirements [49].

Agility has opened the doors for a variety of new possibilities and many agile methodologies have arisen since the Agile Manifesto. DevOps has brought the operational cycle of the software into the agile world, as previously explained. Section 3.2 will present some of the most adopted agile methodologies, focusing on the framework known as Scrum.

The already mentioned SMS (Systematic Mapping Study [87]) has shown Scrum as the most adopted agile framework. The company who was analyzed in the field study also adopts Scrum as its agile framework and these are the reasons we will focus on it.

3.2 Agile Methodologies

Since the Agile Manifesto, many different agile methodologies have emerged, such as Scrum [51], Extreme Programming (XP) [16], Feature Driven Development (FDD) [8], Kanban [4] and others. All these methodologies follow the agile principles stated in the Agile Manifesto. Scrum is one of the most adopted agile methodologies in software development projects [51] [97] [82] [83].

The Scrum framework has adopted all the agile philosophy, creating some important concepts for implementing the agility. The main practices introduced by Scrum will be briefly discussed in the next paragraphs. The first one is the concept of user stories, which specifies system requirements by the user's point of view. The user stories concept was first introduced in agile by XP methodology. It is largely adopted even by practitioners of other agile methodologies such as Scrum [16]. After having the users stories, in other words the user requirements, the development team is responsible for transforming them into software features. They use a process known as refinement to detail each user story (requirement) [82] [71] [43] [3].

Related to user stories, some companies have adopted a specific pattern named Gherkin for writing such stories [72] [101]. In fact, Gherkin is a Domain Specific Language (DSL) which represents an open source tool for test automation based on business-readable specifications [101] [33]. It disciplines how to write user stories by providing syntax and structure [72]. It is important to state that Gherkin is not a native practice of Scrum framework.

Following the Scrum cycle, an artifact named product backlog is created at the beginning of the project. The product backlog includes all the available user stories, all the features to be developed. In the next step, some of these stories are selected, classified and prioritized by the development team. The selected stories are moved from the product backlog into the sprint backlog. A sprint is a fixed amount of time where the team will develop and deliver the picked user stories (sprint backlog) as a piece of workable and testable software [82] [71] [43] [3].

The definition of done is another important concept of Scrum. It is a group of consistent criteria that defines when an item can be considered ready to be delivered and accepted by the users. The increment of software delivered at each sprint must match the definition of done [82] [71] [43] [3]. There are also roles and events, which are intended to keep the agility flowing from the first to the last sprint.

The Scrum Master is the responsible for maintaining the agile framework running as well as untying the knots that block the performance of the development team. The Product Owner is the one who deeply knows the business rules, being able to point the system functionalities, to define and to prioritize the product backlog as well as to clarify doubts of the development team. [82] [71] [43] [3].

The development team is a self-organized group of three to nine people who will develop, test, document and deliver the pieces of working software along the project. The team also should have all the required knowledge in order to cope with the entire cycle of the project or product [82] [71] [43] [3].

The Scrum's events are the sprint planning, the daily scrum, the sprint review and the sprint retrospective. In sum, the sprint planning is the moment where the team selects the user stories for the sprint backlog and also set how the work will be done. The daily scrum is a short meeting amongst the development team members, normally at the beginning of the day, where each one briefly responds what was executed the day before, what will be performed in that day and what are their impediments [82] [71] [43] [3].

The sprint review is a moment to validate what was delivered in the current sprint. Sprint retrospective is another moment for the development team to evaluate and discuss ways to improve performance on the following sprints [82] [71] [43] [3]. Figure 3.1 provides an overview of the Scrum framework.

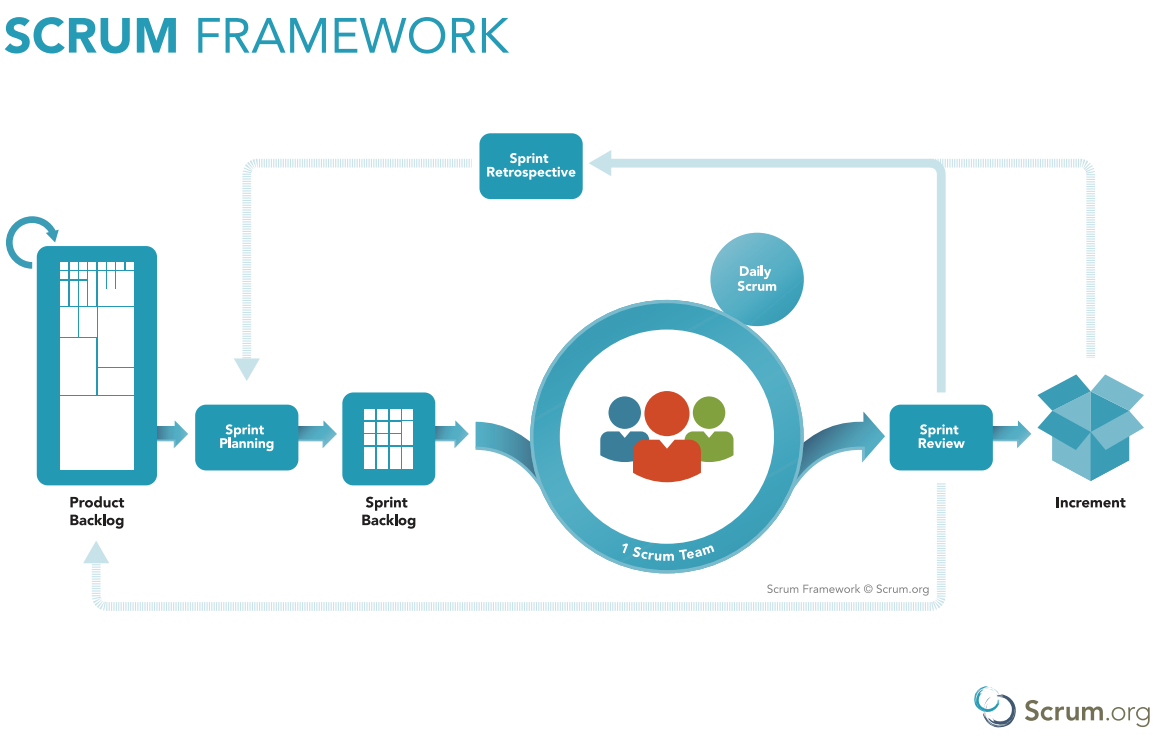

Figure 3.1 – The Scrum Framework [71]

The current section has presented the basic concepts of a recognized agile framework named Scrum, which is the focus here for being one of the most used agile methodologies, as mentioned before. Furthermore, Scrum is the adopted agile methodology within the studied corporation. Section 3.3 will present some important concepts related to the security requirements in an agile environment, as well as similar work in the intersection between both areas (security requirements and agile development).

3.3 Background and Related Work - Agile and Security

Information security cannot be considered an add-on feature when dealing with software development, given that information is a worthy organization asset. Companies have to mitigate risks related to attacks, missuses and any kind of threat [62]. It seems to be clear that security should be seriously taken into account during the software development cycle, but agile development members frequently view the traditional security engineering process as an additional effort. There seem to be a tendency of questioning such kind of effort within agile teams [22] [18].

The concept of adequate security is not a consensus and each organization has its own patterns and policies. Adequate security is directly linked to the risk tolerance a company intends to take [2]. Approaches such as Microsoft Security Development Lifecycle (SDL) for Agile [66] have proposed integrating security activities into agile development. However, security has been approached in different ways by the organizations who have adopted agile [18]. SDL for agile will be detailed in the Section 4.2.

Studies have shown some security practices considered compatible with agile by different organizations [90] [11] [8]. Eliciting security requirements, risk analysis and penetration testing were mentioned as agile compatible practices [11] [8].

A Systematic Mapping Study (SMS) [87] has presented an overview on the intersection between agile and security areas. A sample of 38 papers related to such intersection was analyzed in order to answer what is being researched within the area. It has also provided understanding on how agile has been dealing with security requirements [87].

We have found there are just a few primary studies, being the vast majority of them secondary ones [87]. Extending the agile framework seem to be the preferable way of coping with security requirements in agile projects. An extended framework stands for any kind of practice or modification introduced in the original agile framework, aiming to increase the level of security in the products produced by an agile project [87].

Security awareness and security stories were two other practices found to be used within agile projects when dealing with security requirements [87]. Security awareness can

be understood as training the development team for dealing with safe development, keeping them aware of the subject relevance [87].

Security story is a way of focusing on security requirements by using an agile familiar concept named user stories [87]. Security stories are user stories built for handling security requirements [87].

Fong and Okun [38] have shown that more than 70 percent of all security vulnerabilities are in the application layer. That give us a strong base to convince agile teams about security relevance, it does not matter whether agile and security frameworks are compatible or not. The real issue is how to increase security level in agile environments [13].

While searching for related work for agile and security requirements, we have found some researches approaching how to comply agile frameworks to government security regulations. In the last decade some countries have started a movement towards software security. The so called VAHTI is one example of such government regulation over software security issues.

The Finish government has written a security guide named VAHTI, which is a Finish acronym that stands for the Government Information Security Management Board. Since 2014, the developed software have to be compliance with VATHI for any kind of communication established with the Finish government [82]. Rindell et al. [82] has researched how to secure the Scrum agile framework for adhering to VAHTI.

There are other examples of government regulations regarded to security issues, which should also be taken into account in agile environments. In 2016, European Union (EU) has published the so called General Data Protection Regulation (GDPR) [29]. That law sets the EU citizens have the right over their personal data, being valid all over EU and applied to any company that manipulates EU citizens' data. The newest version has just taken into its full effect within last May, 2018.

GDPR states that all EU inhabitants must have complete access to their personal data. They have to be able to consult, alter and even delete their information from a given application. Companies who are proved not to be in compliance with GDPR are susceptible to fines that can reach four percent over their global revenues [29].

Brazil has also launched his regulation regarded to data protection. The so called LGPD, which stands for General Data Protection Law in Portuguese, regulates how people and companies have to deal with personal data [24]. Deloitte [35] states that the Brazilian LGPD was signed by the president in august 2018. It was inspired on the European GDPR, imposing that companies will have 18 months to be comply with the law and posing fines up to 2 percent of the organization revenue [24] [35].

Table 3.1 brings a brief overview on what has been studied in the intersection between agile and security requirements. It represents the perspective of agile studies on how to tackle security requirements in agile environments.

Table 3.1 – Agile Perspective - Overview

Issue Description	Strategy to cope with
Security issues undervalued	Some framework such as SDL for agile [22] [62] [13] [18] [66] [87]
Security activities not agile compatible	Risk Analysis, Penetration tests [18] [90] [11] [8] [87]
Lack of primary study	Proceed with more primary studies within the intersection between agile and security [87]
Lack of security awareness	Training on security, Security experts to the team [9] [13] [73] [87]
Low level of focus on security requirements	Security Stories [5] [90] [13] [95] [10] [87]
Increase of security relevance (VATHI, GDPR, LGPD)	Training on security, Security experts [2] [82] [87]

Considering the low amount of primary studies in the intersection between agile and security requirements [87], we intend to contribute by adding a primary one. We think it is important to observe the phenomenon in its natural environment in order to provide pragmatic suggestions for dealing with security requirements in agile environments.

We also intend to make security requirements more visible for the entire agile team through security awareness. We believe making security requirements more explicit is one possible way of spreading the required security knowledge as well as dealing with security issues along the agile cycle. By proposing a new artifact named misuser story, we want to make security requirements more explicit for the whole agile team.

The scenario presented until now has provided us a good perspective on the relevance of security requirements in agile software development. Therefore, it is important to understand the existing software security frameworks in order to combine agility with security. Some of these frameworks will be discussed in Chapter 4.

4. SECURITY

The present research focus in the intersection between security requirements and agile software development. After approaching agile concepts, now it is time to highlight security requirements, security agile approaches and security frameworks. Section 4.1 starts with security requirements in agile development.

4.1 Security Requirements in Agile Development

Haley et al. [44] argues that security requirements are constraints to the functions of a system. They can limit what is allowed and who has the right to perform some activity [44]. Tondel et al. [94] believe that handling security requirements may demand a wide variety of knowledge, given its nature.

Tondel et al. [94] still state requirements in general should be specific, measurable, appropriate, reasonable and traceable (such characteristics form the acronym SMART). Firesmith et al. [88] suggest classifying security requirements into categories, e.g. identification, integrity and privacy requirements [88]. Firesmith et al. [88] and Haley et al. [44] mention some examples of SMART security requirements:

- The system has to identify all users before allowing them to use its functionality (Identification Category);
- The system should not allow unauthorized users to access any type of information (Privacy Category);
- The system has to provide Personal Information only to members of a specific department (Privacy Category).

A list of requirements should include both functional and non-functional requirements. The functional ones are related to the functionality a software is supposed to provide. The non-functional ones are linked to quality, performance, portability and security requirements. Specifically speaking about the non-functional security requirements, nowadays it is critical to think about protecting a system since the very beginning of a project. The development team should deal with security requirements from the early days until the end of a project [59] [64].

Issues related to security should be treated since the very beginning of a software development project, as already mentioned. It is a hard task to build the correct security requirements, being equally difficult to trace and to test such type of requirement along the project lifetime cycle due to the non-functional nature of security requirements [94].

Security requirements are normally not at the top of the priorities for software development. There is a tendency of focusing on functional requirements, those directly linked to the core business. There is an exception when security is a part of basic standards or legal issues, so security requirements may be considered as functional [98] [64]. Firesmith et al. [88] also agree that security requirements are undervalued, adding the fact that software professionals in general are under trained to elicit security requirements [88].

Agile teams can treat security requirements in a poor way because security engineering is considered heavyweight and extremely hard to comprehend and to implement. Nevertheless, just by introducing some practices, it is possible to integrate non-functional requirements such as the security ones for instance [90].

Previous studies have already researched security practices considered agile compatible. By performing a survey of software security activities amongst software developers, it was possible to point out and also to suggest some security activities such as identifying security requirements, risk analysis and penetration tests [11] [8].

As previously mentioned, Microsoft has developed its own framework named as Security Development Lifecycle (SDL) [48], which has an extension of recommendations for integrating security to agile development [66] [65]. There have been other approaches trying to demonstrate how security features may be absorbed within agile development [10]. Section 4.2 will present some of the most known security agile approaches.

4.2 Security Agile Approaches

There are some proposals of methodologies to integrate security activities into agile development, such as Microsoft SDL for Agile [66] [65]. At the same time, there are a number of critics about the similarity of such approaches when compared with the traditional versions of security methodologies. They impose the same workload as the traditional ones [18]. Here it is important to restate that adequate security is not a consensus and each organization has its own levels of risk tolerance [2]. Therefore, security practices should be adjustable.

It is possible to state that agile teams are adapting software security in a way that fits on the organization's necessities. Besides that, more than 70 percent of the reported security vulnerabilities are located in the application layer [38]. It is clear that the main discussion should be on how to improve security within agile software development, instead of discussing whether security and agile are compatible [13].

Microsoft SDL for agile [66] [65] is a set of recommendations (framework) for implementing the Security Development Lifecycle (SDL) [48] within agile environments. SDL

for agile is based on the phases and security practices already proposed by SDL, adding a practice classification according to when it is performed along the agile project [66] [65].

The security practices recommended to be executed in every release are categorized as Every-Sprint Practices. The ones that should be performed along the project on a regular basis, with the possibility to be spread across many sprints, are classified as Bucket Practices. The last category is named One-Time Practices, including the mandatory security practices that are supposed to be ran only once, at the beginning of the agile project [66] [65].

Figure 4.1, Figure 4.2 and Figure 4.3 depict the SDL phases and practices within each phase. The practices are classified as recommended by SDL for agile. It is important to notice that the security practice number one is a pre-requirement, which means that core security training should be a given at the beginning of all agile projects. Excluding the training phase at the beginning and the final phase called response, there are fifteen security activities proposed by SDL for agile, as shown in Figure 4.1, Figure 4.2 and Figure 4.3.

Looking at the already mentioned figures it is possible to identify some similarities with the Waterfall model [82] [79], particularly observing the depicted phases. As SDL for agile is an adaptation of the former SDL [66] [65], there is a possibility that it has inherited some Waterfall model characteristics. Activity number one does not appear in the figures below for representing a given, which means it (security training) should happen always at the beginning.

2. REQUIREMENTS	3. DESIGN	4. IMPLEMENTATION	5. VERIFICATION	6. RELEASE
2. Establish Security Requirements	5. Establish Design Requirements	8. Use Approved Tools	11. Perform Dynamic Analysis	14. Create an Incident Response Plan
3. Create Quality Gates/Bug Bars	6. Perform Attack Surface Analysis/ Reduction	9. Deprecate Unsafe Functions	12. Perform Fuzz Testing	15. Conduct Final Security Review
4. Perform Security and Privacy Risk Assessments	7. Use Threat Modelling	10. Perform Static Analysis	13. Conduct Attack Surface Review	16. Certify Release and Archive

Figure 4.1 – SDL for Agile - Every-Sprint Security Practices [65]

2. REQUIREMENTS	3. DESIGN	4. IMPLEMENTATION	5. VERIFICATION	6. RELEASE
2. Establish Security Requirements	5. Establish Design Requirements	8. Use Approved Tools	11. Perform Dynamic Analysis	14. Create an Incident Response Plan
3. Create Quality Gates/Bug Bars	6. Perform Attack Surface Analysis/ Reduction	9. Deprecate Unsafe Functions	12. Perform Fuzz Testing	15. Conduct Final Security Review
4. Perform Security and Privacy Risk Assessments	7. Use Threat Modelling	10. Perform Static Analysis	13. Conduct Attack Surface Review	16. Certify Release and Archive

Figure 4.2 – SDL for Agile - Bucket Security Practices [65]

2. REQUIREMENTS	3. DESIGN	4. IMPLEMENTATION	5. VERIFICATION	6. RELEASE
2. Establish Security Requirements	5. Establish Design Requirements	8. Use Approved Tools	11. Perform Dynamic Analysis	14. Create an Incident Response Plan
3. Create Quality Gates/Bug Bars	6. Perform Attack Surface Analysis/ Reduction	9. Deprecate Unsafe Functions	12. Perform Fuzz Testing	15. Conduct Final Security Review
4. Perform Security and Privacy Risk Assessments	7. Use Threat Modelling	10. Perform Static Analysis	13. Conduct Attack Surface Review	16. Certify Release and Archive

Figure 4.3 – SDL for Agile – One-Time Security Practices [65]

Beyond the SDL for agile, Haley et al. [44] have recommended four steps for dealing with security requirements. Such steps represent an alternative set of practices, instead of using SDL for agile (for instance). The four steps are [44]:

- Identifying functional requirements;
- Identifying security goals (identification of assets, threats, management principles and business goals);
- Identifying security requirements;
- Verifying the system.

Boström et al. [23] consider specifically the agile development and suggest seven steps for dealing with security requirements [23]:

- Identification of critical assets;
- Formulation of abuser stories;
- Abuser story risk assessment;

- Abuser story and user story negotiation;
- Definition of security-related user stories;
- Definition of security-related coding standards;
- Abuser story – Countermeasure cross-checking.

Abuse Cases [58] is a parallel concept that has emerged around the same time agile has been presented. It is more related to the security area, standing for any harmful interaction between an actor and a system even the unintentional ones [58]. Abuse cases are the basis for creating the abuse stories, also known as security stories. They are similar to the user stories, which are used to define the requirements, but their focus is on the security requirements ones. The only way to write a good security story is by really knowing the abuse cases scenarios. The security stories will be the security features embedded in the software [58] [9].

Peeters [75] has first proposed abuser stories as an agile practice to deal with abuse cases. It represents an informal story on how an attacker abuses the system. Abuse cases are also recommended by McGraw [62] and also by Wyk et al. [96] in their "Touch-points" security model. It is not detailed though, how to implement it in real projects. Peeters [75] has proposed the abuser stories, claiming such kind of stories would represent a suitable extension for agile practices. Abuse stories help agile teams to point out ways attackers (abusers) could abuse the system [75].

As already discussed, Scrum uses the user stories artifact to define all the functionalities that the future software has to have. Security stories are comprehensive even for those who are not Information Technology (IT) professionals. Users can collaborate to define such stories and the development team has to transform user stories into software features [82]. Firesmith's et al. [88] states that security use cases would be introduced similarly to the regular use cases. Security use cases should represent only security functionalities, which means countermeasures to possible attacks. In other words, security use cases must contain the possible attack scenario, the actors and also the countermeasures to avoid such attack [88].

Use cases are used for specifying and documenting software requirements. They are considered an easy way of communicating requirements, being comprehensible for all involved roles such as business staff who are not familiar with technical expressions and models related to the Information Technology (IT) area [50] [31] [1] [86]. Use cases are compatible with the majority of the functional requirements, but there are some critics stating it might conduct to undervaluing the non-functional requirements such as the security ones [90].

Misuse case [5] [89], on the other hand, is a practice that follows the line drawn by abuse cases and security stories. A misuse case represents an extension of a use

case diagram, demonstrating both positive and negatives behaviors. The regular use cases represent the positive behaviors, things one desire a user to proceed in the system. By adding the negative behaviors to the diagram, it is summed all the things one does not want users to do, things that can cause harm to the system.

It is important to state that such harmful things can be either intentional or unintentional. The not wanted behavior can be used to clarify the security requirements [5] [89]. Despite not being an agile concept, misuse cases seem to be adaptive and maybe useful for the agile purposes.

Peterson [78] suggests the usage of use cases combined with misuse cases as the foundation for dealing with security requirements. A use case may represent positives countermeasures to a misuse case as well as a misuse case may represent a threat to a use case [90].

As a basis for misuse cases, the Unified Modeling Language (UML) has previously defined the concepts of user (actor) and use case [70]. The concepts of misuser and misuse case has extended the traditional UML model. A misuser is an actor that is able to initiate a misuse case (negative action). It does not matter whether this action is intentional or not.

A misuse case is a sequence of steps taken by some misuser when interacting with a system, which if allowed to be completed can cause damage to some stakeholder [90]. McDermott and Fox [59] have also taken the standard UML use case notation to express threats to the system, but in their approach abuse cases are separated into other models [59].

The following steps are recommended in order to elicit security requirements using misuse cases [88]:

1. Identifying critical assets in the system;
2. Defining security goals for each asset;
3. Identifying threats to each security goal;
4. Identify and analyze risks for the threats;
5. Defining security requirements for the threats.

The usage of misuse cases to represent security threats was already defended in other researches [25], including by using some of the concepts from the security methodology known as Open Web Application Security Project (OWASP) [74]. Figure 4.4 shows an example of a misuse case diagram.

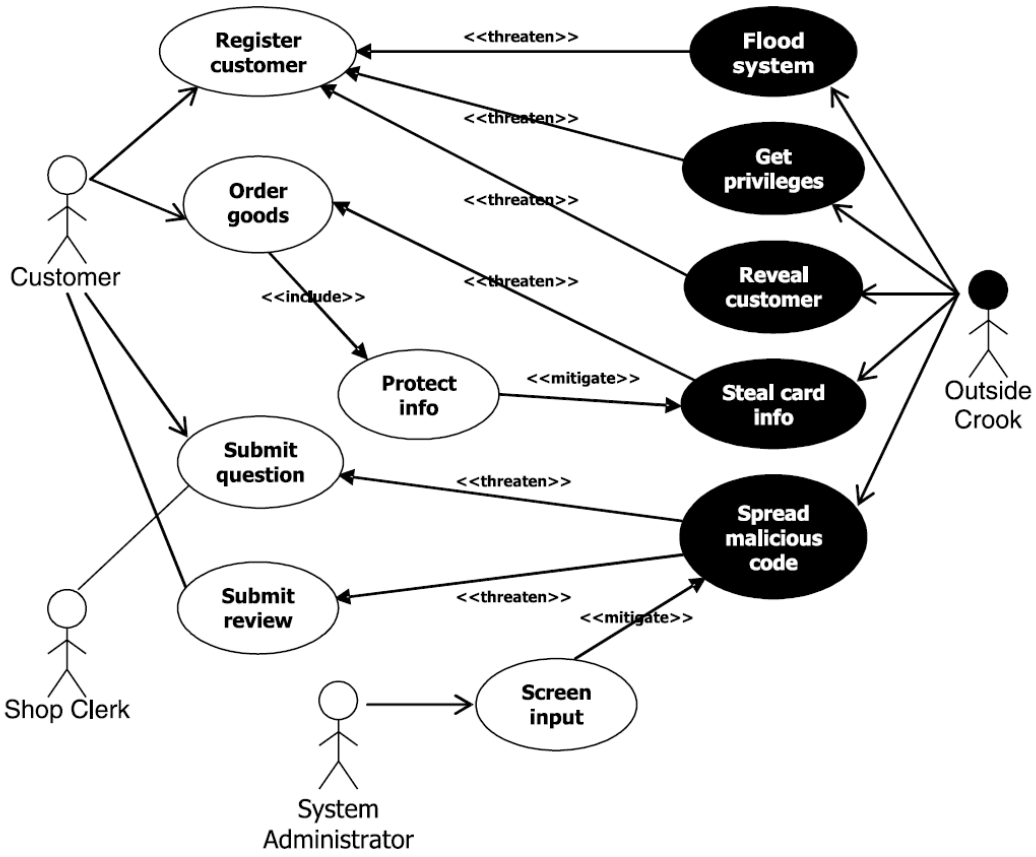

Figure 4.4 – Example of Use and Misuse Cases for an E-Store [5]

There are similarities between misuse cases and abuse cases concepts, but abuse cases were proposed to focus on security requirements and their relations with design and test [58] [59]. Misuse cases are complementary to abuse case because they focus on security requirements elicitation in context with other requirements. Use cases and abuse cases are not demonstrated in relationship, however misuse cases are linked to the use cases in a threat and mitigation relationship (see Figure 4.4) [5] [89].

Within this section it has been introduced some of the main approaches for coping with security requirements in agile projects. Section 4.3 will presents some software security models.

4.3 Security Frameworks

Security tests intend to validate a system specifically from the security weaknesses perspective. Some common vulnerabilities can be pointed as authentication, integrity, data protection and so on. As a security test, it is strongly dependent on the security requirements imposed by business rules, legal issues and other possible sources linked to the area. There are a variety of known security tests such as security assessment and penetration test, which can be applied according to the type of software and context [32] [41] [46].

The present section becomes especially relevant for this research considering the agile context and the Scrum framework context. Scrum states that each sprint should develop an amount of user stories [82] [71] [43] [3]. At the end of the sprint, in the sprint review, the user stories (software features) must match the done definition [82] [71] [43] [3]. Such comparison will define whether the features are really done or not (by the point of view of a user) [82] [71] [43] [3]. Related to the security requirements, security tests are the tools one intend to use in order to define whether a security requirement was fulfilled or not [17].

Not always security features are straight linked into the software requirements. Developers may not know the legislation or may not have the security expertise to cope with security issues. Furthermore, non-functional requirements are frequently ignored or undervalued by the development team. It is clearer for the team testing the functional requirements than testing the non-functional ones. That situation can lead to poor security tests given the fact that security tests are dependent on security requirements [17].

The current activities related to software test are far from enough [55] and agile testing is an increasing demand within the software development industry [32]. Continuous and agile test does not only apply to the agile projects. Companies are using it as a way to learn about any kind of software, using the customer feedback as a guide for the test activities. The agile tests, when proceeded in a continuous way, means incorporating changes as soon as they happen and keep testing the system in order to discover which of these changes have broken the system. For doing so, it is necessary constantly updating the test cases so that regression tests are feasible at any time [37] [67].

Continuous agile test is not a trivial task, requiring from the team members to work effectively as a team. It also demands a higher level of communication between the developers and the testers, who also has to learn how to act as an integrated team [57] [93]. In terms of security, testing can be described as testing of security requirements such as confidentiality, integrity, availability, authentication, authorization and non-repudiation requirements. It also includes the validation of the system's ability to stand against attacks (ability known as resiliency) [6] [36].

In order to proceed a successful security test, a variety of techniques is demanded. There is not a unique test able to cover all the security needs for testing. Although, many companies only adopt one security test type, e.g. the penetration test [7]. Agility privileges delivering value to the customers by feature implementation. In order to achieve such goal, not only the functional requirements are important.

One have to spend proper attention to the non-functional requirements. Testing non-functional requirements can be challenging, considering the multifunctional aspects of security requirements. Non-functional requirements tests are frequently undervalued due to many reasons from lack of experience to cost and time pressures [27]. There have been efforts to join software security with agile methodologies [22] [51] [72].

There are some agile challenges for secure software development, problems regarded to the security assurance in agile projects. Tests in general are insufficient to ensure the implementation of security requirements, being generally hard to be automated [72]. At this point, it is important to clarify some concepts about security tests and to set the theoretical bias this work is based on. The term security test can be used for referring security tests in general, but according to Hertzog [45] it is just one type of test.

It consists on the analysis performed by a security professional where other security test techniques are used in order to evaluate the security issues in general [45]. One of these security test techniques is the penetration test (Pentest) [69] [56], which is directed to exploit the already found security weaknesses in a specific way.

Risk Assessment is another technique that uses interviews and researches about the business strategic guidelines, legislation and everything related to the business area [45]. Figure 4.5 and Figure 4.6 show some aspects of the main security test methodologies.

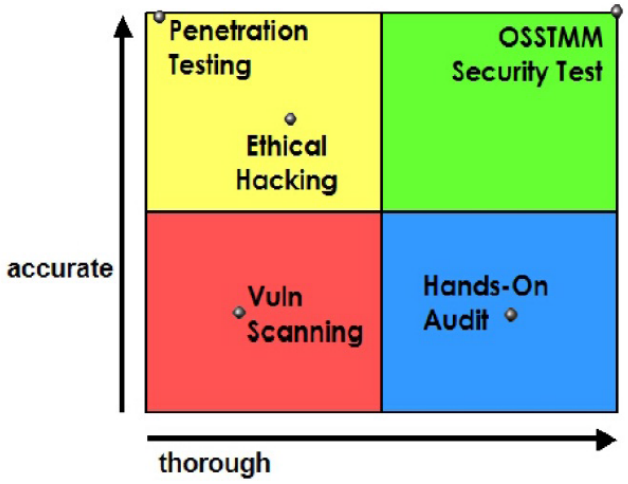

Figure 4.5 – Comparative amongst the Main Security Test Types [45]

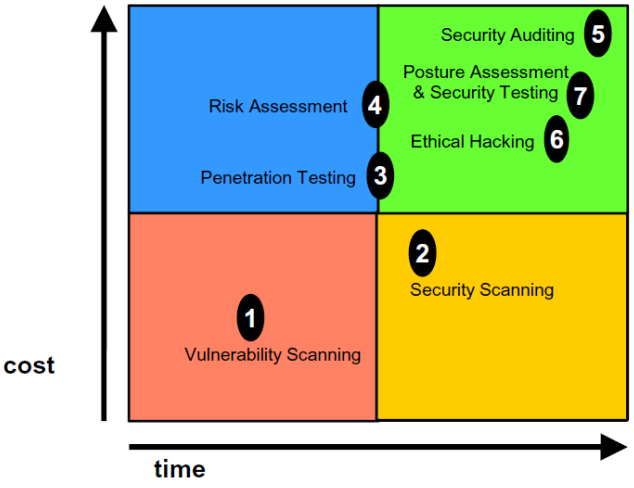

Figure 4.6 – Security Test Types Compared – Cost x Time Relation [45]

The security test techniques usage is a way of mitigate the risks related to the information security [102]. Test necessities comes from a variety of sources (e.g. business, customers, infrastructure changes, technology evolution or even from the usage of a system within the production environment [49] [32] [14]. DevOps has already captured this essence of integrating operation and development teams [49].

A systematic mapping study (SMS) performed by Bertoglio [19] [21] has identified the following security test methodologies:

- OSSTMM (Open Source Security Testing Methodology Manual) [45];
- ISSAF (Information Systems Security Assessment Framework) [42];
- PTES (Penetration Testing Execution Standard) [56];
- NIST (National Institute of Standards and Technology) Guidelines [80];
- OWASP Testing Guide [56] [47] [74].

Open Source Security Testing Methodology Manual (OSSTMM) is an international standard for security tests. It was built by the Institute for Security and Open Methodologies (ISECOM), being part of a larger scope that includes all spectra of the security environment (operational security environment including human factor, physical and wireless channels, telecommunication and data network) [45]. It is considered a consistent and complete model for security test [19] [21].

Information Systems Security Assessment Framework (ISSAF) is a framework for modelling internal information security requirements. It is composed by the planning (choosing strategies), preparing (defining environment, tools, team, contracts) and evaluation (execution and report) phases [42].

Penetration Testing Execution (PTES) consists on a set of instructions of how to execute security tests activities to evaluate the security level of an environment. It does not establish a rigid way related to penetration tests (pentest), being created by a community of analysts and security professionals [69] [56].

The National Institute of Standards and Technology (NIST) guidelines is a four-phases structure for security testing and assessment. The Planning phase is where the possible targets of attack are identified. In the Discovery phase the search for the system vulnerabilities is proceeded. Once the targets and vulnerabilities are pointed, the phase of Attack is executed and soon after that the Report phase presents the results of the whole analysis [80] [92].

The Open Web Application Security Project (OWASP) Testing Guide is a result of the studies of the OWASP community, which are meant to bring security into the software

development. They preach the usage of security tests as a way of bringing security awareness. In fact there are other projects such as Code Review Guide and Development Guide the Testing Guide is based on [73].

The Testing Guide presents three wide blocks: The Introductory one, where the prerequirements and scope of the tests are defined; The Intermediate one, where the OWASP Testing Framework is presented (tasks, tools and techniques related to each phase of the software development life cycle). Here the security tests are executed; The Conclusive one, where the vulnerabilities are described and the reports are built [73].

As a lightweight way of security methodology, the Cigital Touchpoints were presented in 2004 [63] [61]. It has been changing over the years, but its essence is composed by seven touchpoints: Code review, Architectural risk analysis, Penetration testing, Risk-based security tests, Abuse cases, Security requirements, and Security operations.

Tramonto is a penetration test (pentest) strategy introduced in 2016, based on the mainstream security methodologies as OSSTMM, ISSAF, PTES, NIST, OWASP. Tramonto is an attempt to create a pattern for pentest execution as well as trying to solve the lack of a specific pentest methodology [20]. Tramonto consists on five phases [20]:

1. Adequacy (definition of scope, target data, type of tests). Based on these choices, Tramonto will guide the tester through the work that has to be done by recommending activities;
2. Checking (checking the information demanded by the recommended activities checklist). Here the main goal is avoiding mistakes in the next phases;
3. Preparing (Planning, strategy and tools according to the previous phases);
4. Execution (pentest execution). Here Tramonto offers possible filters and ways according to the previous phases;
5. Finalizing (support on creating standardized reports). It provides the possibility of comparing the current test with future test executions and also helps on cleaning registries and traces of the test;
6. Evaluation (provides an evaluation offering alternatives based on the pentest results).

Table 4.1 puts together the information we have raised focusing on the security issues. It brings the intersection between security requirements and agile seen by the perspective of the security studies.

Table 4.1 – Security Perspective - Overview

Issue Description	Strategy to cope with
Non-functional security requirements undervalued	SDL for agile, Treating security since the very beginning [22] [62] [13] [18] [66] [87]
Dealing with Security requirements	Suggested steps (such as the Touchpoints), SDL for agile, Abuser Case/Story, Misuse Case [63] [90] [61] [11] [8] [18] [87]
Dealing with Security tests	Pentest, Automated/Agile tests, Risk Assessment, OSSTMM, ISSAF, PTES, NIST, OWASP, Tramonto [42] [45] [80] [11] [56] [8] [47] [69] [21] [74]

There is a lack of pragmatic suggestions on how to tackle the security requirements issues in real agile projects. Abuse stories is an example of suggestion, but without a description on how to implement it considering real agile projects.

Furthermore, the security strategies presented above do not bring explanation on how to use them in agile environments. That is the main gap we intend to fill on providing a pragmatic set of suggested practices for dealing with security requirements in agile projects.

Together with the suggested practices we will provide a detailed application case on how to execute such practices in real projects. Here we are filling another gap which is a lack of instructions on how to implement the recommendations raised by the literature.

As already mentioned, there is a lack of primary studies in the intersection between security requirements and agile areas. That is another gap we intend to fill on proceeding the present field study.

The theoretical basis of this work has been presented, as well as important concepts supported by influential authors within agile and security areas. Having the theoretical foundations set, Chapter 5 will proceed with the data analysis that has represented the basis of our recommended practices.

5. PROPOSED PRACTICES BASIS

The present chapter embraces the analysis of collected data, representing the basis of the proposed practices that will be shown on Chapter 6. Section 5.1 reinforces some key points on theory and also brings the main conclusions of an SMS released in a technical report format [87]. Section 5.2 approaches the analyzed documents as e-mails and Scrum artifacts such as product backlog and sprint backlog. Section 5.3 brings the proceeded interviews and their respective analysis. Next section starts the analysis by the reinforcement of some important theoretical basis.

5.1 Theoretical Analysis

It was already mentioned that the theoretical basis of this research has been used as the starting point for the proposed idea, especially the authors presented in the Section, 3.3, Section 4.1, and Section 4.2. A Systematic Mapping Study (SMS) was also proceeded before this work, being equally useful as a theoretic basis [87].

In the technical report called The Security Requirements in the Agile Software Development [87], it was pointed out some strategies to deal with security requirements in agile software development projects. It was mentioned three strategies named as Extending Agile Framework, Security Awareness and Security Stories [87].

The first one is related to add events or practices to an agile framework for dealing with security requirements. Such practices must be agile compatible and as non invasive as possible regarded to agility [87].

The second strategy is providing security knowledge to the agile team, aiming to make them capable to develop safer software. That includes training on how to build safe software and also includes spreading to the team the importance of security requirements [87].

The strategy called Security Stories embraces different concepts, such as Abuse Case, Abuser Story, Evil Story and Misuse Case. Such concepts are different but all of them intends to deal with undesirable behavior towards the system, especially the ones related to the security requirements [87].

Another important contribution provided by the SMS [87] is identifying the main agile frameworks in use nowadays. The most researched one was found to be Scrum, followed by Extreme Programming (XP) [87].

As Scrum is one of the most adopted agile frameworks [87], adding the fact that the unit of study also uses the same agile framework, it was plausible to focus this research in

the Scrum framework. Therefore, the suggested practices can be validated through Scrum projects in future researches.

Primary studies are scarce in the intersection between agility and security requirements areas [87]. The present research also intends to contribute by providing a primary study. The map shown in Figure 5.1 presents a resume of the main conclusions reached by the technical report (SMS).

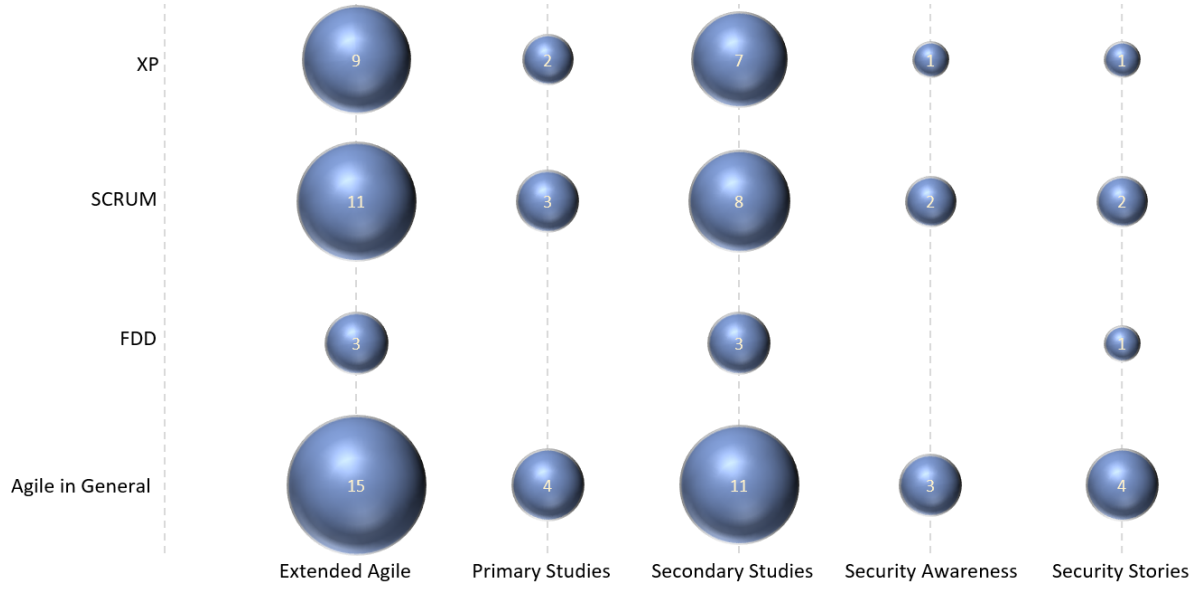

Figure 5.1 – Systematic Mapping Study Conclusions [87]

Microsoft SDL for agile also suggests security awareness through training as a practice [66] [65]. SDL for agile is a set of agile compatible practices, which means they alter the agile framework that is being used [66] [65].

Boström et al. suggest introducing what they call abuser stories for dealing with security requirements [23]. Abuse cases are the basis for creating the abuser stories [58].

A misuse case is another concept that follows the same line (abuse case and security stories) [5] [89]. Despite not being agile concepts, abuse case, misuse case and security story seem to be adaptive and may be useful for the agile purposes.

After proceeding with the SMS and setting the theoretical basis of this work, we have created some categories of gaps that should be tackled in order to improve the security level of software produced in agile projects. On the other hand, the SMS has already presented some strategies suggested by the authors for handling security issues in agile environments.

Table 5.1 presents the grouped gaps related to security requirements and agile frameworks, the ones that have been pointed by theory and secondary studies. It also links the gaps to the suggested strategies showed in the SMS, that ones said to be practices for coping with each category of gap.

Table 5.1 – Grouped Gaps

ID	Gap Category	Suggested Strategies
Gap-01	Lack of appropriate practice	Extended Agile Security stories SDL for agile
Gap-02	Lack of security knowledge	Security Awareness Security Training Security Experts in the team
Gap-03	Lack of primary studies	Proceed with primary studies aiming a safer agile development

These consulted authors represent the theoretical basis of this research. Beyond the theory, some real project documents have been analyzed in order to get a broaden view of the subject. Section 5.2 will describe the documents and their respective analysis.

5.2 Analysis of Agile Documents

We have chosen to analyze some documents to identify important points related to security requirements in real agile projects. As mentioned in Chapter 2, the content analysis and card sorting techniques were used in this research for analyzing the collected data.

Four Scrum Masters were interviewed as it will be presented in Section 5.3. The analyzed documents were produced by the projects in which these Scrum Masters are members. Such projects belong to the Health Care and Financial Services segments, as the scrum masters are allocated in these areas in the company.

The interviews has shown the organization has recently adopted the Scrum as its agile framework (one year ago). Therefore, the analyzed documents were mainly the Scrum artifacts. It is important to state that the documents used during a project as well as their context were explained by the scrum masters during the interviews. It is relevant to put the documents into their context when analyzing them, considering we are using the content analysis technique. Table 5.2 presents the list of documents inspected in this research.

The interviewed Scrum Masters were asked to provide examples of documents produced in their projects, setting such documents should be as recent as possible. They have basically provided user stories, technical stories and product backlogs. The researcher has requested the sprint backlogs and some exchanged emails for enriching the document analysis, providing more context and informal details. Grouped by similarity, Table 5.2 presents

these documents, all of them dating from March 2018 to January 2019.

Table 5.2 – Analyzed Documents

Category	Document/Artifact	Amount
Story	User Stories / Technical Stories	28
Backlog	Product Backlog / Sprint Backlog	20
Informal	E-mails	24
	TOTAL	72

Following the data analysis methodology adopted in this research, we have created some categories based on the document analysis. Table 5.3 shows the final categories and themes as a result of the content analysis technique. It will be discussed and detailed in the following paragraphs. The column category groups themes and each theme groups some document excerpts by their similarities.

The categories in Table 5.3 were identified based on the analysis of the documents and interviews, driven by the goals of this research. The themes represent a subcategory inside the main one, joining the text excerpts according to their similarities. Both categories and themes were reached with the help of card sorting technique [91], as mentioned on Chapter 2. It is important to state that both document and interview analysis have converged into the same themes and categories.

The excerpts represent a pieces of text which characterizes and reinforces a certain subject, being classified in themes inside the categories. After presenting the way data was explored and grouped, we will discuss the results, providing more information and linking the results with the theory.

Content analysis aims to reach understanding of a situation via interpretation and inferences applied on qualitative data [12]. In this sense, it is unfold some aspects over each category and theme, linking them with the goals of the present research.

It is important to reinforce that the proposed practices presented in Chapter 6 take into account the analysis provided in this section as one of its basis. The final result of the content analysis process over the agile documents is presented below (Table 5.3).

Table 5.3 – Document Analysis Categories

Category	Theme	Document Quotes
Information access	Non-legal requirement	"...user should not be allowed to..." "...they can not alter data at this point..." "...administrators set the access level..." "...it was cracked by an external user..."
	Legal requirement	"...for coping with BACEN..."; "...as ANVISA demands..." "...resolution 3490 asks for..." "...following normative ruling 245..."
Technical issue	Technology	"...framework provides security libraries..." "...making security protocol compatible..." "...it has to use a digital certificate..." "...to implement second authentication..."
	Test	"...test cases are outdated..." "...scenario was not tested..." "...review the script of test..." "...is it possible to automate tests on..."
Agile framework	Artifact	"...the technical story asks..." "...there is no user story for security..." "...refining the US for creating TS..." "...the chosen TS for this sprint are..."
	Practice	"...we need help on digital certificate..." "...GCAD is setting our version..." "...the PO can not help about security..." "...an expert would make it safer..."

We have found the sources which generate security requirements in the studied projects can be classified as legal or non-legal. The legal ones come mainly from public agencies, as it was possible to see in the documents. The excerpts always mention some public agency or a specific regulation provided and maintained by them.

In general, the legal requirements can be linked to the functional requirements for being part of the business rules of a system and for being generally known by the product owners. On the other hand, the non-legal requirements can be either functional or non-functional. Despite of that, all the non-functional requirements are also a non-legal one.

Both legal and non-legal themes were classified in the information access category, as we have identified all of them involve some kind of access to the data base. Information access is one important topic when speaking about security [102] [42].

Technical issue is another category which comprises technology and test themes. We have chosen to classify test as a technical issue given the tests are proceeded by the development team during the sprint time. The product owner also tests in the end of the sprint, but it is just for corroborating what has already been proved in the prior tests.

There seem to be a low level of automated tests, as shown by the text excerpts on this theme. Some developers are even demanding the automation of specific tests. In emails exchanged by the team members, it was possible to identify some level of dissatisfaction regarded to the manual process of testing. There are comments on the high costs some manual tests demand in terms of time.

Technology was also labeled in this category for having a straight link with technical issues, being the least known by product owners and final users. Here it was possible to identify issues related to technology are almost exclusively treated as non-functional, being detailed in the technical stories.

The artifact and practice themes were classified as agile framework category and comprises everything related to the agile framework, Scrum in this case. The text excerpts in this category have revealed the usage of an artifact named by the agile team members as technical story.

It was also visible some adopted practices as the GCAD team participation on the security requirements elicitation process. The agile framework category was specially useful for responding specific goals two and three (SG2 and SG3).

Considering an agile environment, the stories ultimately represent the software functionalities. They provide the perspective of a software user, being detailed by the team to be developed and translated into software features [82] [26] [52] [77]. Each scrum master have provided user stories from their projects.

Our main intention on analyzing stories is looking for traces about security requirements considering the perspective of the product owner and the development team. Such analysis has shown security requirements are always present, even though they do not seem to be clearly specified. Security is definitely a concerning, but it can be seen in a implicit way.

The most of the time security requirements get more explicit when the development team starts to refine the user stories. The teams use to create what they call technical stories. Here we can see the usage of a different artifact, as technical stories are not a formal artifact of Scrum.

All the agile teams that participate in this research use technical stories for detailing technical requirements regarded to performance, security, usability and etc. The Scrum Masters have approached the technical stories along the interviews, which will be presented in Section 5.3.

The analysis of the stories also includes the technical stories, as shown in Table 5.2. It was possible to identify the usage of technical stories for dealing with non-functional

requirements such as the security ones. We have identified requirements linked to performance, technology, usability and security inside the technical stories.

This kind of story is normally built by the team without the presence of the product owner. They take into account technical features, ones the users do not have appropriate knowledge about. This is the reason these stories are named as technical.

One of the projects deals with a health care system that is a module inside an Enterprise Resource Planning (ERP) system. The team is only responsible for adding new features to the module, having all the framework and layout ready to be used. That means they can not change anything related to the framework and the layout. Even the programming language is specific and linked to the ERP system.

In the case of this project, security requirements are less visible in the stories, even considering the technical ones. Here the stories focus almost exclusively on functional requirements. It can be corroborated by the interview of the respective scrum master, when this one highlights that the team receives the security issues ready from the framework team. That will be detailed in Section 5.3.

Even in this same project, it is possible the identify some implicit security requirements being treated, mainly in the refinement process when creating some technical stories. Other projects present security issues a little more explicit, especially after the refinement process.

Related to stories, it is interesting to note that nothing was straightly found related to the GDPR regulation or LGPD law. The documents show it is not a priority, at least by the time data was gathered.

All the analyzed stories had their definition of done, making it possible to test the features during the sprint. The tests are basically manual and except by one specific product, there is no formal script for automated tests. There is no official tool for automated tests, at least no one defined by the company. Some individuals use tools for trying to automate parts of the tests, but they are all isolated actions.

During the sprint review, the users were able to corroborate the completeness of the developed requirements. In general, the product owner sends an email for the team confirming the items considered done and the ones not considered as complete.

These emails are considered the formal acceptance, being possible to identify the definition of done was taken into account when accepting or refusing an item. The technical stories were absent in the analyzed emails, being tested and validated just by the development team.

The most of the identified security requirements were described in the technical stories. It is safe to state that the product owner and the stakeholders do not participate on security requirements issues, except when they are directly linked to the functional requirements. In this case, the security requirement is described in a user story.

Considering that the two segments (Health Care and Financial Services) are new on adopting Scrum framework, it is plausible to state that many practices are still being organized and set by the teams. There seem to be a certain degree of freedom about the way the teams test and the tools they use for doing that.

The product backlog is created having mainly functional requirements. As a general rule, it was verified that functional requirements normally come from user stories and non-functional requirements (such as the security ones) almost always come from technical stories.

In general the product owner seem not to understand about security issues when they are not straight linked to the business requirements (the business ones are normally in the legal requirements theme). In this scenario, security requirements tend to emerge after the refinement process proceeded by the development team.

Security requirements are present in the product backlog, but again they appear to be implicit. In the sprint backlog the security requirements get more explicit after user stories pass through the refinement process. The sprint backlogs are richer on technical stories when compared with the product backlog.

When the subject is dealing with security requirements during the sprint time, Section 5.3 will show the scrum masters do not feel that their teams have enough knowledge. Such situation has called our attention when putting together the documents and the interviews analysis. There seem to be a contrast, given the technical stories are built by the development team, but the scrum masters admit their teams do not have enough knowledge about the subject.

Before presenting the set of suggested practices, Scrum Masters were also listened to, intending to reach some insight combined with the theory and the document analysis. Section 5.3 will detail the proceeded interviews.

5.3 Interviews Analysis

Different from theory and document analysis, the interviews were another source of data for this research. The interviews were conducted within the studied company boundaries, considering the elected two segments named Health Care and Financial Services. The four interviewed represent the universe of Scrum Masters in both segments.

The identity of the studied company was kept in secrecy due to confidentiality. A tool was used for voice recording the interviews, with the prior permission of the interviewees. The audio files were transcribed, providing the basis for applying content analysis and card sorting techniques.

Table 5.4 demonstrates the interviewees profile and also relevant information about the conducted interviews. The participants identity was also preserved, therefore they were referred as I1, I2, I3 (Health Care segment) and I4 (Financial Services segment).

Table 5.4 – Interviewees Profile

ID - Age Range	Agile Experience	Interview Time
I1 - 30-40	1 year (1 year at the company)	36 mins 54 secs
I2 - 40-50	2 years (4 months at the company)	24 mins 41 secs
I3 - 30-40	1 year (1 year at the company)	29 mins 33 secs
I4 - 40-50	3 years (1 year at the company)	39 mins 16 secs

It is important to remember that the questions taken into account are described in Chapter 2. The intention at this level is getting the ideas from the interviewees linked to the subject of this research. They are not fixed questions, representing just a script to be followed during the interviews (as the adopted collect data procedure is the semi-structured interviews).

The answers of questions 1 to 3 are demonstrated in Table 5.4, being useful to set the scrum masters profile related to professional and agile experience. Question 4 is related to training in general regarded to agile. All the interviewees are certified as scrum masters and all of them also have participated on workshops, seminars and other agile training.

Questions 1 to 4 are relevant in order to characterize the scrum masters profile. It is possible to say that all of them are relatively new at agile practices. The most experienced one is I4, having 3 years of agile experience. Summarizing the interviews analysis, we have used the same themes and categories as the ones used in the document analysis. It is explained because the analyzed documents were provided by the interviewees and also refer to the same projects approached during the interviews. Based on the interviews transcriptions, the text excerpts were classified in the same themes and categories.

The themes and categories were already explained in Section 5.2. Therefore we will go straightforward to the interpretation and inference process, based in Table 5.5 as suggested by Bardin [12].

Table 5.5 – Interview Analysis Categories

Category	Theme	Interview Quotes
Information access	Non-legal requirement	""...We get security requirements ready..." "...the PO demands security issues..." "...when defining user profiles..." "...non-legal issues as authentication mode..."
	Legal requirement	"...public agencies can set security issues..."; "...ANVISA states some rules for..." "...BACEN demands information about..." "...there are so many public agencies..."
Technical issue	Technology	"...we don't feel having enough knowledge..." "...it demands help from experts..." "...framework provides many features..." "...it is provided by the framework..."
	Test	"...the developer could program the test for..." "...we need to be sure they are really safe..." "...We do not use tools for test automation..." "...lack of automation increases time..."
Agile framework	Artifact	"...security issues in technical stories..." "...during the refinement, technical stories..." "...there are no user stories for security..." "...The PO doesn't participate on TS..."
	Practice	"...passed to other team after committed..." "...GCAD will set the product..." "...GCAD should be involved since..." "...our source of knowledge is GCAD..."

We will present the answers of each remaining question in the following paragraphs. The text excerpts presented in Table 5.5 will be also detailed and analyzed.

It is important to discuss about the experience of both segments related to agile in order to put the analysis into context. Agile is something new for the two segments of the company (Health Care and Financial Services). In the Health Care segment, the three interviewed Scrum Masters have stated that their segment has adopted agile practices (Scrum) since last year. The Health Care segment has a little over a year of usage in Scrum framework in order to get compliance with the company, being one of the last segments to adopt Scrum.

On the other hand, the Financial Services segment has not adopted Scrum in a complete way. There is no formal scrum master at Financial Services segment although one professional acts as informal scrum master, specially when dealing with costumers that

work in an agile way. This is the reason there is only one informal scrum master interviewed at Financial Services segment, where just some agile practices were adopted and not the entire Scrum framework.

When asked why the Financial Services segment has not fully adopted Scrum, I4 has answered there is some degree of internal resistance. Despite of that, there are some planned actions for the second semester of 2019. Such actions will come from the Corporate Engineering Department, as they intend to bring Financial Services segment into the Scrum framework.

Giving the low level of agile experience of Health Care and Financial Services segments, all of the interviewees has shown to be opened-mind when speaking about new agile practices. They seem to be very receptive to test new practices and different ways of following the Scrum framework.

Question 5 asks about the security requirements understanding. In general, the interviewees tend to speak about the infrastructure team when explaining their concepts of the subject. Using the words of I1: "We get the security requirements ready from the infrastructure team". That team is internally known by the acronym GCAD, which stands for Register Management (it stands for *Gestão de Cadastros* in Portuguese). They are the responsible one for the product infrastructure and also for installing the products into the costumers environment.

In the same way, I2 has mentioned GCAD team as a source of knowledge linked to security issues. I3 agrees with I2, saying that they try to bring GCAD team together with the development team when they need to clarify questions about safe development. Diversely, I4 has said that they do not involve GCAD team into their development cycle. I4 follows saying that the development team members use to research for the demanded knowledge when it is needed.

Considering both segments, Health Care and Financial Services, there seem not to be a default recommendation for dealing with security requirements in agile teams. It seems to be treated in different ways according to the opinion and the style of the Scrum team.

I4 has talked about the release process regarded to a new product or even when they need to release a version to fix some problem. The interviewee follows saying that after committing the software code, GCAD team takes the control delivering, installing and setting the product version into the customer environment. Using the words of I4: "The product is passed to another team after being committed and that team will be the one who sets this product in the customer's environment".

I2 and I3 have mentioned a similar release process as I4 quoted. They have added the release process of a new product. According to them, it seems the agile process stops after the development, when the new product is officially released.

I4 has also approached the new products, quoting that agile should be applied to the product as a whole, not just for the development process. I4 thinks there should not be a support team, as it happens nowadays. According to I4, there should be a product team responsible for the development and also for the support of a product.

Here it is important to quote Scrum recommends the team should be complete regarding to the necessary knowledge to build the product [71] [43] [3]. That means the agile team should be able follow and support the whole cycle of a product [71] [43] [3]. As the interviewees can corroborate, the studied company has an apart team for installing and setting the products. It also has another team for supporting that same products.

Still discussing the answers of question 5, all of the interviewees have remarked they do not have the appropriate expertise connected to security issues. They have stated the same quotation about their development team. The interviewee I4 has even quoted they feel unsupported when the topic is security requirements.

Only I2 and I4 seem to link security requirements to the functional requirements, the ones straight linked to the business (explicit) software features. They have highlighted sometimes the product owner or even the business rules impose security requirements to a project. The interviewees I1 and I3 have commented about security issues as if they were only related to non-functional requirements.

All the four interviewed scrum masters have mentioned an artifact called technical story for handling security requirements. I3 have quoted that the development team uses to refine the user stories, creating the technical stories during the refinement process. I1 and I2 have mentioned a similar process. I4 added technical stories are not always created, but the most of the time they depict requirements that are not present in the product backlog.

The technical stories have also appeared during the document analysis. It is an external artifact, not being present in the original Scrum framework [71] [43] [3]. According to the scrum masters, the teams have adopted such artifact for detailing technical requirements. I1 has quoted the product owner does not have any kind of knowledge related to security issues. I1 follows stating the same happens with performance requirements and other technical issues. That is the reason the team creates the technical stories when refining the user stories, according to I1.

At this point, it is possible to state that the teams have used a different artifact out of the core agile framework. The ones provided by native Scrum seem not to be complete for coping with non-functional requirements, specifically speaking about the security ones.

Question 6 has approached the main sources which generate security requirements in agile software development projects. It seems to be a consensus that laws and regulations are the main sources of constraints in their projects. Even though not all of them have directly mentioned the security requirements, they seem to converge that laws and regulations impose ways to treat and deal with information and security issues.

The interviewee I2 has quoted how hard are the laws about patients data manipulation when approaching the health segment in Brazil. I1 has added in Brazil there are some legal and regulatory agencies such as ANVISA (National Agency for Sanitary Vigilance), ANS (National Agency for Supplementary Health) and even the CNS (National Council for Health). They prescribe rules for capturing, maintaining, manipulating and divulging patients information in general, as I1 has said.

I2 has quoted sometimes security constraints come from the product owner (PO). I4 has provided some examples of security issues defined by the PO, such as providing different levels of access for different users profile.

I4 has mentioned BACEN (Central Banking of Brazil) and FEBRABAN (Brazilian Banks Federation) as the main rulers when the subject is security in the financial services segment. According to I4, the regulation is tight mainly when it is related to information security. I4 has also quoted the variety of products offered by the Financial Services segment, stating that it makes even harder to trace security requirements for all these products.

There is an important point to be highlighted here, the fact that no one has even mentioned the Brazilian LGPD law or even the European GDPR regulation. Maybe it has happened because LGPD is something new in Brazil, but it is safe to state that today the main concerning inside these two segments are the already existing laws and regulatory agencies.

The interviewees have highlighted the amount of public agencies, laws and regulations over the health and financial segments in Brazil. It is also possible that the related laws and regulations are so tight that GDPR and LGPD could be seen as just another layer already included inside the existing set of regimentation. Given there are so many public agencies ruling over health and financial segments in Brazil, that can also explain the lack of concerning related to GDPR and LGPD.

When asked about the practices used nowadays for dealing with security requirements (question 7), I1 has said that most of the security practices in software development are adopted by the framework team. In fact that team delivers all libraries and patterns for coping with security, as I1 has declared.

I1 has added that they are in charge of a module inside an Enterprise Resource Planning (ERP) system. That ERP provides all the framework, layouts and even the programming language they have to adopt in their development (including the Interface Development Environment - IDE). That is why I1 believes the team has a low level of responsibility when the subject is security requirements. Again it is clear I1 thinks about security requirements as a non-functional one.

According to I1, the only security issues the team has to worry about are the functional ones raised by the product owner (PO). In other words, the development team only has to develop the security requirements linked to the functional requirements. It is interest-

ing to mention that there seem to be a contradiction here, as I1 was not able to link security requirements to the functional requirements when responding question 5.

Different from I1, the other scrum masters do not have a framework team behind their systems. Their teams develop software from zero, therefore they have to build both the functional and non-functional requirements including screen layouts and security features (as quoted by I3).

I2 has mentioned they try to involve the GCAD team in order to obtain some light at security issues. I3 has said the same, even though recognizing it is not a formal practice. I4 do not count on GCAD team when defining security requirements, but they discuss security issues sometimes, out of the development projects.

All the interviewed scrum masters were questioned about their suggested practices for dealing with security requirements in agile projects. Such question represents question 8 and focus on practices not taken nowadays, different from that practices in use.

I1 has said that they receive a ready framework, having to develop only the functional part of the system. They do not need to develop the layout, the visual elements or even the security libraries. I1 goes on highlighting it is hard thinking about some practice for handling security requirements in such context.

I2 has emphasized again the involvement of the infrastructure team (GCAD) since the beginning of the project. I2 also states that it should be a formal rule, even when defining the user stories. In the words of I2: "The GCAD team should be involved since the very beginning of the project, even in the specification phase". I3 agrees with I2, adding that nowadays the infrastructure team (GCAD) involvement occurs late in the project.

I3 states that the late involvement of the infrastructure team almost certainly will generate rework in the future, it happens many times according to I3. Following the same line, I3 also agrees that the infrastructure team should help the development team to refine the user stories and even to create some stories related to performance, usability and security.

The refinement process was something mentioned by all of the four interviewees. They have highlighted some requirements emerge from that process. I2 and I3 have pointed the most of the security and performance requirements are unveiled through this process.

I3 has raised they use the technical stories in order to complement the user stories. I4 also admits the development team uses technical stories for handling requirements the product owner is not familiar to.

I4 suggests the development team should be trained in security issues in order to broaden their knowledge regarded to the area. I4 also suggests introducing some sprints especially for dealing with security requirements. Using the words of I4: "We need to develop security features, but we also need to test them in the end of the process to be sure they are really safe".

I2 has added automated tests could help on dealing not only with security requirements, but also with all the requirements in general. In the words of I2: "We do not use tools for automated tests and that increases the time we need to spend on testing". I4 agrees with I2 mentioning there is no script for automating tests.

I4 said that Financial services segment uses an out-to-dated tool for test automation but it is used just for a few situations. Still about test automation, I4 states that "it would be better if the developer could program the test for another team member execute the automated test".

Question 9 was answered along the interviews while responding to the prior questions. The scrum masters needed to speak about process and documents in order to basis their comments on questions 5 to 8. The technical stories have emerged as one of the answers for question 9.

It was also possible to understand the refinement process where the technical stories are created. When asked about question 9, the scrum masters basically repeated some parts of the prior responses. Just a little new information was added at this point.

I4 has mentioned an automated test for a specific product. According to I4, "at the end of each sprint the automated test script is up to dated and executed over the newly delivered features". I4 thinks it should be done along the sprint time.

I2 and I3 have brought up the usage of planning poker as a way of estimating the demanded effort for each sprint. They have said their teams use a mobile app designed for planning poker, highlighting they are getting better on estimating efforts at each new sprint.

I1 has raised they have a integrated moment with the framework team at the beginning of each sprint. Normally they spend about 2 hours discussing the integration with another modules of the ERP module they are responsible to.

Before presenting the set of suggested practices, Section 5.4 will put together the main results of the analysis presented in the prior sections of this chapter. That will be useful to show how the suggested practices have emerged as well as how they are linked to the identified gaps.

5.4 Final Analysis

The Scrum framework suggests teams should be self-organized and self-managed [71] [43] [3]. Scrum also recommends the teams should have all the demanded knowledge for responding to the entire process, since the beginning of the development [71] [43] [3]. It does not happen in these two segments as we could identify different teams for installing and supporting the released product.

Focusing on the product being installed in the customer environment, there is a separated team who is responsible for proceeding that. It means the operational team (GCAD) is apart from the development process inside the agile teams. On the contrary way, DevOps recommends the integration between the development and the operational teams [49].

It seems the product development process adopted in both segments makes it harder for agile to flow along the product cycle. That also seems to affect security requirements, which are implicit in the product backlog. Maybe the development process could be adapted to better fit the agile principles and values [71] [43] [3] as well as providing better security requirements. During the interviews, we could notice when a new product is developed it is delivered to another team after being officially released by the product team. It was even clearer in the Financial Services segment.

An agile team responsible for each product would be closer to the agile philosophy [71] [43] [3]. As Scrum suggests, such team should take since the product conception passing through its development and following the product cycle.

We believe it is possible that such changes could lead both segments to rethink other practices inside the company. We also believe that it is exactly what agile provokes inside the companies, the possibility to redesign their processes and practices in order the reach better productivity [71] [43] [3].

One of the things we believe both segments will have to face in a near future is to reevaluate their product portfolio, which seems to offer many options as referred by the interviewees. We think such a variety increases the complexity on managing and providing agile teams for each product.

At last, before presenting the proposed practices, it is clear both segments are relatively new on adopting agile. We believe the company should provide either agile and security training for their agile team. Maybe an external consulting should be considered, mainly for agile and security issues.

The main intention of this section is to brief a general overview over the data analysis process. As we had different sources of data, it is important to put them in the same page linking them with the studied theory.

After presenting the theoretical basis and after analyzing the documents and the interviews, the following Section 6.1 will present the set of proposed practices derived from that process. The proposed practices considered together represent a new extended agile framework. It is important to restate that they have emerged from the reached conclusions obtained in the proceeded Systematic Mapping Study (SMS), the theoretical basis of this research and the data analysis process (document analysis and interviews analysis).

6. PROPOSED PRACTICES

In order to accomplish the main goal of this research, a set of practices were proposed for dealing with security requirements in agile development. Added to the Scrum framework, such practices will form an extended agile framework.

An extended agile framework is one of the main suggestions for handling security requirements in agile development [87]. One can modify an agile methodology just by adding practices to the original framework with the purpose of making agile development safer [18].

Agility has shown to be flexible even welcoming changes along the project time. It understands that change is a constant and not an exception. One of the agile values states it when highlighting that change response is better than following a tough plan. That brings freedom to the agile teams, including the Scrum ones, for deciding what are the practices they may take in order to keep themselves productive while delivering safe piece of software [15] [82] [71] [43] [3].

Security activities have already been recommended [66], but our main intention is proposing practices that are agile compatible and lightweight. In other words, we desire to interfere as minimum as possible in the agile flow by keeping the simplicity agile faces the development process as a whole (empirical process) [71] [43] [3]. Here we want to comply with the agile manifesto, which states that it values communication over process, working software over dense documentation, collaboration over contract and change response over following a plan [15].

As a criterion, we have stated the suggested practices have to be agile friendly, which means they have to be agile compatible (Scrum compatible in this case). It has already been shown that some security activities are considered agile compatible [90] [11] [8].

At this point, it is relevant to say that the set of suggested practices was based and also inspired on agile compatible practices suggested by the available literature (Sindre et al., Baca et al., Ayalew et al., just for mention some of them [90] [11] [8]). It has represented the starting point of the present research, providing insights for building some recommendation. The document and interview analysis have enriched the process providing more ideas and corroborating with other ideas already raised in the literature. Section 6.1 will bring the proposed practices.

6.1 Set of Proposed Practices

Inspired on the theoretical basis, on document analysis and also on the answers provided by the scrum masters, we have elaborated a set of agile practices for dealing with

security requirements in agile software development projects. It is relevant to highlight that the main intention is using an extended agile framework with the minimum interference on the activities flow, that flow proposed by the original agile framework.

It was already mentioned that Scrum is the chosen agile framework in this research. The reasons for such choice are both, the usage of Scrum by the studied company and also the large usage of Scrum by companies all over the world [51] [97] [82] [83].

We have chosen to introduce a new artifact named as "Misuser Story", being a suggestion for dealing with security requirements. That new artifact is the core of our proposed practices as a misuser story is supposed to enter in the agile flow, following the agile cycle. Section 6.1.1 is introducing the misuser story concept.

6.1.1 Misuser Story

We start the current section by rescuing the concept of a user story, which stands for a short and simple form of describing features from the users' point of view [82] [71] [43] [3]. In other words, it is a short story told by users about how they see the usage of certain functionality, a desired one to be developed.

It is important to show the user story concept before presenting the misuser story one for highlighting their similarities and contrasts in terms of perspective. Remembering that our main intention on bringing such a new artifact is improving the way agile copes with security requirements.

The theory about the subject has pointed to some practices for treating security requirements along the agile flow such as abuser stories, security stories and misuse case [87] [23] [5] [89]. Misuser story is a concept that represents a mixture of those mentioned above, in other words it is an adaptation.

Security stories, which embrace abuse story and evil story concepts, are considered an easy way of describing abuses a system may suffer [23] [87]. It is an attempt to foresee possible attacks as well as their related attackers [23] [87]. A misuser story has absorbed the same feature, adding the fact it is written in a specific syntax so that it can be tested along the sprints.

Misuser stories also have inherited some features of misuse cases [5] [90] [87] in the sense that both points to an undesirable behavior towards the system [5] [90] [87]. Different from misuse case, a misuser story has its embedded done definition, the scenario, the attacker profile and also the non-desired action. It is drafted for having its tests failed.

A misuser story points to an undesirable behavior towards the system, it does not matter whether it is intentional or unintentional. It has the same structure as a user story

already adopted by Scrum, but represents something one does not want to happen. As it is based on stories, it represents a familiar concept to agile practitioners.

Related to desirable and undesirable behavior towards a system, we can trace a parallel with the liveness and safety proprieties of a system. Such proprieties were first proposed by Lamport [54] [76] in order to prove the correctness of a system [54] [76].

A misuser story can be linked to the safety proprieties of a system, as it states something that will not happen [54] [76]. On the other hand, the user stories can be related to the liveness proprieties, since they describe something that must happen [54] [76].

Misuser story represents a short and simple form of describing features or actions a misuser should not perform, it does not matter whether the misuser is an internal, external, well intended or a bad intended one. A misuser story tells us about something one is not allowed to proceed and such actor is called a misuser. Putting it in different words, a misuser story is a short misuse story told by a misuser perspective.

A misuser can be an external person trying to invade some organization system or it can be an internal one searching for information they should not have access. A misuser can also be a bad intended one which means they really want to perform some harmful action against the system, but they can also be a regular user of a system meaning they do not intend to execute anything wrong or potentially dangerous.

A misuser story also has its done definition and also has to be tested along the sprint and corroborated by the product owner in the sprint review. The difference is when one tests a user story it is expected a successful test. That means the test has to execute (complete) the story defined by the user (it has to meet the done definition). On the other hand, when testing a misuser story, it is expected that the test fails. The test has to fail because a misuser may not be successful on its dangerous behavior, even though it is not intentional. Again, a misuser story represents a non-desirable behavior (it has not to meet the done definition).

An example of a user story will be presented as well as a fictional misuser story in order to elicit the differences between both of them. We believe the misuser story is a familiar concept for the Scrum practitioners, just changing the focus of a story for lightening security requirements or even other technical issues.

The adopted scenario for building such examples of user and misuser stories stands for a banking account website. It allows the bank customers to access their banking account data, consult their banking statement and check their balance.

Table 6.1 shows an example of a fictional user story, its actor (a user) and also its respective acceptance criteria (done definition). It is important to show a regular user story at this point so that we can highlight the contrasts between the two types of stories by presenting another example of a misuser story.

A misuser story example is presented in Table 6.2, considering the same context already mentioned. It also brings its done definition considering a misuser, which is the actor of such stories.

Table 6.1 – User Story Example

User Story	Acceptance Criteria
As a Customer, I want to access my banking account statement so that I can follow my debits and credits and also check my updated balance	Ensure the Customers are able to: • Log in the banking account; • Find the banking statement option; • Choose the consult period; • See the debits and credits; • See the updated balance.

Table 6.2 – Misuser Story Example

Misuser Story	Acceptance Criteria
As a Misuser, I want to see the last data typed into the agency, account and password fields so that I can access banking accounts different from mine.	The Misuser is able to: • Get data typed into any field; • Store any kind of data when typing; • Access with wrong credentials; • Insert wrong credentials more than 3 times; • See the characters typed into the password field.

As it is possible to see at the example above, there is no need to provide more details about the actor in a misuser story. The active subjects in the stories are always misusers, independently on what they do or where they are located. The stories may represent any potential harmful behavior, being it intentional or not.

The done definition points to the actions that can cause damage to the system or even threat its data consistency. Having settled the new artifact (misuser story), it has to be introduced in the Scrum flow. As a new rule, it is stated that the product backlog has to have

both, user stories and misuser stories. The same is imposed to the sprint backlog, which must have both kinds of stories.

At this section we have introduced the concept of misuser story as a new artifact for dealing with security requirements. Section 6.1.2 will show how that new artifact is introduced in the agile flow of Scrum.

6.1.2 Misuser Story in the Agile Cycle

After presenting the new artifact concept and its main implications, it is time to show how that misuser stories should flow through the agile cycle. At this section we show when the misuser stories should be conceived, how they should be written, who should write them and the way they should follow along the Scrum sprints.

As a general rule, misuser stories must be created by the development team together with the product owner and the operational team. The operational team is the one who is responsible for the infrastructure, the one who installs and set up the system in the costumers' environment.

The idea of integrating the teams comes as an attempting to mitigate the conflict between the development team and the operational team. It is something already suggested by DevOps, as the operations may also be a source of security requirements [49]. The main responsible for creating the misuser stories is the operational team, even though they should involve all the agile team along the process.

The product owner is able to point the security requirements linked to the functional requirements (business rules). The operational team is capable to show the security requirements regarded to the non-functional requirements, which sometimes can be ignored by the development team. This is the argument that basis the rule of creating the misuser stories mixing development team, product owner and operational team.

The functional and non-functional requirements were already discussed in the Chapter 4 [98] [64]. The trend to undervalue or even ignore non-functional requirements was also approached in the same chapter [17].

Considering such issue, the idea of putting together the product owner, the development team and the operational team has emerged. We believe it is possible to make agile safer just by including the operational team on building and testing the misuser stories.

The literature has pointed it is normally a tough task to automate the security requirements tests [72]. Such difficult task was also a point raised by the interviewed scrum masters. In order to tackle that problem, we suggest adopting the Gherkin syntax [33] for writing both types of stories, the user stories and the misuser ones.

Gherkin is an open source tool for automating tests based on business-readable specifications. It is available on any modern development stack [33]. It is possible to automate the tests that will be proceeded in the sprint review when one uses the full Gherkin procedures, even though we believe only by adopting Gherkin syntax is enough to improve security level in an agile development project.

It is important to state that the Gherkin syntax [33] can be used to write both the user stories and also the misuser stories (that is our recommendation). Tables 6.3 presents a similar example as demonstrated on table 6.2, but now written in the Gherkin syntax.

Table 6.3 – Gherkin - Misuser Story Example

Gherkin Syntax
• Background: Given an account type named "Banking Account" And an internet banking application named "Agile Bank" And a non-customer named "Misuser"
• Scenario: Misuser tries to get data typed into the credentials fields Given another user has accessed their banking account previously in this station When I try to get data typed into Credentials fields Then I should be able to get the data I want
• Scenario: Misuser tries to store data typed for other users Given another user will access their banking account in the same station When I try to store the typed data Then I should be able to store and recover data
• Scenario: Misuser tries to access a banking account different from his own Given I provide any wrong credential When I try to access a banking account Then I should be able to try again indefinitely
• Scenario: Misuser tries to see data typed into a password field Given another user has accessed their banking account previously in this station When I try to see the typed password Then I should be able to see the password

As it happens with the user stories, the misuser stories must be tested in the sprint review by the development team, the product owner and the operational team. They are also

tested against their done definition, but the tool used to test the misuser stories must be the security tests.

Security tests as the Pentest [69] [56] are able to test misuser stories, especially the ones related to infrastructure and non-functional requirements. The security tests must be applied by the operational team.

We have chosen to introduce the usage of the Tramonto Framework [20] in order to support the security tests executed by the operational team. As a security test framework, Tramonto will guide and help the operational team on proceeding the security tests, independently on the security test strategy they choose to follow [20].

Figure 6.1 shows the proposed set of practices in perspective. It provides an overview of the extended Scrum framework described until now. Figure 6.2 shows a magnified image on the sprint review event.

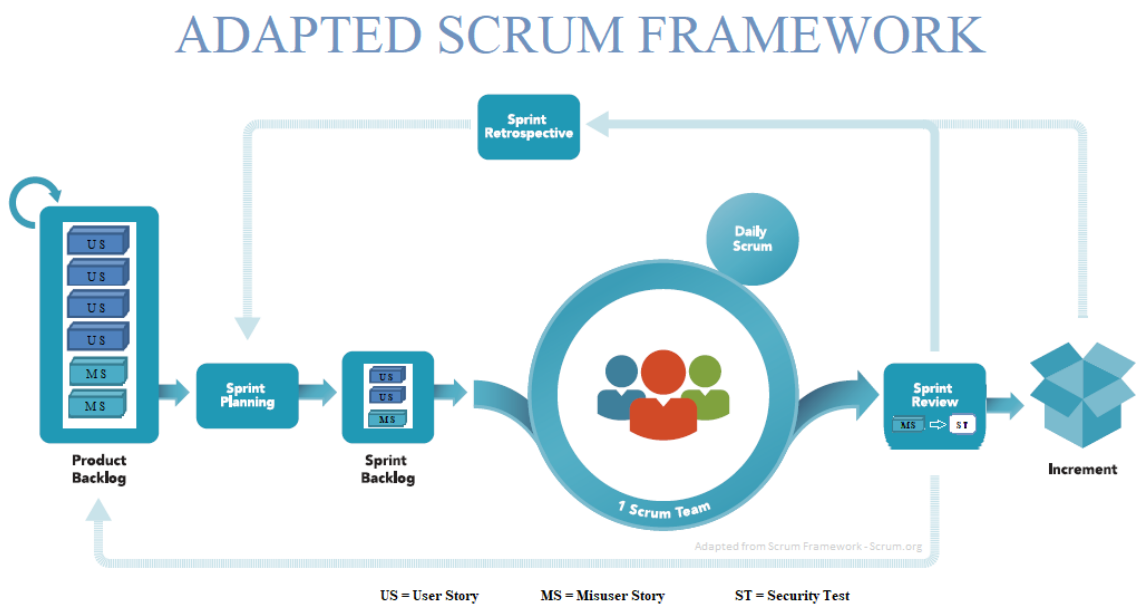

Figure 6.1 – Adapted Scrum Framework - Adapted from SCRUM.ORG [71]

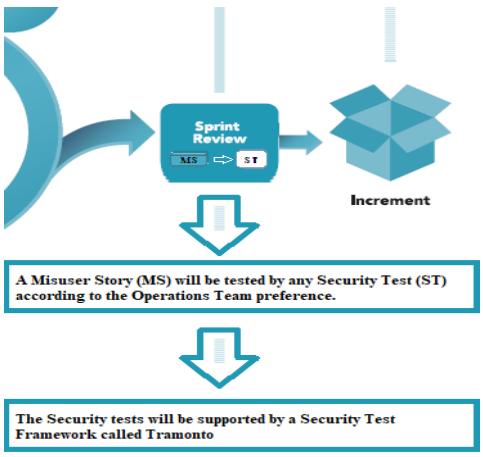

Figure 6.2 – Sprint Review - Magnified View - Adapted from SCRUM.ORG [71]

Figure 6.2 brings a magnified view on the sprint review event presented in the figure above (6.1). It explains the idea of using security tests for testing misuser stories.

The agile team is free for choosing the security test strategy they prefer. It is suggested the usage of Tramonto for supporting the security tests and providing a default output for the results.

This chapter has presented the set of suggested practices for coping with security requirements in agile software development. It is important to remember the practices were inspired on the theoretical basis of this research, the interviews and the agile document analysis. Table 6.4 brings a summary related to the proposed practices.

Table 6.4 – Summary - Set of Proposed Practices

Type	Name	Description
Actor	Misuser	The user who can cause damage (intentional or not)
Artifact	Misuser Story (MS)	Story for dealing with security requirements
Practice	Backlog Content	MS should be present in both Product and Sprint Backlog
Practice	Operational Team	The Operational Team should be an active part in the agile team, defining the MS
Practice	Misuser Story Test	MS should be tested in the Sprint Review through some security test strategy
Practice	Misuser Story Tester	The Operational Team should test the MS along the sprints and also in the Sprint Review
Tool	Tramonto Framework	The security test should be supported by the Tramonto Framework
Tool	Gherkin Syntax	The stories should be written following the Gherkin Syntax

The just introduced set of suggested practices needs to be validated in real projects, following the next natural step of an exploratory research. Section 6.2 will present an application case which can be used to apply the suggested set of practices in real situations.

6.2 Application Case

This section intends to be a model which can be used to apply the suggested set of practices in real situations. Therefore, we will pass through all the regular cycle of Scrum as showed in Figure 3.1.

At each step of the regular Scrum, we will present how the set of suggested practices should be introduced and conducted by the agile team, including the development team, scrum master, product owner and operational team.

By presenting that, we believe to be providing a how to model on approaching the regular cycle of Scrum as presented in Figure 3.1 to the extended framework of Scrum as proposed in Figure 6.1.

We will follow the Figure 3.1 sequence on presenting each artifact, event and how each role should act in this cycle. The starting point is the new artifact called misuser story, which must be present in product backlog as suggested by the set of practices.

Section 6.2.1 presents a model on how to build misuser stories as well as who would build them. Section 6.2.2 will present how the product backlog and the sprint backlog should be treated in terms of misuser stories. Section 6.2.4 and Section 6.2.5 will show how the suggested practices should be approached in the sprint review and the sprint retrospective.

Our suggestion as a natural sequence of the present research is proceeding a new research using the action research strategy. In this case the researcher would interfere in the agile teams by changing the way they work and also by providing training on how to adopt the proposed practices. The practices would be tested during a predefined amount of sprints and the researcher would coach the team along all the process. The following sections intend to support the future action research.

6.2.1 Misuser Story and User Story

Miuser stories represent the undesirable behavior of a misuser against the system. As it is undesirable, it will depict something the misuser may not proceed and that means it must fail when tested. The misuser must always be the main actor of a miuser story, which means it is not needed to detail the misuser actor profile.

Such stories must be written in the Gherkin syntax, as it makes possible the automated tests based on business-readable specifications [33]. Even if the company does not use automated tests or if it does not intend to take Gherkin as its automated test tool, Gherkin syntax is easy to understand and makes the path ready for the next improvement on automating tests.

The user stories also must be written using Gherkin syntax in order to make them easier to understand and also ready to have their tests automated. The intention is to implement a default pattern for the stories (user and misuser).

Table 6.5 and Table 6.6 provide a model on how user and misuser stories must be written using the Gherkin syntax. It is possible to see that the definition of done is embedded in each scenario.

Table 6.5 – User Story Model

Story example - Customer Base Access - Gherkin Syntax
- Feature: Different level of access according to the profile; Except administrators, who can access and alter all fields. - Background: Given a default profile named "Administrators"; And other profile named "Any-Other"; And a system named "Customer Base"; And an "Administrators" user named "Alex"; And an "Any-Other" user named "Peter". - Scenario: Alex access the "Customer Base"; Given I belong to "Administrators" profile; When I try to alter any available field; Then I should see "The register was successfully saved". - Scenario: Peter access the "Customer Base"; Given I belong to "Any-Other" profile; When I try to alter any available field; Then I should see "You do not have permission to alter this field". Scenario: Peter access the "Customer Base"; Given I belong to "Any-Other" profile; And "Any-Other" profile is set to hide the field "Phone Number"; When I try to find the field "Phone Number"; Then I should not find the filed "Phone Number" in the screen.

Table 6.5 provides an example of a story built to be successfully tested, which means it has to execute when tested. Table 6.6 shows a misuser story example, being expected to fail when tested as it represents an undesirable behavior.

It is important to reinforce that negative behaviors should be written in an affirmative way, as if we wanted them to succeed. It will make possible the future failure of the test, as expected. As the done definition is stated within the scenarios, they will be the basis to corroborate the failure of the tests and that means the security requirement was well developed and implemented.

Table 6.6 – Misuser Story Model

Story example - Getting someone's password - Gherkin Syntax
- Feature: User password stored in the database is cracked. - Background: Given an actor with a user named "Alex"; And a system named "Customer Base"; And Alex encrypted password stored as "XyzKb89!"; And Alex non-encrypted password being "Alex2@19"; And another actor named "Misuser"; - Scenario: "Misuser" tries to crack "Alex" password; Given I am inside the network; When I try crack Alex password cryptography through brute force; Then I am able to get the non-encrypted password "Alex2@19"; - Scenario: Misuser tries to get the "Alex" password; Given I am inside the network; And I have a tool installed in some workstation; And that tool is watching what is being typed on the keyboard; When I try to get the "Alex" typed password; Then I am able to get the typed password "Alex2@19". - Scenario: Misuser tries to alter the "Alex" password; Given I am inside the network; And I could reach the database layer; When I try to change "Alex" password out of the system; Then I am able to store a new password I can decrypt.

It is possible the most of the functional requirements will be treated through user stories, even though there might be misuser stories for both functional and non-functional requirements.

The non-functional requirements related to security are the most relevant here, for the purpose of this research. Misuser stories must focus on things the product owner does not have enough knowledge about, specially in terms of security.

User stories could be built by the product owner, but misuser stories must be written by the development team together with the operational team. Both are mandatory roles on

building misuser stories, as well as the product owner. That would also act as security training, spreading security awareness to the agile team.

The operational team is the one responsible for deploy, install, set and maintain the product in the production environment. It is the team who knows about the product framework in terms of security infrastructure. Therefore, they have appropriate background for helping on security requirements, specially the non-functional ones.

Table 6.7 summarizes the rules applied to the user stories and the misuser stories.

Table 6.7 – Rules for Stories

Story Type **Who builds**	**Syntax** **Test tool**	**Who tests** **Test result**
User Agile team	Gherkin Free choice	Agile team Success
Misuser Agile and Operational teams	Gherkin Security test via Tramonto	Operational team Fail

It is important to emphasizes that both user and misuser stories must be present in the product backlog. That determines misuser stories must be written in parallel with user stories so that either user and misuser stories could be part of the product backlog.

Having set the rules for building user stories and misuser stories, Section 6.2.2 presents how both artifacts must appear in the product backlog. The product owner provides all the user stories for a product backlog. The misuser stories are provided by the operational team together with the development team and the product owner.

6.2.2 Product Backlog and Sprint Backlog

The product backlog consists on all the user stories a product should have. Considering the suggested practices, a product backlog must have both user stories and the new artifact called misuser stories.

As misuser stories deals mainly with security requirements linked to non-functional requirements, it can happen many misuser stories are the same, for different projects. In this case the ones which are in common must be only instantiated in the new product backlog.

That is what we expect to happen as the usage of misuser stories goes on and the agile teams increase their maturity level on using the new artifact. We believe it will be naturally emerged a set of default misuser stories which can be used in other projects.

There are no suggested percentage of misuser stories, related to the total amount of stories a product backlog have. Despite of that, there must be enough of them in order

to bring into the light the security requirements hidden by technical or infrastructure issues. That is why the operational team must be responsible for building the misuser stories.

There is no need to adopt the technical stories, mentioned by the studied teams. The misuser stories are able to replace the technical stories embracing even more items than these prior ones.

Considering the studied company, the technical stories used to emerge in the refinement process, before picking the user stories for the sprint backlog. Now the refinement process must focus on decomposing both types of stories (user and misuser).

As the misuser stories were previously written in parallel with the user stories, when picking the stories from the product backlog to the sprint backlog, there must be both user and misuser stories in all of the sprint backlogs along the project.

The development team must make it visible for the product owner about the misuser stories they are dealing with in each sprint. Therefore, it must be negotiated between the development team and the product owner what misuser stories will be taken into account in the next sprint.

The Product owner must be able to understand the importance of each misuser story. Considering the product backlog and the sprint backlog, the most important issue is having misuser stories in both all over the project time. Following the flow of scrum, Section 6.2.3 approaches the procedures during the sprint planning and also during the sprint time.

6.2.3 Sprint Planning and Sprint time

During the sprint planning, misuser stories must be treated in the same way as the user stories. They should be estimated and prioritized as a regular user story.

The same happens during the sprint time, when the development team must implement both stories as if they were regular user stories. The development team may use security tests supported by Tramonto framework along the sprint time for testing the misuser stories.

The daily scrum may be proceeded in the same way it is regularly done, adding the misuser stories in the event agenda. Here it is important to emphasizes the integration amongst the development team, the product owner and the operational team.

It would be better having all the required profiles into the development team, including the one present in the operational team. In other words, we recommend having at least one operational team member as a member of the development team.

Even working with different teams, development and operational, it is possible to implement the set of proposed practices just by making sure there are integration between both teams. That means they have to work together in a close professional relation all over

the project time, participating on the Scrum events such as building the product backlog, sprint planning, daily scrum, sprint review and sprint retrospective.

After the sprint, it is time to verify the delivered piece of software. At this time, the product owner have to accept or deny the developed features according to their done definition (present in the stories). Section 6.2.4 will approach the sprint review and the security test.

6.2.4 Sprint Review

At the sprint review event the delivered piece of software must be tested and accepted by the product owner and also by the users. The stories have to be tested, the final test before releasing the new features.

The user stories can be tested using any kind of manual or automated test. We believe Gherking would make the tests easier to be automated. The misuser stories must be tested using security tests supported by Tramonto.

Gherking may help here again by providing structured misuser stories which can have their tests automated at any time. In the sprint review, it is mandatory that misuser stories have to be tested by the operational team. We suggest automated tests as we believe it may reduce the time invested on tests execution. Despite of that, the present set of practices can be adopted even considering a scenario where tests are entirely manual.

The agile team should be present or represented during the misuser story tests, including the development team and the product owner. That is another opportunity to spread security awareness to the whole agile team. We believe this moment will act as security training, increasing the level of knowledge related to the subject. Following the Scrum cycle, Section 6.2.5 takes into account the sprint retrospective event.

6.2.5 Sprint Retrospective

The sprint retrospective is time for discussing what has been done in the prior sprint as well as how they have been executed. It is an important time for making improvements considering the continuous improvement cycle proposed by Scrum.

Considering the misuser stories, it is time for discussing them and trying to create a set of reusable ones. They may be improved, rewritten or even altered for better usage in future sprints or projects.

It can be discussed how to automate the misuser stories tests or even making the already existing scripts better for the next usage. The sprint review should be ran the same

way it is regularly done adding the discussion about misuser stories, how to make them better and mainly how to make them reusable.

Beyond the Scrum events, it is important to set the role each actor has related to the proposed practices. Section 6.2.6 presents a brief comment on involvement of each Scrum actor.

6.2.6 Scrum Roles

We have been discussing the set of proposed practices while presenting the action of the Scrum roles. It is important to state that the operational team is now an active role in the agile team.

The regular agile team is already formed by the development team, the product owner and the scrum master. In the suggested practices it is added another role making the operational team as a member of the agile team.

The operational team is responsible for creating the misuser stories, for testing them in the sprint review and also for supporting the development team along the sprints. They provide security training and security advise.

The development team must include the misuser stories in their daily routine as if they were a regular user story. They must develop the misuser stories and also test them before delivering to the final test in the sprint review.

The product owners also must include the misuser stories in their routine. Their major responsibility keeps being over the user stories, but they have to participate on the misuser stories building as well as their tests.

Considering the set of proposed practices, the scrum masters must be the ones who deeply know about how to implement the misuser stories. They also need to know the usage of the required tools and techniques, such as Gherkin syntax and Tramonto framework.

In the regular scrum, the scrum master role is the one who provides agile knowledge and techniques for maintaining the team at their highest level of production. In other words, the scrum master is the agile guardian who keeps the agility flowing during the sprints.

As the guardian of Scrum and considering the proposed practices, the scrum master is now the guardian of the extended Scrum framework. The scrum master must train and follow the team on the usage of misuser stories, Gherking syntax and Tramonto framework.

The set of suggested practices and the application case were presented to the interviewed scrum masters in order to get their first impressions on how to implement that in real projects. Section 6.3 will approach the opinion of the interviewees related to the subject.

6.3 Final Considerations

The interviewed Scrum Masters were asked to provide their opinion about the suggested set of practices as well as the challenges they foresee related to the adoption of such practices in real projects. The next natural step after concluding this study is to apply the suggested practices into real projects and to evaluate the results after some sprints.

I1 has shown to be a little skeptic about some practices, mentioning it is hard to see their adherence in an environment where all the framework is already set by another department. Using the word of I1: "...As we receive a lot of things ready and defined from another team, that new practices should be introduced to this other team first".

Despite this opinion, I1 believes the misuser stories could be adopted independently from the other teams. I1 has shown to be curious on testing the artifact in real projects.

I4 has also focused on the misuser stories concept. According to I4, the misuser stories could be compared to the technical stories. I4 believes technical stories could be easier to understand when compared to the misuser ones just because they spotlight the desired result, instead of the undesired one.

I2 and I3 disagree with I4, stating that sometimes it is good to know what is not allowed or undesirable. According to I3, knowing what should not be done is as important as knowing what should be done. I2 adds to I3, mentioning maybe misuser stories would be a more complete way of writing the technical stories.

I3 has pointed the security tests on the sprint review, quoting that it should be proceeded all over the sprint and not only in the sprint review. In the I3 words: "It could be late, I mean in the sprint review, and that is not what agile preaches". I3 agrees the security tests could be automated and a script could be followed as suggested but it should be applied along the sprint and also in the sprint review.

I4 agrees with I3 adding that the security tests could be corroborated in the sprint review by the product owner, but security tests should be taken along the sprint time. I2 goes in the same line as I4, stating it would be good having a script to be followed related to security tests during the sprint time.

I1 believes there should not be a rule related to the amount of user stories and misuser stories in the sprint backlog, as suggested in the set of practices. In the words of I1: "The development team should be free to decide on what stories they will pick, independent on what kind of stories they are".

Overall, I1 thinks it would be good testing the set of practices, but pointing that it would make things more bureaucratic. I3 believes the set of proposed practices should be tested along a few sprints for entering in the continuous improvement process.

I4 adds the set of practices should be tested, excluding Gherkin model and Tramonto framework. In the I4 point of view: "The teams may be free for choosing the tools and methodologies they think better fits".

I2 would like to see the whole set of proposed practices running in a few sprints, but stating that it would demand prior training on misuser stories, Tramonto and Gherkin usage. I2 thinks it would request a preparing time before effectively applying the set of practices.

After presenting the data analysis, the set of proposed practices, the application case and also the first impressions of the scrum masters, Chapter 7 will bring the conclusion of this research.

7. CONCLUSION

In this chapter we answer the three specific goals of the present research. By doing that, we have provided a set of practices as a suggestion for integrating security requirements in agile software development projects, which represents our answer for the general research question.

The specific goal one was set as "To identify the main sources which generate security requirements in an agile software development project". It was identified that the main sources of security requirements are the public agencies that regulates their segments.

That means laws and regulations written by the agencies that rule over financial and health segments are the main sources of security requirements in the projects conducted by these two segments within the studied company.

We have also identified that security requirements also come from infrastructure and technical sources, but they are not so clear as the prior ones. Security requirements are straight linked to both functional requirements and the non-functional ones. The functional ones, mainly the legal, are more explicit for representing business rules linked to the software functionalities. The non-functional ones must become more explicit in order to not be ignored or undervalued.

The legal sources of security requirements were already mentioned in the literature, as showed in this research. The interviews and documents analyzed in this field study have corroborated what the literature has raised.

Moving forward to the specific goal two, the recommended practices for agile teams to cope with security requirements were also identified. It was found the teams tend to extend the agile framework in order to deal with security requirements.

Such extension may include new practices or even new artifacts that help the team to deal with security issues. The literature has pointed the extended agile framework as a way for dealing with security requirements. It has also mentioned security awareness and security stories as strategies for doing the same.

The interviews and document analysis has showed similar practices. The studied teams have adopted the technical stories as a new artifact for handling security issues. They also trust in another team (GCAD as the operational team) when they need support about questions related to security.

Based on the literature and also in the proceeded field study, we have built a set of suggested practices that are compatible with the agile philosophy. Such set of practices represent the answer of the specific goal three.

The set of proposed practices were presented in Chapter 6, including a new artifact named misuser story for dealing with security issues. That new artifact will follow all the Scrum flow in order to bring security requirements into the spotlight of the formality.

The field study has shown security requirements are implicit in many situations. The misuser story and all the changes it provokes related to practices were presented in Sections 6.1 and 6.2. Misuser stories must be created and tested with the active participation of the operational team. It helps to provide security awareness for the entire agile team, making security requirements visible during all the project time.

SMS, literature, interviews and document analysis have pointed to the risk of a low level of knowledge related to security issues within the agile development team. It has motivated us to bring the operational team into the development process, even in the very beginning when the definition of the user stories and misuser stories are proceeded.

Misuser stories will be shared by the entire agile team, which means all members will be in touch with the security issues. It will also act as security training and security awareness, considering both are pointed by the literature as recommended practices for increasing the security level in software development [66] [65] [87].

Some practices for dealing with security requirements in agile projects are considered heavyweight by agile practitioners. In this sense, we intended to propose a lightweight framework. The misuser story concept is something similar to the user stories, therefore being a familiar practice for those adapted to an agile environment.

A misuser story flows through the Scrum events in the same way as the user stories do. We strongly believe it will represent a light practice that will provide a larger focus on security requirements. A user story points to something allowed to be performed but we believe that showing the things not allowed to be performed is as important as showing the allowed ones. By doing that we think the security requirements may be enlighten and treated in a more formal way.

We also believe the syntax in which misuser stories are suggested to be written will help people to understand about the subject, as they are simple to comprehend even for those who are not familiar with technical issues. That syntax will also make it easier for creating automated tests by providing a structured way of building stories, even the user ones.

By answering the three specific goals, we have answered the general research question on how to integrate security requirements in agile software development. Our answer is an extended agile framework with an application case for helping and guiding during the implementation process.

We have also captured the first impressions of the involved scrum masters (the interviewed ones) regarded to the suggested practices (extended agile framework). Despite

some notes and suggestions, the interviewed scrum masters seem to be opened to test the proposed practices in real projects.

There seem to be a lack of primary studies related to agile projects and security requirements, as presented in the SMS executed prior to this research. That was one of the main reasons we have decided to take a field study for exploring the subject and also for opening the way for a future action research.

In order to summarize our contribution for handling security requirements in agile environments we bring again the main gaps as identified in the beginning of the present research. At this point we are able to add how the proposed extended framework tackles these gaps, as showed in Table 7.1.

Table 7.1 – How the Extended Framework tackles the Gaps

ID	Gap Category	Our Contribution
Gap-01	Lack of appropriate practice	Misuser Stories Extended agile framework Application Case (how to implement)
Gap-02	Lack of security knowledge	Security more explicit It spreads security knowledge Security experts into the process
Gap-03	Lack of primary studies	Primary study (qualitative)

The agile philosophy, specially Scrum, faces the projects as empirical process where each sprint must be evaluated in order to make the next sprint better than the prior one. Such improvement process is continuous, based on the learned lessons of the last sprints [71] [43] [3].

In this sense, we believe the set of proposed practices suggested here (extended agile framework) should be applied in real projects. By doing that, the continuous improvement process would begin and that could be a new research subject. We also believe the set of proposed practices tackles the main gaps raised by the literature, related to security requirements in agile software development.

As a natural next step, an action research should be proceeded in real agile projects. At this time, the researcher could coach the agile team, interfering in the agile process by including the set of proposed practices on their daily routine.

We believe such a new research could validate the recommended practices presented here, contributing to evolve them by making them better. The focus could be even broaden by embracing other non-functional and technical requirements as the performance or the usability ones.

8. LIMITATIONS AND FUTURE RESEARCH

The main limitation of this research derives from its strategy, as a field study can not have its results generalized. In fact the intention here was exploring the subject and creating a set of suggested practices for being tested and validated through another research.

The suggested practices were not tested in real projects which represents another limitation of this study. In order to mitigate such limitation we have collected the first impressions of the interviewed scrum master. We have also provided an application case for guiding the implementation of the suggested practices in real projects.

There were four interviewed scrum masters and that number may be considered low for a scientific purpose. The number four is justified for being the universe of the existing scrum masters in both segments of the studied company, Financial Services and Health Care.

Related to the number of interviewees, Yin [100] believes the amount of interviewees is not the most important point when it comes for a qualitative research. Marconi and Lakatos [34] goes beyond adding the interviewees must be chosen by privileging the most experienced individuals or the ones with leadership positions in the company.

Gil [39] follow the same line stating that the researcher should try to capture the whole scenario, stopping the interviews when things start to be repetitive. As the universe of scrum masters was interviewed, the requirements pointed by Yin[100], Marconi and Lakatos [34] and Gil [39] were fully attended.

The personal bias is another limitation to be considered, as the researcher is an employee in the studied company. That means the researcher is an active part at the unit of study. Such situation might have a negative impact over the results, but on the other hand the prior knowledge of the researcher related to the organization and its business may have provided a deeper basis for understanding the issues raised during the interviews and document analysis.

The level of experience related to the Scrum framework is another point to be highlighted, as both the studied segments within the organization have adopted Scrum about a year ago. Maybe the present research should be applied in other companies, the ones who are more experienced when it comes to agility, specially Scrum.

We strongly believe a future research should follow the present one, adopting an action research strategy. That would be the next natural step for validating the set of proposed practices (extended framework) on dealing with security requirements. Such a research may even improve the quality of the proposed practices by observing them in agile environments through real projects.

Within this next research, the researcher should provide training for the agile teams on how to implement such practices based on the application case presented at Section 6.2.

Here we believe the set of proposed practices would enter in the continuous improvement cycle mentioned by the Scrum framework [71] [43] [3].

As already mentioned, Scrum views a project as an empirical process where the quality of the work should be improved at each new sprint based on the learned lessons of the prior sprints [71] [43] [3]. By validating the set of proposed practices (extended agile framework) in real projects through an action research strategy, we believe that continuous improvement cycle would be started.

Despite the present research has approached just one agile framework, we believe the set of proposed practices are not restricted only to the Scrum one. It may be necessary some degree of adaptation but it is possible to test them considering other agile frameworks.

REFERENCES

[1] Alistair, C. “Writing effective use cases”. Michigan: Addison-Wesley, 2001, 204p.

[2] Allen, J. H. “Governing for enterprise security”, Technical Report, Carnegie Mellon University, 2005, 81p.

[3] Alliance. “Learn about scrum”. Source: https: //www.scrumalliance.org/learn-about-scrum, Dec 2018.

[4] Anderson, D. J. “Kanban: successful evolutionary change for your technology business”. Blue Hole Press, 2010, 262p.

[5] Andreas, L.; Sindre, G. “Eliciting security requirements by misuse cases”, *IEEE Computer Society*, vol. 10, Jan 2000, pp. 120–131.

[6] Arkin, B.; Stender, S.; McGraw, G. “Software penetration testing”, *IEEE Security & Privacy*, vol. 3, Jan-Feb 2005, pp. 84–87.

[7] Austin, A.; Williams, L. “One technique is not enough: A comparison of vulnerability discovery techniques”. In: Proceedings of the International Symposium on Empirical Software Engineering and Measurement, 2011, pp. 97–106.

[8] Ayalew, T.; Kidane, T.; Carlsson, B. “Identification and evaluation of security activities in agile projects”. In: Proceedings of the Nordic Conference on Secure IT Systems, 2013, pp. 139–153.

[9] Azham, Z.; Ghani, I.; Ithnin, N. “Security backlog in scrum security practices”. In: Proceedings of the Fifth Malaysian Conference in the Software Engineering, 2011, pp. 414–417.

[10] Baca, D.; Boldt, M.; Carlsson, B.; Jacobsson, A. “A novel security-enhanced agile software development process applied in an industrial setting”. In: Proceedings of the Tenth International Conference on Availability, Reliability and Security, 2015, pp. 11–19.

[11] Baca, D.; Carlsson, B. “Agile development with security engineering activities”. In: Proceedings of the International Conference on Software and Systems Process, 2011, pp. 149–158.

[12] BARDIN, L. “Análise de Conteúdo”. Edições 70, 2011, 229p.

[13] Bartsch, S. “Practitioners’ perspectives on security in agile development”. In: Proceedings of the Sixth International Conference on the Availability, Reliability and Security, 2011, pp. 479–484.

[14] Bass, L.; Nord, R.; Wood, W.; Zubrow, D. “Risk themes discovered through architecture evaluations”. In: Proceedings of the the Working IEEE/IFIP Conference on Software Architecture, 2007, pp. 1–1.

[15] Beck, K.; Beedle, M.; Van Bennekum, A.; Cockburn, A.; Cunningham, W.; Fowler, M.; Grenning, J.; Highsmith, J.; Hunt, A.; Jeffries, R.; et al.. “Manifesto for agile software development”. Source: http://agilemanifesto.org/, May 2018.

[16] Beck, K.; Gamma, E. “Extreme programming explained: embrace change”. Addison-Wesley Professional, 2000, 189p.

[17] Bellomo, S.; Ernst, N.; Nord, R. L.; Ozkaya, I. “Evolutionary improvements of cross-cutting concerns: Performance in practice”. In: Proceedings of the the International Conference on Software Maintenance and Evolution, 2014, pp. 545–548.

[18] Ben Othmane, L.; Angin, P.; Weffers, H.; Bhargava, B. “Extending the agile development process to develop acceptably secure software”, *IEEE Transactions on Dependable and Secure Computing*, vol. 11, Nov-Dec 2014, pp. 497–509.

[19] Bertoglio, D. D.; Zorzo, A. F. “Um mapeamento sistemático sobre testes de penetração”, Technical Report, Pontifícia Universidade Católica do Rio Grande do Sul, 2015, 42p.

[20] Bertoglio, D. D.; Zorzo, A. F. “Tramonto: Uma estratégia de recomendações para testes de penetração”. In: Proceedings of the XVI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais, 2016, pp. 366–379.

[21] Bertoglio, D. D.; Zorzo, A. F. “Overview and open issues on penetration test”, *Journal of the Brazilian Computer Society*, vol. 23, Feb 2017, pp. 16.

[22] Beznosov, K.; Kruchten, P. “Towards agile security assurance”. In: Proceedings of the Workshop on New Security Paradigms, 2004, pp. 47–54.

[23] Boström, G.; Wäyrynen, J.; Bodén, M.; Beznosov, K.; Kruchten, P. “Extending xp practices to support security requirements engineering”. In: Proceedings of the International Workshop on Software Engineering for Secure Systems, 2006, pp. 11–18.

[24] Brazil. “Lei 13.709”. Source: http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2018/Lei/L13709.htm, Feb 2019.

[25] Breivik, G. “Abstract misuse patterns—a new approach to security requirements”, *Department of Information Science University of Bergen*, vol. 23, Jan 2002, pp. 16.

[26] Budgen, D.; Turner, M.; Brereton, P.; Kitchenham, B. “Using mapping studies in software engineering”. In: Proceedings of the Psychology of Programming Interest Group, 2008, pp. 195–204.

[27] Camacho, C. R.; Marczak, S.; Cruzes, D. S. “Agile team members perceptions on non-functional testing: influencing factors from an empirical study”. In: Proceedings of the Eleventh International Conference on Availability, Reliability and Security, 2016, pp. 582–589.

[28] Cao, L.; Ramesh, B. “Agile requirements engineering practices: An empirical study”, *IEEE Software*, vol. 25, Jan-Feb 2008, pp. 60–67.

[29] Coffman, E.; Galambos, J.; Martello, S.; Vigo, D. “Guide software development with data protection by design and by default”. Source: https://www.datatilsynet.no/en/regulations-and-tools/guidelines/data-protection-by-design-and-by-default/, Jun 2018.

[30] Collins, E. F.; de Lucena, V. F. “Software test automation practices in agile development environment: An industry experience report”. In: Proceedings of the Seventh International Workshop on Automation of Software Test, 2012, pp. 57–63.

[31] Constantine, L. L.; Lockwood, L. A. “Software for use: a practical guide to the models and methods of usage-centered design”. Pearson Education, 1999, 579p.

[32] Crispin, L.; Gregory, J. “Agile testing: A practical guide for testers and agile teams”. Pearson Education, 2009, 577p.

[33] Cucumber. “Gherkin syntax”. Source: https://cucumber.io/docs/gherkin/, May 2019.

[34] de Andrade Marconi, M.; Lakatos, E. M. “Técnicas de Pesquisa”. Atlas, 2002, 277p.

[35] Deloitte. “Lei geral de proteção de dados”. Source: https://www2.deloitte.com/br/pt/pages/risk/articles/lgpd.html, Mar 2019.

[36] Felderer, M.; Büchler, M.; Johns, M.; Brucker, A. D.; Breu, R.; Pretschner, A. “Security testing: A survey”. In: *Advances in Computers*, Elsevier, 2016.

[37] Fitzgerald, B.; Stol, K.-J. “Continuous software engineering: A roadmap and agenda”, *Journal of Systems and Software*, vol. 123, Jan 2017, pp. 176–189.

[38] Fong, E.; Okun, V. “Web application scanners: definitions and functions”. In: Proceedings of the Fortieth Annual Hawaii International Conference on System Sciences, 2007, pp. 280b–280b.

[39] Gil, A. C. “Como elaborar projetos de pesquisa”. Atlas, 2002, 176p.

[40] Godoy, A. S. “Pesquisa qualitativa: tipos fundamentais”, *Revista de Administração de Empresas*, vol. 35, May-Jun 1995, pp. 20–29.

[41] Gregory, J.; Crispin, L. “More Agile Testing: Learning Journeys for the Whole Team”. Addison-Wesley Professional, 2014, 544p.

[42] Group, O. I. S. S. “Information systems security assessment framework. open information systems security group”. Source: http://www.oissg.org/issaf.html, May 2018.

[43] Guides.org, S. “Scrum guides”. Source: https://www.scrumguides.org/, Jun 2018.

[44] Haley, C.; Laney, R.; Moffett, J.; Nuseibeh, B. “Security requirements engineering: A framework for representation and analysis”, *IEEE Transactions on Software Engineering*, vol. 34, Jan-Feb 2008, pp. 133–153.

[45] Hertzog, P. J. “Open source security testing methodology manual”. Source: http://www.isecom.org/osstmm, May 2018.

[46] Hibshi, H.; Slavin, R.; Niu, J.; Breaux, T. D. “Rethinking security requirements in re research”, Technical Report, University of Texas at San Antonio, 2014, 13p.

[47] Holik, F.; Horalek, J.; Marik, O.; Neradova, S.; Zitta, S. “Effective penetration testing with metasploit framework and methodologies”. In: Proceedings of the Fifteenth International Symposium on Computational Intelligence and Informatics, 2014, pp. 237–242.

[48] Howard, M.; Lipner, S. “The security development lifecycle”. Microsoft Press Redmond, 2006, 23p.

[49] Httermann, M. “DevOps for developers”. Apress, 2012, 196p.

[50] Jacobson, I. “Object-oriented software engineering: a use case driven approach”. Pearson Education India, 1993, 552p.

[51] Keramati, H.; Mirian-Hosseinabadi, S.-H. “Integrating software development security activities with agile methodologies”. In: Proceedings of the IEEE/ACS International Conference on Computer Systems and Applications, 2008, pp. 749–754.

[52] Kitchenham, B. “What’s up with software metrics?–a preliminary mapping study”, *Journal of Systems and Software*, vol. 83, Jan 2010, pp. 37–51.

[53] Kitchenham, B.; Charters, S. “Guidelines for performing systematic literature reviews in software engineering”, Technical Report, Software Engineering Group School of Computer Science and Mathematics, 2007, 65p.

[54] Lamport, L. “Proving the correctness of multiprocess programs”, *IEEE Transactions on Software Engineering*, vol. SE-3, Mar 1977, pp. 125–143.

[55] Lee, J.; Kang, S.; Lee, D. “Survey on software testing practices”, *IET Software*, vol. 6, Jun 2012, pp. 275–282.

[56] Liu, B.; Shi, L.; Cai, Z.; Li, M. “Software vulnerability discovery techniques: A survey”. In: Proceedings of the Fourth International Conference on Multimedia Information Networking and Security, 2012, pp. 152–156.

[57] Marczak, S.; Damian, D. “How interaction between roles shapes the communication structure in requirements-driven collaboration”. In: Proceedings of the Nineteenth IEEE International Requirements Engineering Conference, 2011, pp. 47–56.

[58] McDermott, J. “Abuse-case-based assurance arguments”. In: Proceedings of the Seventeenth Annual Computer Security Applications Conference, 2001, pp. 366–374.

[59] McDermott, J.; Fox, C. “Using abuse case models for security requirements analysis”. In: Proceedings of the Fifteenth Annual Computer Security Applications Conference, 1999, pp. 55–64.

[60] McGrath, J. E. “Methodology matters: Doing research in the behavioral and social sciences”. In: *Readings in Human–Computer Interaction*, Elsevier, 1995.

[61] McGraw, G. “Seven touchpoints for software security”. Source: http://www.swsec.com/resources/touchpoints/, Maio 2018.

[62] McGraw, G. “Software security: building security in”. Addison-Wesley Professional, 2006, 448p.

[63] McGraw, G. “Software security”, *IEEE Security and Privacy*, vol. 2, Mar-Apr 2004, pp. 80–83.

[64] Mellado, D.; Blanco, C.; Sánchez, L. E.; Fernández-Medina, E. “A systematic review of security requirements engineering”, *Computer Standards & Interfaces*, vol. 32, Jun 2010, pp. 153–165.

[65] Microsoft. “Agile development using microsoft security development lifecycle”. Source: http://www.microsoft.com/en-us/sdl/discover/sdlagile.aspx, May 2018.

[66] Microsoft. “Security development lifecycle for agile development”. Source: https://msdn.microsoft.com/en-us/library/windows/desktop/ee790621.aspx, Jun 2018.

[67] Moe, N. B.; Cruzes, D.; Dybå, T.; Mikkelsen, E. “Continuous software testing in a globally distributed project”. In: Proceedings of the IEEE Tenth International Conference on Global Software Engineering, 2015, pp. 130–134.

[68] Mylopoulos, J.; Chung, L.; Nixon, B. “Representing and using nonfunctional requirements: A process-oriented approach”, *IEEE Transactions on Software Engineering*, vol. 18, Jun 1992, pp. 483–497.

[69] NICKERSON, C.; Kennedy, D.; Smith, E.; Rabie, A.; Friedli, S.; Searle, J.; Knight, B.; Gates, C.; McCray, J. “Penetration testing execution standard”. Source: http://www.pentest-standard.org/index.php/Main_Page, Nov 2018.

[70] OMG, O. M. G. “Unified modeling language”. Source: https://www.omg.org/spec/UML, Jun 2018.

[71] ORG, S. “The home of scrum”. Source: https://www.scrum.org/, Jun 2018.

[72] Oueslati, H.; Rahman, M. M.; ben Othmane, L.; Ghani, I.; Arbain, A. F. B. “Evaluation of the challenges of developing secure software using the agile approach”, *International Journal of Secure Software Engineering*, vol. 7, Jan 2016, pp. 17–37.

[73] OWASP. “Owasp testing guide v4 table of contents”. Source: https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents, May 2018.

[74] OWASP. “The open web application security project”. Source: http://www.owasp.org, Jun 2018.

[75] Peeters, J. “Agile security requirements engineering”. In: Proceedings of the Symposium on Requirements Engineering for Information Security, 2005, pp. 1–4.

[76] Peled, D.; Havelund, K. “Refining the safety–liveness classification of temporal properties according to monitorability”. In: *Models, Mindsets, Meta: The What, the How, and the Why Not?*, Springer, 2019.

[77] Petersen, K.; Feldt, R.; Mujtaba, S.; Mattsson, M. “Systematic mapping studies in software engineering.” In: Proceedings of the International Conference on Evaluation Assessment in Software Engineering, 2008, pp. 68–77.

[78] Peterson, G. “Collaboration in a secure development process part 1”, *Information Security Bull*, vol. 9, Jun 2004, pp. 165–172.

[79] PMI. “PMBOK® GUIDE Project Management Body of Knowledge 6th Edition”. Project Management Institute, 2017, 448p.

[80] Prandini, M.; Ramilli, M. “Towards a practical and effective security testing methodology”. In: Proceedings of the IEEE Symposium on Computers and Communications, 2010, pp. 320–325.

[81] Richardson, R. J. “Pesquisa Social: métodos e técnicas”. Atlas, 1999, 334p.

[82] Rindell, K.; Hyrynsalmi, S.; Leppänen, V. “Securing scrum for vahti.” In: Proceedings of the Symposium on Programming Languages and Software Tools, 2015, pp. 236–250.

[83] Rindell, K.; Hyrynsalmi, S.; Leppänen, V. “A comparison of security assurance support of agile software development methods”. In: Proceedings of the Sixteenth International Conference on Computer Systems and Technologies, 2015, pp. 61–68.

[84] Rindell, K.; Hyrynsalmi, S.; Leppänen, V. “Case study of security development in an agile environment: building identity management for a government agency”. In: Proceedings of the Eleventh International Conference on Availability, Reliability and Security, 2016, pp. 556–563.

[85] Roesch, S. M. A.; et al.. “Projetos de estágio e de pesquisa em administração”. Atlas, 1999, 336p.

[86] Rumbaugh, J. “Getting started”, *Journal of Object-Oriented Programming*, vol. 7, Sep 1994, pp. 8–23.

[87] Saldanha, L. R. “Security requirements in the agile software development: a systematic mapping study”, Technical Report, Pontifical Catholic University of Rio Grande Do Sul, 2019, 32p.

[88] Sindre, G.; Firesmith, D. G.; Opdahl, A. L. “A reuse-based approach to determining security requirements”. In: Proceedings of the Ninth International Workshop on Requirements Engineering: Foundation for Software Quality, 2003.

[89] Sindre, G.; Opdahl, A. L. “Templates for misuse case description”. In: Proceedings of the Seventh International Workshop on Requirements Engineering, Foundation for Software Quality, 2001, pp. 1–13.

[90] Sindre, G.; Opdahl, A. L. “Eliciting security requirements with misuse cases”, *Requirements Engineering*, vol. 10, Jan 2005, pp. 34–44.

[91] Spencer, D. “Card sorting: Designing usable categories”. Rosenfeld Media, 2009, 176p.

[92] Stouffer, K.; Falco, J.; Scarfone, K. “Nist sp 800-115: technical guide to information security testing and assessment”. Source: https://www.nist.gov/publications/technical-guide-information-security-testing-and-assessment, May 2018.

[93] Taipale, O.; Smolander, K. “Improving software testing by observing practice”. In: Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering, 2006, pp. 262–271.

[94] Tondel, I. A.; Jaatun, M. G.; Meland, P. H. “Security requirements for the rest of us: A survey”, *IEEE Software*, vol. 25, Jan-Feb 2008, pp. 20–27.

[95] Vähä-Sipilä, A. “Software security in agile product management”. Source: https://www.owasp.org/index.php/Main_Page, Oct 2018.

[96] van Wyk, K. R.; McGraw, G. “Bridging the gap between software development and information security”, *IEEE Security & Privacy*, vol. 3, Sep-Oct 2005, pp. 75–79.

[97] VersionOne. “12th annual state of agile report”. Source: https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report, Jun 2018.

[98] Viega, J. “Building security requirements with clasp”. In: Proceedings of the ACM Software Engineering Notes, 2005, pp. 1–7.

[99] Wohlin, C. “Guidelines for snowballing in systematic literature studies and a replication in software engineering”. In: Proceedings of the Eighteenth International Conference on Evaluation and Assessment in Software Engineering, 2014, pp. 38.

[100] Yin, R. K. “Estudo de Caso-: Planejamento e Métodos”. Bookman editora, 2015, 334p.

[101] Zanin, A.; Zorzo, A. F.; Nunes, H. C. “An example of aquila usage: A dsl for model based testing in agile environments”, Technical Report, Pontifical Catholic University of Rio Grande Do Sul, 2018, 14p.

[102] Zhao, J. J.; Zhao, S. Y. “Opportunities and threats: A security assessment of state e-government websites”, *Government Information Quarterly*, vol. 27, Jan 2010, pp. 49–56.

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br