COPPE
UFRJ
Instituto Alberto Luiz Coimbra de
Pós-Graduação e Pesquisa de Engenharia

UNDERSTANDING FACTORS AND PRACTICES OF SOFTWARE SECURITY AND PERFORMANCE VERIFICATION

Victor Vidigal Ribeiro

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Guilherme Horta Travassos
Daniela Soares Cruzes

Rio de Janeiro
Setembro de 2019

UNDERSTANDING FACTORS AND PRACTICES OF SOFTWARE SECURITY AND PERFORMANCE VERIFICATION

Victor Vidigal Ribeiro

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

____
Prof. Guilherme Horta Travassos, D.Sc.

____
Profa. Daniela Soares Cruzes, D.Sc.

____
Profa. Claudia Maria Lima Werner, D.Sc.

____
Prof. Toacy Cavalcante de Oliveira, D.Sc.

____
Prof. Marcos Kalinowski, D.Sc.

____
Prof. Raphael Carlos Santos Machado, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2019

Ribeiro, Victor Vidigal

Understanding factors and practices of software security and performance verification / Victor Vidigal Ribeiro. – Rio de Janeiro: UFRJ/COPPE, 2019.

XVII, 245 p.: il.; 29,7 cm.

Orientadores: Guilherme Horta Travassos
Daniela Soares Cruzes

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2019.

Referências Bibliográficas: p. 146-158.

1. Verificação de software. 2. Verificação de segurança. 3. Verificação de desempenho. I. Travassos, Guilherme Horta, *et al*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

À minha esposa Thalita e minha filha Lara.

AGRADECIMENTOS

Entendo o doutorado como uma longa jornada que nos permite desfrutar de sensações em intensidades extremas que sem ele, talvez, não seriam possíveis. Ao final desta jornada o resultado é evolução. Uma pessoa que passa por um doutorado aprende a pensar e a utilizar sua capacidade intelectual como ferramenta para qualquer situação de sua vida.

Contudo, este longo caminho não é percorrido de forma solitária, mas acompanhado pelas pessoas que nos cercam. Por isso, sou eternamente grato a absolutamente todos que passaram por mim vida durante esses anos. Algumas pessoas são especiais, pois acompanharam esta jornada de perto. Assim, inicio agradecendo meus orientadores.

Ao meu Orientador, Professor Guilherme H. Travassos, que acompanha minha jornada acadêmica desde o mestrado. Agradeço por você representar o genuíno significado da palavra Orientador. Você fez a diferença em minha vida, me transformando em uma versão evoluída de mim mesmo. Agradeço por compartilhar sua visão sobre Engenharia de software com tanta dedicação e paciência. Agradeço também por seus ensinamentos que ultrapassam os limites da pesquisa e atingiram todas as áreas da minha vida. Você será sempre minha referência e fonte de inspiração.

À minha orientadora Daniela S. Cruzes, pelos conselhos relacionados à pesquisa e motivações pessoais durante os momentos de baixa autoestima. Agradeço por ter viabilizado minha viagem à Noruega, experiência extremamente importante para fortalecer minha formação.

À Profa. Claudia M. L. Werner, por participar da minha banca de defesa de doutorado e pelo conhecimento que pude adquirir nas disciplinas disponibilizadas por ela: reutilização de software (2010) e tópicos especiais em engenharia de software IV (2013).

Agradeço ao professor Toacy Cavalcante pela disponibilidade em participar de minha banca de doutorado, pelas importantes contribuições durante minha qualificação e pela troca de experiências durante a disciplina de tópicos especiais em engenharia de software I (2010).

Agradeço ao Prof. Marcos Kalinowski, por ter se disponibilizado em participar da minha banca de doutorado. Agradeço também pelas importantes contribuições fornecidas durante minha qualificação. Agradeço também por sua postura como

Pesquisador em engenharia de software, pois você é exemplo e inspiração para novos pesquisadores.

Agradeço ao Prof. Raphael C. S. Machado, por participar da minha banca de doutorado. Mesmo conhecendo seu trabalho a pouco tempo, tenho certeza que suas contribuições serão de extrema importância para minha pesquisa.

Ao Prof. Marco Antônio P. Araújo, meu professor na graduação, por ter possibilitado meu primeiro contato com o universo da pesquisa em engenharia de software.

Ao meu Pai (in memoriam), pois mesmo sem ter estudado, teve a inteligência de perceber e me convencer que estudar é o melhor caminho. Agradeço à minha Mãe, pelo apoio constante e irrestrito à todas as decisões que tomei em minha vida.

À minha esposa, Thalita, por me apoiar e suportar todas as ausências e ansiedades resultantes desta fase da minha vida. Agradeço também à minha filha Lara, pois ela foi uma fonte de motivação para que eu fosse adiante nessa jornada.

Agradeço a todos os 34 companheiros do grupo ESE dos quais fui contemporâneo. Em especial, agradeço aqueles que tiveram participação mais intensa em minha pesquisa: Talita V. Ribeiro, Hélvio J. Junior, Rebeca C. Motta, Thiago Souza, Bruno Pedraça, Luciana Nascimento, Jobson Massolar, Paulo Sérgio M. dos Santos, Andrea Doreste, Breno de França. Aqui aprendi a admirar diferentes personalidades. Acredito que fiz amizades verdadeiras que levarei para vida toda.

Ao pessoal do PESC, Gutierrez da Costa, Claudia Prata, Maria Mercedes, Solange Santos e Sônia Galliano pela ajuda administrativa.

À COPPE/UFRJ, por dispor de um ambiente propício à realização deste trabalho.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

COMPREENDENDO PRÁTICAS E FATORES RELACIONADOS À VERIFICAÇÃO DE SEGURANÇA E PERFORMANCE DE SOFTWARE

Victor Vidigal Ribeiro

Setembro/2019

Orientadores: Guilherme Horta Travassos
Daniela Soares Cruzes

Programa: Engenharia de Sistemas e Computação

As atividades diárias da sociedade são fortemente apoiadas por sistemas de software. Assim, falhas de software podem trazer grandes perdas. Por isso, as atividades de verificação são essenciais e devem avaliar as funcionalidades do software e também as propriedades representadas pelos requisitos não-funcionais. Desta forma, esse trabalho torna disponível um corpo de conhecimento contendo a caracterização dos requisitos não-funcionais mais relevantes para os sistemas de software e as abordagens de teste que podem ser utilizadas para avaliar esses requisitos. Particularmente, o trabalho aprofunda-se na caracterização das práticas de verificação de segurança e desempenho utilizadas por organizações de desenvolvimento de software e os fatores que apoiam as tomadas de decisão relacionadas à essas práticas. Adicionalmente, são apresentados fatores moderadores das atividades de verificação de segurança e desempenho bem como ações que podem ser realizadas para promovê-los. A organização desse corpo de conhecimento fez uso de revisões estruturadas da literatura, estudos de caso, *rapid reviews* e *survey*, os quais permitiram gerar resultados baseados em evidência que podem ser utilizados com mais confiança pela indústria para aprimorar suas práticas de verificação de requisitos não-funcionais e pela academia para identificação de desafios de pesquisas na área.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

UNDERSTANDING FACTORS AND PRACTICES OF SOFTWARE SECURITY AND PERFORMANCE VERIFICATION

Victor Vidigal Ribeiro

September/2019

Advisors: Guilherme Horta Travassos
Daniela Soares Cruzes

Department: Computer Science and Systems Engineering

Software systems strongly support the daily activities of society. Thus, software failures can bring huge losses. Therefore, verification activities are essential and should evaluate not only the functionality of the software but also its properties represented by the non-functional requirements. Thus, this work proposes a body of knowledge containing the characterization of the most relevant non-functional requirements for software systems. Besides, the body of knowledge includes the testing approaches that can be used to verify these requirements. In particular, the work goes more in-depth into characterizing security and performance verification practices used by software development organizations and the factors that support decision-making related to these practices. Additionally, moderating factors of security and performance verification activities are presented, as well as actions to their promotion. The organization of this body of knowledge made use of structured literature reviews, case studies, rapid reviews, and surveys. Such investigation strategies enabled the generation of evidence-based results, which can support the software practitioners to improve their non-functional requirement verification practices and software researchers to identify research challenges in the area.

INDEX

INDEX OF FIGURES

INDEX OF TABLES

1 Introduction

This chapter describes what motivated this investigation, defining the research goals and questions. Besides, it presents an overview of the research approach, highlighting the main results. Finally, it presents the outline of this document.

1.1 Motivation and context

The first software-based systems were built to operate in isolation; they had a simple user interface based on a keyboard and a command terminal. Besides, such systems were built to run in specific hardware or platform, and the users operating these systems had significant experience in the application domain and, usually, having computing knowledge.

Such needs changed over time, and the contemporary software systems require integration with other systems for their proper functioning. For instance, software systems use the services provided by a third-party system to user authentication; other systems use maps services of third-party systems or integrate with various social networks (a kind of software).

Additionally, the systems' user interface became more complex, still based on traditional interfaces such as keyboard and mouse, but adding new ways of interactions such as touchscreen, voice, and gesture.

Many contemporary systems run in a set of distinct hardware and platforms to reach a broad spectrum of users. Many software systems now should be able to run in different operating systems (Linux, Windows, Android, iOS) and devices (desktops, laptops, smartphones, tablets, wearables). Such devices can have different configurations and hardware capabilities. For instance, a contemporary software system may have to run on smartphones with Android and iOS. Besides, such smartphones could have different hardware capabilities: processing power, amount of memory, screen size, and availability or absence of technologies such as GPS and 4G. Such needs impose a significant challenge to the developers.

Contemporary systems also have another challenge concerning the users that operate them. Nowadays, daily tasks are performed using software: shopping, food delivery, search for transit routes. Thus, these systems should be built to be operated by people who have little knowledge in the application domain and may not have proper technical expertise.

Besides, many contemporary systems should be context-sensitive, adapting themselves according to the environment. For instance, a video streaming service can decrease the video quality if it detects a slow internet connection. Other systems may provide custom suggestions to their users using artificial intelligence algorithms based on previously collected data.

Therefore, it is feasible to say that the popularization and extensive use of software systems bring benefits to modern life. However, the importance of software systems to contemporary society increases specific concerns regarding some critical quality properties.

Software engineers usually classify such properties as non-functional requirements (NFRs). NFRs represent software properties that are not related to the problem domain, such as security, performance, usability, maintainability, portability. NFRs have always been essential to the success of software systems [Hammani 2014] [Ameller et al. 2012], but contemporary software systems have NFRs as essential properties. For instance, energy efficiency, and portability are crucial features of mobile applications as the energy source of these devices is a battery, and there are different platforms and hardware settings with which the software should be performed [Joorabchi et al. 2013] [Rashid et al. 2015].

Thus, although there are several technologies supporting software development, this is a human-dependent activity and, therefore, error-prone. Therefore, as software systems should meet NFRs, the software development organizations include quality assurance activities throughout the software life cycle to evaluate these properties, preventing the occurrence of failures after software release. Software verification [IEEE-610.12 1990], including testing and reviews, encompasses a set of activities to analyze whether the software is meeting their requirements (including NFRs) without presenting defects.

Therefore, the overall motivation of this thesis is summarized as follows:

- The importance of non-functional requirements for contemporary software systems;
- The need to include quality assurance activities (verification) aiming to assess if the software meets non-functional requirements.

1.2 Problem definition and research questions

Software verification encompasses a set of activities aiming to identify defects. The general idea behind verification activities is to exercise the software or analyze its artifacts to identify defects before the software release [Delamaro et al. 2007]. However, these activities may consume much of development effort, influencing on software cost and time to delivery [NIST 2002].

Some organizations include software verification activities in their software development life cycle, but these activities are often performed *ad-hoc*, without using any specific approach or planning. For instance, there are organizations performing software testing, but exercising only the *happy path* of a use case [Ng et al. 2004].

Regarding the verification of NFRs, the scenario becomes more worrying than the verification of functional requirements. Usually, (1) there is no assessment of NFRs, (2) the assessment of NFRs is performed only at the end of the development process, or (3) verification activities do not prioritize these requirements. Besides, (4) some organizations only run a toolkit without awareness of its efficiency or the techniques implemented by these tools [Larsson et al. 2016] [Camacho et al. 2016].

In this scenario, researches addressing NFRs verification can improve the way the organizations assess such kind of requirement. Thus, our research strategy started with the identification of what are the relevant NFRs and what are the software testing approaches supporting their assessment (Chapter 3). It is essential to mention that in this thesis the relevance of an NFR is the number of works citing it as important.

After this first research cycle, it was possible to increase the understanding of NFRs and order them by relevance. Thus, we realized that the number of relevant NFRs is too extensive, and so we decided to focus on the most relevant NFRs: security and performance (S&P).

Security is relevant owing to critical and sensitive information manipulated and stored by the software systems while performing their tasks. For instance, software systems are responsible for the manipulation of personal data, strategic information of organizations, and control of financial transactions. This information usually requires high confidence and different levels of classification, resulting in a growing interest in accessing it to obtain improper benefits [Labs 2016] [Threat and Index 2017].

Performance is relevant owing to the limitations of computational resources [Zhu et al. 2015]. Long response time can make users migrate to rival software systems, a delayed financial transaction can result in financial losses, and excessive power consumption can make the use of systems unfeasible (if hosted on battery-based devices) or can increase the energy costs of systems running in large data centers.

Security and performance verification are activities that search for defects regarding these specific quality perspectives. Various verification practices and techniques can be used individually or combined, promoting specific benefits and posing various challenges to the verification of S&P [Atifi et al. 2017] [Felderer et al. 2016] [Meira et al. 2016].

However, despite the existence of some S&P verification techniques, software systems still present several defects related to these quality properties. Performance issues account for a significant fault category in specific domains (*e.g.*, telecommunications) [Bertolino 2007], and news reporting systems attacks are increasingly frequent [Symantec 2017]. These may be consequences of (1) the inefficiency of security and performance verification practices, (2) the fact that software organizations do not adopt suitable verification practices, or (3) the lack of evidence-based verification practices owing to the apparent disconnection between academy and industry in this context [Garousi and Felderer 2017]. Furthermore, automated attack scripts, the abundance of attack information, and global interconnection make it easier to attack systems than it was previously [Vaughn et al. 2002].

1.2.1 Research goals

The research goal in its broader scope is to characterize the state of the practice regarding non-functional requirements verification. By presenting the reasoning of the issues of this topic, this work may provide insights for further research to investigate essential points in depth. It is a broad goal, so we have focused on some specific goals that are following listed.

- Propose a *Body of knowledge,* including a characterization of relevant non-functional requirements and the software techniques that can be used to assess such requirements (NFR-BoK):
    - Identify and understand the non-functional requirements representing the most important software properties, *i.e.*, the most relevant non-functional requirements;

  - Identify software testing approaches supporting the assessment of non-functional requirements;
  - Provide information on which testing approaches can be used for each of the non-functional requirements;
  - Provide information on the capability of testing approaches to cover test dimensions (process phases, levels, and techniques);
  - Provide information on which requirements do not have testing techniques that support their evaluation;
- Identify and characterize the security and performance *verification practices* used by software development organizations regarding used techniques, the definition of *done* criteria, automation level, and asset covered;
- Identify the *decision-making factors* related to security and performance verification used by software development organizations;
- Identify the *moderator factors* influencing the security and performance verification; and
- Identify *actions* used to promote security and performance moderator factors.

1.2.2 Research questions

Different groups of research questions guided the two cycles (Section 1.3), composing this research. The first one, aiming to improve the understanding of non-functional requirements and the software testing approaches used to assess them. The results allowed us to observe that security and performance were the most relevant NFRs. Therefore, the second investigation cycle focuses on security and performance verification.

1.2.2.1 Research questions of cycle 1

- **RQ0.1** What are the most relevant non-functional requirements, according to software practitioners?
- **RQ0.2** What are the software testing approaches used to test non-functional requirements?

- **RQ0.3** To what extent do testing approaches support the assessment of non-functional requirements?
    - **RQ0.3.1** What are the relevant NFRs that are not covered by testing approaches?
    - **RQ0.3.2** What are the test dimensions met by the test approaches?

1.2.2.2 Research questions of cycle 2

- **RQ1** Which are the *practices* used by the organizations to support the verification of security and performance?
    - **RQ1.1** What are the standard *techniques*?
    - **RQ1.2** Which definition of *done* do they adopt?
    - **RQ1.3** How is the *level of automation*?
    - **RQ1.4** What are the *assets* covered?
- **RQ2** How do the organizations define their security and performance verification strategies?
    - **RQ2.1** What are the factors influencing the decision-making regarding security and performance verification strategies?
    - **RQ2.2** When are the decisions on the verification strategy made?
    - **RQ2.3** How often are the decisions on the verification strategy made?
    - **RQ2.4** Who is responsible for the decision making regarding the verification strategy?
- **RQ3** What are the moderator factors influencing security and performance verification?
    - **RQ3.1** What actions can be taken to promote moderator factors?

1.3 Research approach, primary results, and contributions

As illustrated in Figure 1, two investigation cycles with six steps compose this research. The scope of the first investigation cycle was related to software testing approaches supporting the assessment of non-functional requirements. Thus, we use the technical literature to gain a better understanding of non-functional testing approaches.

However, as we gained knowledge on the topic, we realized that it would not be feasible to investigate all NFRs in-depth and that software testing is not a suitable approach to assess some NFRs. Therefore, in the second investigation cycle, the scope of this thesis has been adjusted to focus on security and performance, and to encompass static verification activities (including software review).

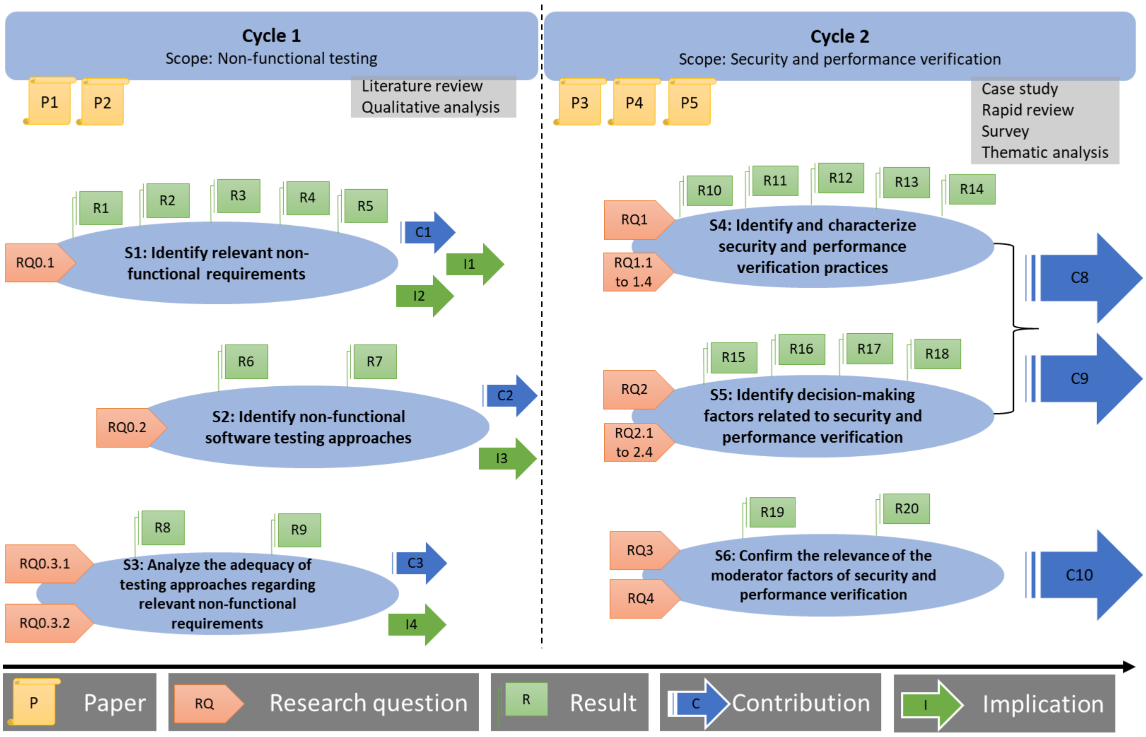

Cycle 1
Scope: Non-functional testing
P1
P2
Literature review
Qualitative analysis
R1
R2
R3
R4
R5
RQ0.1
S1: Identify relevant non-functional requirements
C1
I1
I2
R6
R7
RQ0.2
S2: Identify non-functional software testing approaches
C2
I3
R8
R9
RQ0.3.1
RQ0.3.2
S3: Analyze the adequacy of testing approaches regarding relevant non-functional requirements
C3
I4
Cycle 2
Scope: Security and performance verification
P3
P4
P5
Case study
Rapid review
Survey
Thematic analysis
R10
R11
R12
R13
R14
RQ1
RQ1.1 to 1.4
S4: Identify and characterize security and performance verification practices
C8
R15
R16
R17
R18
RQ2
RQ2.1 to 2.4
S5: Identify decision-making factors related to security and performance verification
C9
R19
R20
RQ3
RQ4
S6: Confirm the relevance of the moderator factors of security and performance verification
C10
P
Paper
RQ
Research question
R
Result
C
Contribution
I
Implication

Figure 1 - Research steps overview

1.3.1 *The first investigation cycle*

- **S1: this step aims to <u>identify relevant non-functional requirements</u>** (Section 3.3)
  - **RQ0.1:** What are the most relevant non-functional requirements, according to software practitioners?
  - **Results**
    - **R1:** identification of 224 non-functional requirements classified as relevant;
    - **R2:** identification of 87 non-functional requirements having description;

        - **R3:** identification of 137 requirements without description. These requirements were not analyzed in depth because there was no confidence in the properties of the software they represented;
        - **R4:** characterization of 87 non-functional requirements;
        - **R5:** identification of non-testable requirements.
    - **Contributions**
        - **C1:** provides information to build a body of knowledge with 87 non-functional requirements (NFR-BoK) and their characterization;
    - **Implications**
        - **I1:** tunes the thesis scope to the most relevant non-functional requirements: security and performance;
        - **I2:** tunes the thesis scope from testing to verification (testing and reviews) (Section 3.3.1).

- **S2: this step aims to <u>identify software testing approaches to assess non-functional requirements</u>** (Section 3.4)
    - **RQ0.2:** What are the software testing approaches used to test non-functional requirements?
    - **Results**
        - **R6:** identification and characterization of 47 non-functional testing approaches;
        - **R7:** lack of confidence in testing approaches as they are not empirically evaluated.
    - **Contributions**
        - **C2:** populates the NFR-BoK with the testing approaches that support the assessment of the relevant non-functional requirements;
    - **Implications**
        - **I3:** strengthens the need for research on the proposed theme.

- **S3: it aims to <u>analyze the adequacy of testing approaches</u> regarding the relevant NFRs** (Section 3.5)
    - **RQ0.3.1:** What are the relevant NFRs that are not covered by testing approaches?
    - **RQ0.3.2:** What are the test dimensions met by the test approaches?

- **Results**
    - **R8:** identification of 13 NFRs with no testing approach to assess them;
    - **R9:** software testing approaches do not cover all testing dimensions (levels and type of techniques) and testing process phases (planning, design, implementation, execution, and analysis).
- **Contributions**
    - **C3:** includes the information in the NFR-BoK on how testing approaches met relevant NFRs.
    - **I4:** Highlight the strengthens the need for researches on the proposed theme owing to the lack of suitable testing approaches;

1.3.2 The second investigation cycle

- **S4: it aims to <u>identify and characterize the security and performance verification practices</u> have been used by software development organizations** (Chapter 4)
    - **RQ1:** Which are the *practices* used by the organizations to support the verification of security and performance?
        - **RQ1.1:** What are the standard *techniques*?
        - **RQ1.2:** Which *definition of done* do they adopt?
        - **RQ1.3:** How is the *level of automation*?
        - **RQ1.4:** What are the *assets* covered?
    - **Results**
        - **R10:** identification of six verification practices the software developments organizations use to assess security and performance;
        - **R11:** identification of the techniques type of each verification practice;
        - **R12:** identification of the definition of *done* criteria
        - **R13:** identification of the automation level – if the practice is automated or manual, and what are the supporting tools;
        - **R14:** identification of the assets covered by the practices.

- **S5: this step aims to identify decision-making factors related to security and performance verification** (Chapter 4)
    - **RQ2:** How does the organization define its security and performance verification strategies?
        - **RQ2.1:** What are the factors influencing the decision-making regarding security and performance verification strategies?
        - **RQ2.2:** When are the decisions on the verification strategy made?
        - **RQ2.3:** How often are the decisions on the verification strategy made?
        - **RQ2.4:** Who is responsible for the decision making regarding the verification strategy?
    - **Results**
        - **R15:** Identification of decision-making factors influencing the choice of security and performance *verification practices*;
        - **R16:** identification of decision-making factors influencing the choice of *support tools*
        - **R17:** identification of decision-making factors influencing the choice of *coverage criterion*
        - **R18:** identification of decision-making factors influencing the choice of *definition of done*

The results of steps **S4** and **S5** were analyzed together. Thus, the following implications are the consequences of these two steps:

- **Contributions of S4 and S5**
    - **C8:** improvement of the knowledge of how verification practices are performed in software development organizations;
    - **C9:** identification of nine conjectures related to security and performance verification – these conjectures were validated through the technical literature (rapid reviews) and practitioners' opinion (survey), and then they became the eight moderator factors.

- **S6: this step aims to <u>confirm the relevance of the moderator factors</u> of security and performance verification** (Chapter 5)
    - **RQ3:** What are the moderator factors influencing security and performance verification?
        - **RQ3.1:** What actions can be taken to promote moderator factors?
    - **Results**
        - **R19:** confirmation of the relevance of eight moderator factors of security and performance verification;
        - **R20:** identification of a set of actions to promote the moderator factors.
    - **Contributions**
        - **C10:** Provides a set of moderating factors influencing the security and performance verification and actions to promote them.

1.3.3 Publications

Five papers disclose the findings of this thesis. Our publishing strategy was to disseminate the results when we understood that they would be relevant to the community of researchers or practitioners:

- **P1:** "Testing non-functional requirements: lacking technologies or researching opportunities?" XV Brazilian Symposium on Software Quality (2016) – This paper consolidates the results of the first investigation cycle. It presents the NFR-BoK including the most relevant NFRs and the software testing approaches used to assess them (Chapter 3);
- **P2:** "*Tecnologia de apoio à composição de estratégias de verificação de segurança e desempenho* (A technology to support the combination of security and performance verification strategies)", XV Workshop de Teses e Dissertações em Qualidade de Software (2017) – It presents the proposal of technology to support the definition of a testing strategy to assess security and performance. This proposal was not evolved, but the paper presents findings that can help future researches;

- **P3:** "*Desafios na verificação de segurança de sistemas de software* (Challenges of software systems security verification)," Workshop de Qualidade de Produto de Software (2017) - This paper provides a warning about the challenges of verifying the software security (Section 2.6.1);
- **P4:** "A perception of the practice of software security and performance verification in the Brazilian industry," 25th Australian Software Engineering Conference (2018) – This paper presents the characterization of security and performance practices used by software development organizations. Besides, it is the first disclosure of a set of nine conjectures that later became the moderator factors of security and performance verification (Chapter 4).
- **P5 (*under review*):** "Study on practices, moderator factors, and decision-making factors of security and performance verification", Software Quality Journal – This journal article encompasses the complete findings of the performed case study, including the characterization of S&P verification practices, the decision-making factors related to S&P verification, and the eight moderator factors influencing S&P verification (Chapters 4 and 5).

1.3.4 Methodological contributions

The main methodological contribution of this thesis is to show how different research methods can be combined into a research project. The description of the methodological steps is presented at the begin of each thesis chapter. Thus, it is possible to understand the applied research methods, increasing the confidence of the findings.

Additionally, Appendix A provides the entire description of the followed methodology to facilitate a broad understanding of the methodological steps followed.

The main contributions regarding methodological issues are the following:

- Demonstrate how to use *structured literature reviews* to build a trustworthy body of knowledge;
- Demonstrate how to use the *coding phase* of grounded theory to analyze the results of a literature review;
- Relevant insights on how to use *rapid reviews* to increase the confidence of *case study* findings;

- Example of use of *thematic analysis* to analyze the data of a *case study* research;
- Present an approach on how to use a *survey* to bring knowledge from industry to academia, validating theoretical results.

1.4 Document outline

After the introduction chapter, this document is organized as follows:

- **Chapter 2:** *From Non-Functional Requirements to Security and Performance Verification: Essential Definitions and Perspectives* – presents the theoretical background, including the definitions of main concepts related to this thesis. The main objective of this chapter is to bring the reader to our perspective, facilitating the thesis understanding.

- **Chapter 3:** *Testing Non-Functional Requirements: A Body of Knowledge* – presents the findings of the first research cycle. It includes a body of knowledge encompassing the most relevant non-functional requirements and the testing approaches used to assess them. The findings presented in this chapter helped to define the scope of this thesis. Methodological issues regarding the of two structured literature reviews performed and the data analysis are also presented.

- **Chapter 4:** *A Perception of the State of the Practice of Security and Performance Verification* – it starts describing the methodology followed by the case study that supports the presented findings. Next, it presents the security and performance verification practices used by software development organizations. These findings come from a case study research and show a characterization of the practices regarding their techniques, the definition of done, automation level, and asset covered. Besides, it presents the decision-making criteria related to verification practices.

- **Chapter 5:** *Moderator Factors of Security and Performance Verification* – this chapter presents the methodology of performed research methods (case study, rapid reviews, and survey). Next, it presents moderation factors that influence the verification of security and performance. Such moderation factors emerged from observations of the practice (case study), and then they were later confirmed through technical literature (rapid reviews) and practitioners' opinions (survey).

- **Chapter 6:** *Conclusion* – This chapter presents the final considerations of the thesis highlighting the research contributions. Besides, it describes the threats to validity and indicates possible future works.

2 From Non-Functional Requirements to Security and Performance Verification: Essential Definitions and Perspectives

This chapter presents the main concepts of the thesis, trying to bring the reader to our perspective. It is crucial owing to the lack of consensus regarding such fundamental concepts. In this way, the chapter presents a discussion about non-functional requirements, showing issues regarding this classification. Besides, it presents the concepts of software verification, software security, and software performance.

2.1 Shortcut to main concepts

Table 1 presents the main global concepts adopted by this work, and the next sections provide the reasoning that led to the adoption of these.

Table 1 - Main concepts index

Concept	Adopted definition
Functional requirement	Describes properties[1] related to the problem domain, specifying functions that a software system must perform.
Non-functional requirement	Describes properties that define conditions[2] for the software system.
Software security	A kind of non-functional requirement. It represents the capability of a software system to protect information and functionalities while allowing authorized users to access information and functionality they have

[1] "*a quality or trait belonging and especially peculiar to an individual or thing*" [Merriam-Webster.com 2019a]

[2] "*a restricting or modifying factor*" [Merriam-Webster.com 2019b]

	permission.
Software performance	A kind of non-functional requirement. It represents the capability of a software system to provide data and functionalities using specified resources and time.
Software verification	A set of activities performed during the software lifecycle aiming to identify discrepancies between what was specified and what is accomplished. It encompasses two different groups of activities: testing and reviews.
Software testing	Dynamic technique aiming to identify software *failures*. Dynamic means the artifact under testing must be executed.
Software review	Static technique aiming to identify software *faults*. Static means the artifact reviewed is not executed.
Security verification	A set of activities aiming to identify failures and faults related to the security of a software system.
Performance verification	A set of activities aiming to identify failures and faults related to the performance of a software system.

In addition to the concepts previously presented, other specific concepts are also crucial for a better understanding of this thesis.

- **Asset:** The part of the system covered by the verification practice, *e.g.*, the source code is an asset regarding static code analysis. It is not defined as an artifact because the verification may target the running system, which is not an artifact.
- **Attack:** The steps a malicious entity performs to the end of turning a threat into an actual corruption of an asset’s properties. Usually, this is done by exploiting a vulnerability. A user who does not have the explicit intention to violate the system can also perform an attack if it performs harmful steps inadvertently.
- **Defect:** a general concept used as a synonym for Failure or Fault. The term ‘vulnerability’ represents a security defect (or fault).

- **Definition of done, acceptance criteria, or stop criteria:** overlapping concepts. The definition of done is used as a criterion to conclude a verification activity.
- **Exploit:** From the perspective of an invader, it is a concrete malicious input making use of the vulnerability in the SUT aiming to violate the property of an asset. From the perspective of the verification team, it is a good test case to identify a software security failure.
- **Failure:** the inability of a system or component to perform its required functions. A manifestation of a Fault.
- **Fault:** an incorrect step, process, or data definition in a computer program. A Vulnerability represents a security Fault.
- **Invader, malicious user, or cracker:** a person who has the explicit intention to exploit the software vulnerabilities. Usually, invaders have great expertise in computation.
- **Malicious insider:** a specific kind of invader having privileged information about the software.
- **Mistake:** developers' action that introduces faults in a software artifact.
- **Performance mechanism:** a software component aiming to increase performance capability. The use of cache technology is an example of a performance mechanism.
- **Security mechanism:** a software component aiming to protect the software against attacks. The authentication functionality is an example of a security mechanism.
- **Threat:** the potential cause of an undesired incident that harms or reduces the value of an asset. For instance, a threat may be a hacker, power outages, or malicious insiders.
- **Verification practice:** what is performed for supporting verification, *e.g.*, unit testing and source code inspection.
- **Vulnerability:** designate security-related faults.

2.2 A perspective about non-functional requirements

Software systems are developed to meet a goal, a purpose. For instance, manage banking transactions, solve civil engineering calculations, or control autonomous vehicles. Therefore, software systems should have a set of properties that lead them to achieve such goals. In the context of software development, such properties are named software requirements.

IEEE 610.12 [1990] provides a widely known definition of a software requirement:

> (1) A condition or capability needed by a user to solve a problem or achieve an objective.
> (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.

The presented definition leads to the perception that requirements always arise from the users' needs or a formal document. However, requirements may arise from other stakeholders (*e.g.*, technical staff) [Aurum and Wohlin 2005], or they can represent technical constraints. Despite, such technical constraints may conflict with users' requirements.

Therefore, this work extends the IEEE 610.12 [1990] concept of software requirements so that it is not relevant from who or what the requirement arises. Thus, **requirement represents properties specifying a system capability or a condition a system must meet.**

The set of a system's properties may be numerous, requiring a large number of requirements to represent them. Besides, the properties may be heterogeneous so that requirements may be classified in different ways. Thus, there are different classification proposals of software requirements [Glinz 2007] [Boehm and Kukreja 2015].

However, this work adopts the classification that split the requirements between functional (FR) and non-functional requirements (NFR). Despite some criticisms regarding that classification, it is widely used by researchers and practitioners, easing the understanding of this work. Thus, it supports the decision to choose this classification.

Additionally, it is essential to note that there is no consensus definition for RF and NFR. In this way, this work adopts the definitions proposed by IEEE 610.12 [1990], but evolving them based on the discussions presented by Afreen *et al.* [2016], Glinz [2005], Chung and Leite [2009], Broy [2015], Boehm and Kukreja [2015], and Eckhardt *et al.* [2016].

A functional requirement describes properties related to the problem domain, specifying functions that a software system must perform. For instance, “The system should allow users to obtain a monthly bank statement.” It is important to note that an FR always represents a property of the product at runtime.

A non-functional requirement describes properties that define conditions for the software system. For example, (1) “The bank statement should be displayed to the user in a maximum of 4 seconds” or (2) “The system’s methods should not have more than five cyclomatic complexity.” It is important to note that an NFR can represent a property of the product at runtime (1) at the development phase (2).

Some authors understand the NFRs as a more general class, encompassing a broader range of system properties. Sommerville [2011] understand that NFRs may represent properties of the product, organizational, or external. Within this kind of understanding, properties such as ‘price,’ ‘cost,’ and ‘time to produce’ can be classified as NFRs [Becha and Amyot 2012].

However, this work aims to deal with the product-related NFRs only, both those representing runtime and development time properties. Such limitation is significant because it is not possible to observe the properties that are not directly related to the product through verification activities.

2.2.1 Functional vs. non-functional requirements: a dangerous but widely used classification

The core issue regarding FR-NFR classification is that it is an exclude-based classification. In this case, the class representing FR is well defined, but the class representing NFRs encompasses every requirement that is not a functional requirement.

An analogy can be used for a better understanding of this issue. Imagine a classification of the existing auto-vehicles (Figure 2). If the auto-vehicles is classified into the classes “Cars” and “Non-cars,” the first class will encompass elements that should have similar pre-defined characteristics (four wheels, a road vehicle, and others). However, the second class will contain elements with distinct characteristics, since there is not a set of pre-defined characteristics defining that an element belongs to this class, but the single criterion an auto-vehicle needs to belong to the class “Non-cars” is not to be a car (exclusion-based classification).

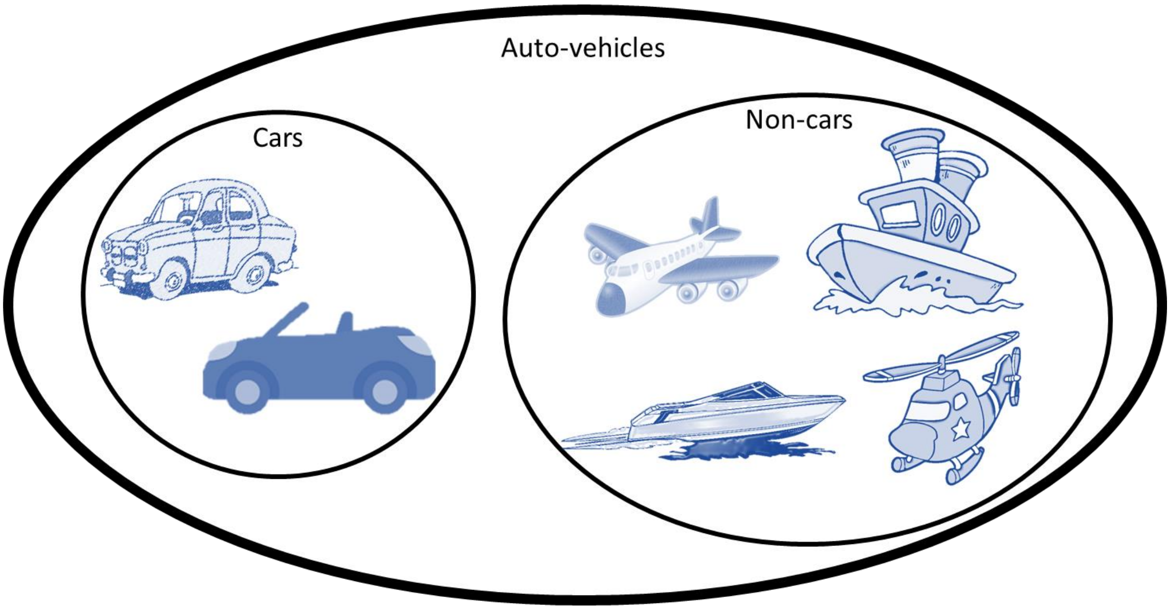
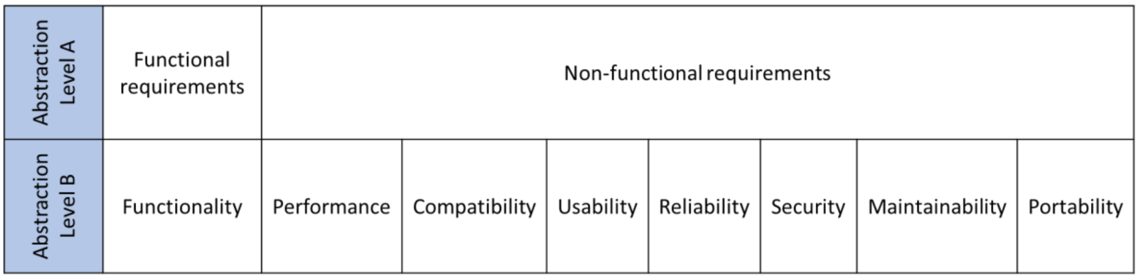


Figure 2 – Auto-vehicles classification analogy

In order to solve this issue, it is necessary to replace the class created by exclusion (“Non-cars”) with other classes that specify well-defined characteristics to its elements - for instance, replacing the “Non-cars” by the classes “aircraft” and “watercraft.”

Similarly, it is essential to note class NFR should be divided into other classes, allowing a better classification of the requirements of a software system. In this way, Figure 3 shows a more suitable classification of the requirements within two abstraction levels.

Therefore, class FR is subdivided into only one class (Functionality) so that it encompasses requirements representing software functionalities related to the problem domain. Class NFR is subdivided into seven classes so that each one encompasses requirements representing conditions related to a specific quality attribute. For example, the class “Performance” encompasses requirements representing conditions related to the response time of the system.



Abstraction Level A	Functional requirements	Non-functional requirements						
Abstraction Level B	Functionality	Performance	Compatibility	Usability	Reliability	Security	Maintainability	Portability

Figure 3 - FR-NFR classification level of abstraction

2.2.2 The importance of NFRs and the lack of suitable supporting technologies

If compared with previous software systems, contemporary software systems require a more significant and heterogeneous set of properties to their success. The NFRs, such as interoperability, portability, usability, performance, and security, represent these properties, and so the importance of NFRs is increased.

Such reasoning let us coin two hypotheses: (I) NFRs represent the most important properties of a software system, and (II) there are not suitable testing techniques to handle such NFRs. However, this initial perception could be a researcher's bias caused by his contextualized development experience, and it might not be a reality in other projects.

In this way, an ad-hoc literature review was performed aiming to identify if there is an overall perception of the NFRs' importance and if there are suitable techniques to handle such NFRs. Table 2 highlights the main assertions that led to the hypotheses confirmation, and afterward, we provide descriptions of the included papers.

Table 2 - Assertions regarding NFRs importance and the lack of technologies

Importance of NFRs
Compliance with FR is not enough [Ebert, 1998].
RNFs are as important as RF [Ameller et al. 2013].
The NFRs have a global and multiplicative influence on the system [Boehm and Kukreja 2015].
The economics of a software product is related to the RNFs [Barney et al. 2008] [Berntsson Svensson et al. 2009] [Boehm and Kukreja 2015].
Non-compliance to NFRs harms the user experience [Baltes et al. 2015].
Compliance with NFRs is essential to keep the software systems alive along the time. It is necessary for the economic viability and ongoing functioning of the organizations and segments of society that depend on those software systems [Carroll et al. 2015].
Lack of suitable technologies targeting NFRs
Inadequate Verification & Validation technologies targeting NFRs [Borg et al. 2003] [Ameller et al. 2012] [Hammani 2014] [Larsson et al. 2016] [Camacho et al. 2016].
Unsuitable technologies to identify, document, and manage trade-offs of the NFRs [Chung et al. 2000] [Borg et al. 2003] [Svensson et al. 2010] [Ullah et al. 2011] [Hammani 2014].
Lack of technologies to estimate the cost of NFRs [Svensson et al. 2010].
FRs drive most technologies to system design [Chung et al. 2000].

2.2.2.1 The importance of the NFRs

Ebert [1998] discussed the importance of NFRs to the success of a software product. He states that even if a system meets the FRs, it could present a set of operating issues such as a large number of failures.

Barney *et al.* [2008] performed a case study aiming to identify the factors used to prioritize requirements. This study concluded that the value of a software product is directly related to NFRs.

In a study encompassing five software development organizations, Berntsson Svensson *et al.* [2009] concludes the NFRs are essential for software products to achieve their goals. They got statements like "*If the product is not usable, we will not be able to sell it.*"

A survey aiming to identify the opinion of developers and practitioners regarding the importance of NFRs compared to FRs. Among 31 valid answers, 21 participants considered NFRs as important as FRs, and 4 participants considered NFRs more important than FRs [Ameller et al. 2013].

According to Baltes *et al.* [2015], a software system that does not meet the NFRs presents critical failures as they corrupt the user experience, reduce the system performance, and result in loss of computing resources.

The NFR effect on the system is system-wide because one NFR can make an influence in a set of FR. For example, if the maximum response time of 1s is defined globally, every FR of the system should meet such response time condition. Besides, the effect of NFRs on the system cost is multiplicative. In a real example, changing the maximum response time from 1 to 4 seconds reduced the cost to build the software from $100 million to $30 million. In this example, it was possible to show that a response time of 4 seconds was acceptable for 90% of the transactions and achievable via industrial technology. Thereby, the use of an expensive custom technology was avoidable [Boehm and Kukreja 2015].

The (non-)compliance with NFRs can result in consequences beyond the software system boundary. For instance, the maintainability is essential to keep a software system operational and valuable for a long time. Besides, the long life of software systems is essential for the viability and ongoing functioning of organizations and segments of the society that depend on those systems [Carroll et al. 2015].

Therefore, it is possible to conclude that the importance of NFRs is a consensus between researchers and practitioners.

2.2.2.2 The lack of technologies targeting NFRs

According to Chung *et al.* [2000], most of the conventional approaches to design software systems are guided and prioritized by RFs, although some approaches include NFRs in a non-systematic way and often do not documenting them. Consequently, NFRs are seen more because of development decisions than a goal to be achieved.

Borg *et al.* [2003] presented the findings of a case study showing the software development organizations face difficulties in identifying NFRs because requirements gathering phase focus on RFs. Thus, NFRs documentation is often vague. Besides, NFRs management is insufficient and sometimes missing. Regarding the testing phase, they conclude that to evaluate RNFs is a challenge because of their nature and the way they are represented (non-measurable). Thus, NFRs testing is sometimes an impossible or time-consuming activity.

Svensson *et al.* [2010] presented the findings of a literature review aiming to identify experimental studies of NFRs management. Such a study concludes that there is no clear view of how to elicit NFRs; some NFRs are inaccurately specified, impairing their assessment; only one of the approaches found presented a way to estimate the cost of NFRs. Additionally, another literature review aiming to identify challenges regarding NFRs elicitation concluded that most of the techniques that handle NFRs are partial and incomplete [Ullah et al. 2011].

Ameller *et al.* [2012] performed a set of structured interviews with software architects. The researchers concluded that the users understand the importance of the NFRs in the final product. However, during the requirements gathering phase, the users do not mention this kind of requirement. Besides, most of the respondents said that they do not document NFRs, and the NFRs assessment (verification & validation) is subjective.

Hammani [2014] published the findings of a literature review regarding approaches to modeling NFRs in the context of software product lines. The identified approaches addressed specific NFRs (reliability and performance), excluding important NFRs, and most of them were initial proposals of limited tools.

Larsson et al. [2016] highlighted five challenges of NFRs testing: (1) changing NFRs documentation because NFRs evolve according to system understanding increase; (2) Managers need to understand the business because NFRs cannot be interpreted in isolation; (3) NFRs are not quantified, impairing assessment; (4) NFR are not prioritized; (5) challenge to generate test cases data to simulate the possible system production time states.

Finally, Camacho, Marczak, and Cruzes [2016] performed research aiming to characterize how agile organizations are conducting NFRs testing. The researchers identified seven factors influencing NFRs testing activities. (1) Priority – focusing on NFR depends on the moment, business priority, budget, feature characteristic. (2) Time pressure – it is a factor to define testing priority, and FR testing takes the prioritization over NFR in case of time pressure. (3) Cost – if the cost of NFRs testing is higher than the cost of a failure, it is unfeasible. (4) Technical issues – representing the lack of techniques addressing NFRs and the lack of awareness about their importance. (5) Awareness – every stakeholder should be aware of the importance of NFRs. (6) Culture – it is related to the default developers' habits. (7) Experience – the most experienced team members defend NFRs testing.

Therefore, the above-presented challenges reflect the lack of technologies addressing NFRs during every phase of the software life cycle.

2.3 Software security

First, it is crucial to understand that the term 'software security' refers to a different concept of 'security software.' Security software is a kind of software system that aims to protect other systems against security issues — for example, firewalls, antivirus, antispyware. This thesis does not intend to address such kind of software system. This thesis address security as a software property, classifying security as an NFR. Therefore, this research is about software security.

Software security or security engineering is the idea of engineering a software system so that it keeps working correctly even under malicious attack [Mcgraw 2004], *i.e.*, how to use software engineering to build secure software. Thus, it is essential to provide the meaning of the term security.

This thesis adopts the definition of ISO-25010 [2011] so that **security is a capability of the software to protect information and functionalities while allowing authorized users to access information and functionality they have permission.**

Additionally, security may be broken down in other more specialized NFRs, despite there is no consensus regarding the NFRs composing security. Therefore, an *ad-hoc* literature review was performed to identify the NFRs composing security [McDonald et al. 2006] [ISO/IEC 25010 2011] [Stallings et al. 2013] [Felderer et al. 2016].

- **Authenticity:** software capability to recognize that the claimed identity of an entity[3] is true.
- **Availability:** software capability to guarantee timely and reliable access to data and information services for authorized uses.
- **Confidentiality:** software capability to not disclose information to an entity that does not have access permission.
- **Integrity:** software capability to ensure that information and the system itself is changed only by entities having the correct access permission.
- **Non-repudiation:** software capability to provide mechanisms to prove the actions and events so that they cannot be denied.
- **Accountability:** software capability to provide mechanisms allowing the performed actions could be assigned to only one entity.

2.4 Software performance

Performance is a low complexity NFR as it is easy to understand and observe. However, it is identified as one of the most critical NFR [Ribeiro and Travassos, 2016]. In this thesis, **performance is the capability of a software system to provide data and functionalities using specified resources and time**.

Performance encompasses three more specialized NFRs representing the dimensions of time, resources, and usage intensity. The NFRs composing performance were identified through an *ad-hoc* literature review [Vara et al. 2011] [Daud and Kadir 2012] [Soares et al. 2014] [Mairiza et al. 2010] [Ameller et al. 2016] [Caro et al. 2008] [Becha and Amyot 2012] [Ermilov et al. 2014].

- **Resource consumption:** software capability to provide data and functionalities using the specified amount of resources. Usually, the resources are related to

[3] Users, process, systems, resources, messages, transmissions

hardware, such as memory usage, CPU, and disk storage. However, other kinds of resources can also be considered, for example, papers used in prints or the amount of printer ink.

- **Time behavior:** software capability to provide data and functionalities obeying specified time constraints. For instance, it encompasses the time from a request to a response, the processing time required by a functionality, time to communicate with other systems or devices.
- **Scalability:** software capability to maintain the specified performance when used under high demand. For example, a system must use the specified amount of memory and response on time, even when a significant number of users are using it.

2.5 Software verification

Despite the software development technologies evolution (methodologies, methods, supporting tools, process), it remains an intensive human activity, so that software development is error-prone. For example, the development of a software product is divided into phases: requirements, design, coding, and assessment. Each phase generates a model representing the software (requirements document, UML diagrams, source code) so that the next phase uses the previous model, evolving them to represent a more detailed view of the software system. The creation of the models representing the system and the transformations of such models depend on the knowledge and interpretation of the development team - being an error-prone activity.

Software engineering provides technologies to support software development activities aiming to produce software that meets their goals (avoiding failures), on time and cost. However, such technologies are also performed by humans so that the risk of mistakes is imminent. Therefore, software engineering contemplates a topic aiming to mitigate such risk - Verification and Validation (V&V) [Sommerville 2011].

The validation aims to assess the system regarding users' needs, that is, assess if the correct system was built. They are essential activities because failure-free software can be built, but if it does not meet users' needs, then it is useless [Delamaro et al. 2007]. However, validation activities are not the subject of this thesis.

The verification aims to assess if the built software system meets its requirements, that is, assess if the system was built correctly [Sommerville 2011].

Verification techniques are classified regarding the need to execute (or not) the software. The dynamic techniques are those requiring software execution, and they are named software testing. The static techniques do not require the execution of the software, and they are named software reviews.

At this point, it is essential to advertise that some practitioners are misleading such classification as they coined the concept of static testing. For example, they are classifying automated code analysis as a static testing technique because there is a software running. However, what is running is the code analysis tool, not the assessed software. Besides, such techniques could be manually performed. Therefore, what some practitioners are naming static testing is an automated software review.

2.5.1 Software review

Software reviews are static verification techniques aiming to identify defects on software artifacts (*e.g.*, requirements document, design diagrams, source code). They are classified as static techniques as they do not require the execution of the software. The inspections stand out among the other review categories because they have a rigorous and well-defined process for defect identification [Fagan 1976].

The cost of defect fixing increases according to software development progress [Boehm 1984]. Therefore, the reviews are important activities as they can detect defects since initial phases of software development. Besides, researchers are showing that the reviews decrease the global development effort [Conradi et al. 1999], time [Gilb and Graham 1993], and cost [Laitenberger and Atkinson 1999]. It can also increase productivity as Russell [1991] reported: for each hour of inspection activity, there was a reduction of 33 hours of software maintenance.

The traditional inspection process developed by Fagan [1976] includes the phases overview, preparation, inspection, rework, and follow-up. Besides, the participants of inspections are moderators, authors, or inspector.

In the overview, an optional phase, the authors present the characteristics of the artifacts to the inspectors. Following, in the preparation phase, the inspectors study each of the artifacts to gain an understanding of it. Errors can be found, but not as many as will be found at the next phase. The inspection phase involves all team members. A reader reads the artifacts, and the other members can stop the reader and raise any issues they have discovered. Next, in the rework phase, the author of the artifact fixes the identified defects. Finally, the moderator assesses the quality of the fixed artifacts defining if a new round of inspections is needed.

Since then, the process of inspection has been evolving, and the significant change is that some authors do not consider the correction of a defect as part of the inspection [Macdonald and Miller 1995] [Kalinowski 2011].

2.5.1.1 The importance of reviews in the context of this thesis

Non-functional requirements represent different properties of a software system, and some NFRs are not observable during software runtime. For instance, it is unfeasible to observe the maintainability of the software by operating it. Consequently, it is essential to use static techniques to assess the software to enable the observation of NFRs (or a piece of the NFR) that are not visible at runtime.

Besides, there are other benefits of reviews as they can look for defects since initial phases of software development (Section 2.5.1). Therefore, they are essential to the verification of security and performance requirements.

2.5.2 Software testing

Software testing is a dynamic verification technique as it requires the software execution. Testing aims to identify software failures running a piece of software that contains faults. By knowing the failures, it is possible to identify their cause (fault) through the software debugging activity.

Software testing by itself does not increase software quality because it does not include the activity of fault fixing. However, it is reasonable to say that software testing contributes to software quality because the identification of failure is the first step to fix a fault [Delamaro et al. 2007].

Six phases constitute the software process: planning, design and implementation, environment set-up, execution, and incident reporting [ISO 29119-2 2013].

The planning phase includes the creation of a testing plan specifying the testing project scope, testing strategy, schedule, and necessary resources.

At the design and implementation phase, the test cases and test procedures are built based on artifacts (*e.g.*, requirements document, design diagrams) and the techniques previous defined at the test plan.

The environment set-up phase contemplates activities aiming to create and maintain the testing environment, for instance, defining the starting parameters of the software and the initial database data.

Next, the execution phase consists of the execution of the test procedures in the configured environment.

Finally, at the incident-reporting phase, the incidents are reported to key stakeholders. The identified issues are classified as incidents as they can be false positives. Besides, it is essential to note that the software testing process does not include a fault correction phase.

2.5.2.1 The importance of testing in the context of this thesis

Despite the effectiveness and efficiency of the review techniques, they have some weaknesses, and thus, they cannot replace software testing.

Reviews techniques are not suitable to identify some defect categories [Basili and Selby 1987] [Myers 1978] [Wood et al. 1997]. Besides, inspectors can make mistakes during the application of a review technique. Additionally, it is only possible to ensure a system is meeting an RNF through its running. For example, the inspection of source code can detect structures impacting the system response time, but the real response time of the system is observed only through the system execution.

Therefore, software testing and reviews are complementary techniques so that when they are combined, the possibility of defect detection increases.

2.5.3 Verification strategy

The software verification activities may consume a significant amount of project effort, budget, and time. Thus, if such activities are not conducted with some formality or criterion, then they may lose their effectiveness regarding defect detection capability, becoming a waste of resources. For example, using unsuitable methodologies for software testing results in an annual loss of USD 59.5 billion to American society [NIST 2002].

Consequently, a verification strategy is essential because it helps in the definition of the most suitable way to perform the verification activities so that such activities can perform efficiently and effectively.

This thesis adopts the concept of the strategy proposed by ISO-29119 [2013], using six dimensions to define a software verification strategy (Figure 4): type, level, technique, the definition of *done*, practice, automation level.

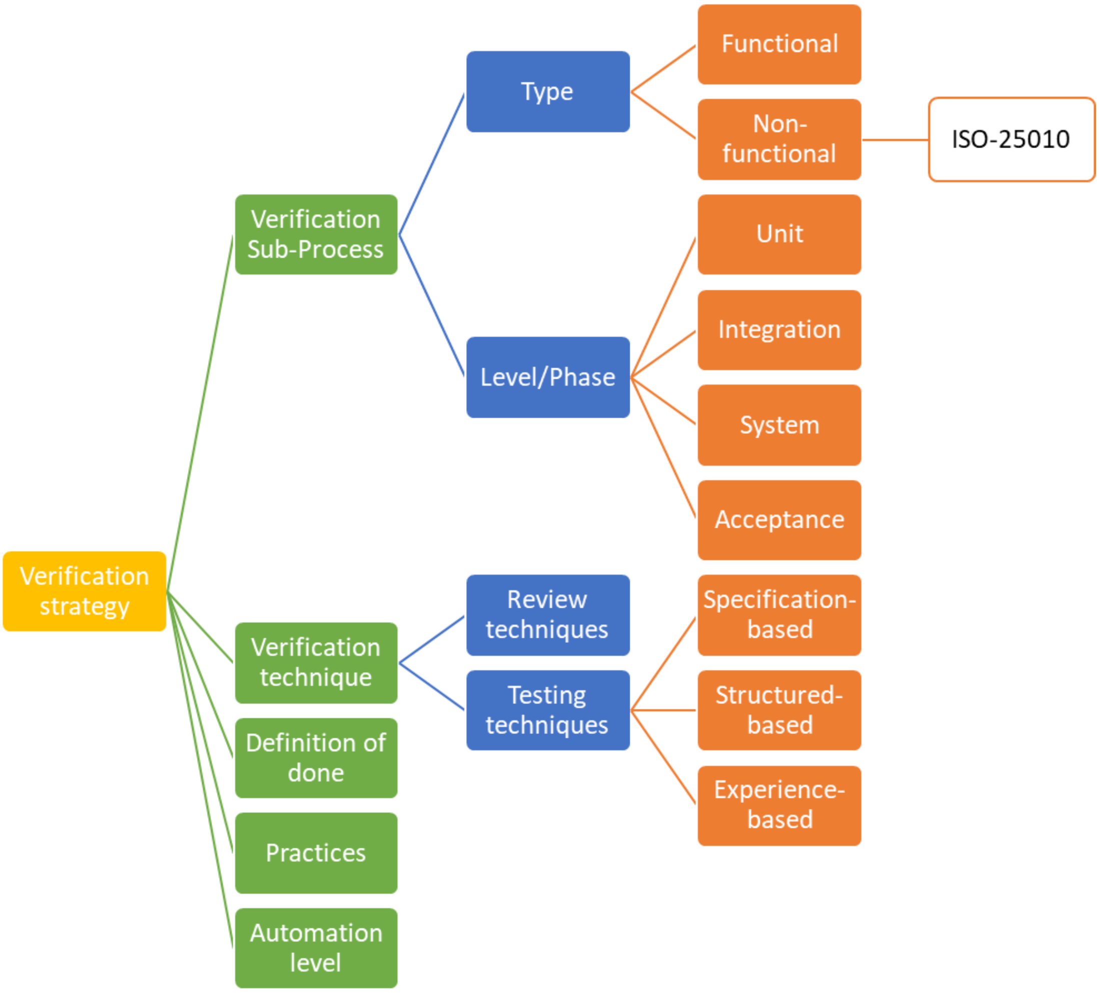


Figure 4 - Software verification strategy scheme – Adapted from ISO-29119 [2013]

- **Verification sub-process:** a sequence of activities used to perform a specific type or level/phase of verification.
    - **Type:** a set of verification activities for a specific quality attribute (non-functional requirements, including functionalities). The type is a sub-process because it is possible to define specific configurations of a strategy according to it. For instance, it is possible to define a strategy in

which the security verification should include the unit and integration testing levels, but the performance verification should include only the unit testing level.

  - **Level/phase:** it defines the granularity of the verification (unit, integration, system), and it is a specific instantiation of the test sub-process. For example, it is possible to define a strategy in which the unit test level should use structure-based techniques, but the system testing level should use specification-based techniques.

- **Verification technique:** activities, concepts, standards, or process used to (1) identify assessment conditions to a software asset, (2) derive corresponding test coverage items, and (3) derive or select test cases. Boundary value analysis, equivalence class partitioning are examples of testing techniques. Examples of review techniques are OORTs (Object-Oriented Reading Techniques) [Travassos et al. 2000] and the techniques proposed by Conte *et al.* [2007].
- **Definition of *done*:** it represents criteria defining that verification activities are finished. Usually, the definition of *done* criteria is the verification coverage, the number of defects identified at the last verification battery, project budget, or the schedule.
- **Practices:** it is the theoretical framework used for decision-making regarding verification activities. Informally, it is the philosophy that the organization will base verification strategies. Examples of types of practices are specification-based, risk-based, model-based, and mathematical based.
- **Automation level:** it defines whether a verification activity should be performed manually or automated. In the case of automation, it includes the identification of the supporting tools. Examples of supporting tools are verification management tools, testing execution tools, and static code analysis tools.

It is important to note that a strategy may be defined based on the Type or the Levels/Phases. If based on the Type, the team select the set of requirements (*e.g.*, usability, performance, security) should be assessed and then defines the verification Levels, Techniques, Definition of done, and Automation level for each verification Type (quality attributes such as security and performance). The second option is to define the verification Levels/Phases and then define the verification Type, Techniques, Definition of done, and Automation level. These two ways to define a verification strategy have the same result. However, the reasoning to get the strategy is distinct.

Besides, these strategy dimensions were used in the case study presented in this thesis aiming to characterize the practices employed on security and performance verification.

2.6 Security and Performance verification

The security verification aims to identify defects (faults and failures) that make the software violating security requirements, including the sub-requirements authenticity, availability, confidentiality, integrity, non-repudiation, and accountability. In the same vein, performance verification aims to identify defects that cause the software to not meet performance requirements (including resource consumption, time behavior, and scalability).

Therefore, security and performance verification is a Type of verification, and they can benefit from the already consolidated knowledge of the verification area. For instance, it is possible to use the same conceptual framework in a verification strategy (section 2.5.3) to evaluate a functional requirement and to evaluate the security (or performance) of a software system. Thus, it may be a mistake to discard all experience of software verification and to handle security and performance assessment as a topic unrelated to it.

It is important to emphasize that the content that composes the conceptual structure must be specific to the requirement in evaluation. What should be used is the conceptual structure and not its content. For example, the technique for test case selection named boundary value analysis aims to identify functional defects. Therefore, this technique should not be used to assess security or performance. However, it is necessary to use a specific technique to select test cases for security and performance.

2.6.1 Security verification challenges

The security requirement has some characteristics that make its verification painful. For example, it is hard to observe the security of a software system because it should be invisible to the end-users. It is easier to observe the insecurity of software because the failure consequences are visible to the end-users.

For instance, users do not know if their passwords are being stored encrypted or not. They only find out their passwords are stored in plain text after an invader publishes them.

Thus, the intrinsic characteristics of the security NFR impose some challenges to security verification [Ribeiro 2017].

- **Challenge 1: Deep technology expertise**

  Security is related to the solution domain because it is an NFR. Thus, the technologies employed in software building have a direct influence on their security. In this way, the effectiveness of security verification requires expertise in the set of technologies used to build it. For instance, when inspecting a source code of a system built with the C language, the inspector needs to know the details of the language to identify code patterns that can lead to buffer overflow failures. However, this concern does not have the same priority in languages that do not directly manipulate memory, such as Java or Python.

  Since the set of technologies used to build software can be numerous, the need for deep expertise of each of the technologies is a challenge for security verification activities.

- **Challenge 2: Explicit invader intention to cause a failure**

  Every day, users hope the software correctly works so that they can take advantage of its functionalities. However, the invader has the explicit intention of causing failures. Additionally, the attacker usually has computer knowledge and uses this knowledge as an advantage. For instance, a user operates the software using its default interface. However, the invaders look for alternative ways to interact with the system, such as using hidden web form fields, disabling browser JavaScript validations, or using malicious tools.

  Therefore, the security team should be updated regarding invasion methods to protect the system against them.

- **Challenge 3: Lack of security requirements specification**

  The lack of security requirements description or the existence of imprecise descriptions has a negative influence on verification activities. The requirements are the oracle for the verification activities. Therefore, without an oracle, it is not possible to asses if the software behavior is correct.

- **Challenge 4: Security verification coverage**

  The verification coverage is regarding the parts of the software should be embraced by verification and the intensity of it. The challenge is to know what the suitable coverage is so that there is a balance among employed resources and the defects remaining.

  In this respect, developers have a disadvantage if compared with the invaders. The developers need to find as many defects as possible, and the invaders only need to find a vulnerability (defect) to misuse the system and its information.

- **Challenge 5: Lack of secure development culture**

  Software development organizations do not have secure development as a culture. They are probably influenced by the past, when systems were not made available over the internet, preventing access by a large number of malicious users. The security verification is neglected, or it is only performed when the software is in production, jeopardizing the identification of the defects.

2.6.2 Performance verification challenges

It was possible to identify two challenges of performance verification. The first one regarding the verification environment and the second is related to intensive software demand.

- **Challenge 1: set-up an isolated verification environment**

  Regarding the environment, the organizations' first challenge is to replicate the configuration of the production environment in the verification environment. The servers running the system at production time are often powerful and consequently expensive. Thus, it becomes financially unfeasible to replicate such configuration in the verification environment.

  Besides, the performance verification often uses the standard infrastructure of the organization (*e.g.*, network, switches, routers). It can impact the results of performance verification because a device may be overloaded due to requests from other users of the infrastructure.

- **Challenge 2: reproduce a significant amount of requests**

  One of the performance activities goals is to assess software behavior when it is massively being used, for example, when numerous users are operating the software. Performing such kind of assessment, the verification team uses a computer (or a few) to simulate requests against the system (agents). Each agent can simulate tens, hundreds, or thousands of users operating the software. However, the performance discrepancies found by this kind of testing can be results of computational resources limitations of the agents and not performance failures of the system under evaluation.

3 Testing Non-Functional Requirements: A Body of Knowledge

This chapter presents a body of knowledge, including the most relevant non-functional requirements and the testing approaches to assess them. The knowledge used to build the body of knowledge arose from the findings of two structured literature reviews. The body of knowledge represents a significant achievement of this research as it was used to direct the research objectives.

3.1 The methodology supporting this chapter

The findings presented in this chapter result from two structured literature reviews described in the following.

3.1.1 A literature review on the relevant non-functional requirements

LR1 followed the protocol presented in Appendix A. It aimed to identify the relevant NFRs, searching for secondary studies or surveys presenting NFRs mentioned as relevant by practitioners. It was carried out in March 2015, retrieving papers from 1996 to 2015, and driven by the following research question:

- **What are the most relevant non-functional requirements, according to software practitioners?**

Aiming to answer this question, it was built a search string with two parts — the first one to filter systematic reviews or survey researches; the second part to limit the search for non-functional requirements.

("systematic review" OR "systematic literature review" OR "systematic mapping" OR "systematic investigation" OR "systematic analysis" OR "mapping study" OR "structured literature review" OR "evidence-based literature review" OR "survey" OR "review of studies" OR "structured review" OR "systematic review" OR "literature review" OR "systematic literature review" OR "literature analysis" OR "meta-analysis" OR "analysis of research" OR "empirical body of knowledge" OR "overview of existing research" OR "body of published knowledge")

AND

("non-functional requirements" OR "non-functional software requirement" OR "non-behavioral requirement" OR "non-functional property" OR "quality attribute" OR "quality requirement" OR "software characteristic")

The search string was applied to the search engine Scopus so that 266 papers could be found. After the search string execution, the primary author reads the title and abstract of each paper, classifying them on Included or Excluded using the following criteria:

- **Inclusion criteria**
  - The paper must present a systematic literature review, a survey or a similar study; AND
  - The paper must Identify relevant non-functional requirements; AND
  - The paper must express practitioners’ opinions, OR practitioners must.
- **Exclusion criteria**
  - The paper is not available; AND
  - The paper presents an already included study (duplicity).

Then, another researcher of this thesis analyzed the excluded papers set and reclassified them on Included or kept it out. Table 3 shows the number of papers of LR1. It is important to note that there is a paper manually included because the search engine was not indexing correctly.

Table 3 - Amount of LR1 papers

Papers found	Excluded	Included	Manual included	Total included
266	252	14	1	15

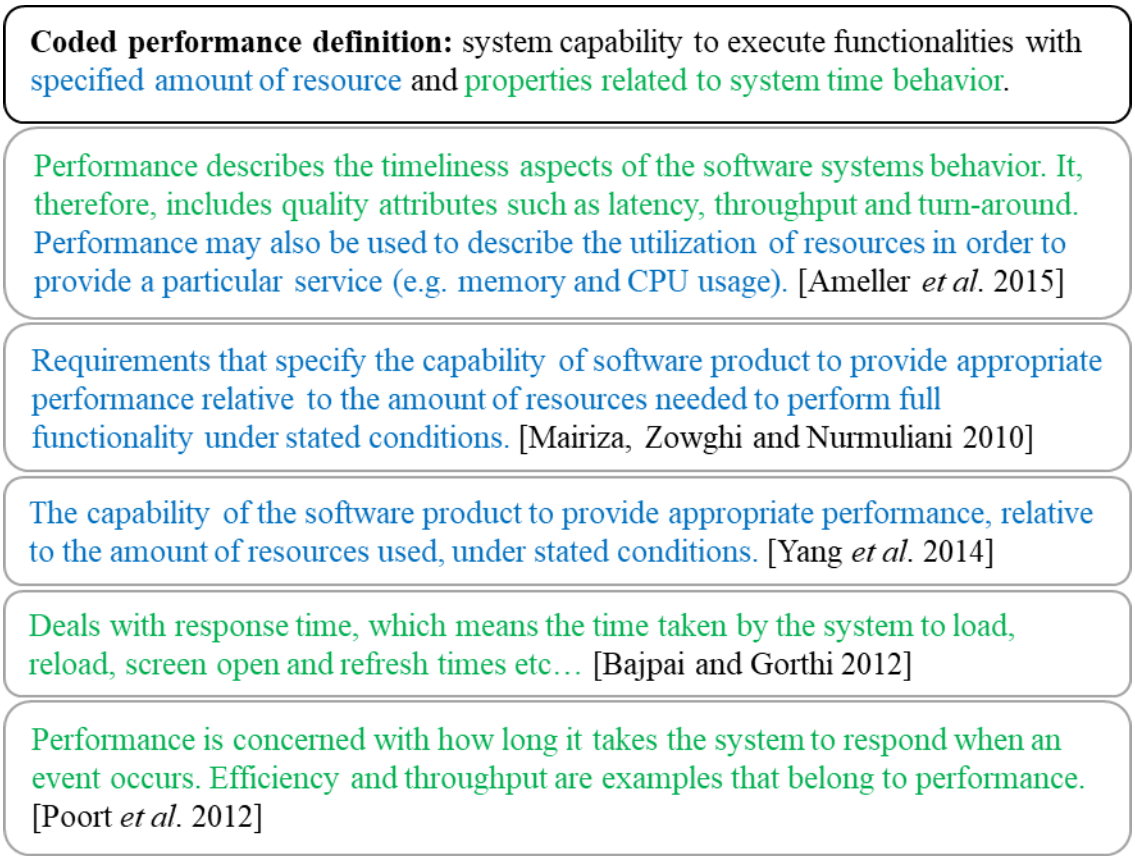
The researchers analyzed the 15 included papers and extracted the following information:

- **Reference information:** it aims to identify the paper by title, author, and publisher;
- **Abstract:** it aims to contextualize the research when querying the form;
- **Study type:** it identifies the type of study, *e.g.*, systematic literature review, survey, along with others;
- **System domain/type:** it identifies the system type or domain in which the research has been done;
- **Non-functional requirements:** it identifies the NFRs presented in the paper and their description when presented.

At this point, we had the extraction form of each NFR. Analyzing extracted information was possible to realize that some NFRs did not have a description. Thus, for the sake of comprehensibility, the group of NFRs without description was not considered at this point.

The next followed step relates to the understanding of each NFR to organize them into a body of knowledge. However, it was possible to identify a lack of agreement regarding NFRs' names and descriptions. Thus, to organize all described NFRs, we performed *open coding*, as described in Grounded Theory [Corbin and Strauss 1998].

Figure 5 shows an example of the resulting code on performance definition where the first box is the final performance definition extracted from subject papers. For instance, the highlighted text in blue associate performance on resource consumption and the text of green color to time behavior.



Coded performance definition: system capability to execute functionalities with specified amount of resource and properties related to system time behavior.

Performance describes the timeliness aspects of the software systems behavior. It, therefore, includes quality attributes such as latency, throughput and turn-around. Performance may also be used to describe the utilization of resources in order to provide a particular service (e.g. memory and CPU usage). [Ameller *et al*. 2015]

Requirements that specify the capability of software product to provide appropriate performance relative to the amount of resources needed to perform full functionality under stated conditions. [Mairiza, Zowghi and Nurmuliani 2010]

The capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions. [Yang *et al*. 2014]

Deals with response time, which means the time taken by the system to load, reload, screen open and refresh times etc… [Bajpai and Gorthi 2012]

Performance is concerned with how long it takes the system to respond when an event occurs. Efficiency and throughput are examples that belong to performance. [Poort *et al*. 2012]

Figure 5 - Open coding example

Section 3.6 presents the body of knowledge (NRF-BoK) organized as a result of this analysis.

The coding process allowed identifying a hierarchical structure of NFRs. Figure 6 shows the structure in which the class NFR represents high-level abstract system properties such as Usability, Security, and Performance. These properties are perceived through a set of *Sub_NFR*, which are also NFR but, they represent more specific system properties such as Navigability (Usability), Confidentiality (Security), or Resource Consumption (Performance). Moreover, some NFRs may own *Operationalization,* which are features that must be present on the system for it meets the NFR. For instance, the usage of an image compression algorithm is one operationalization of Resource Consumption.

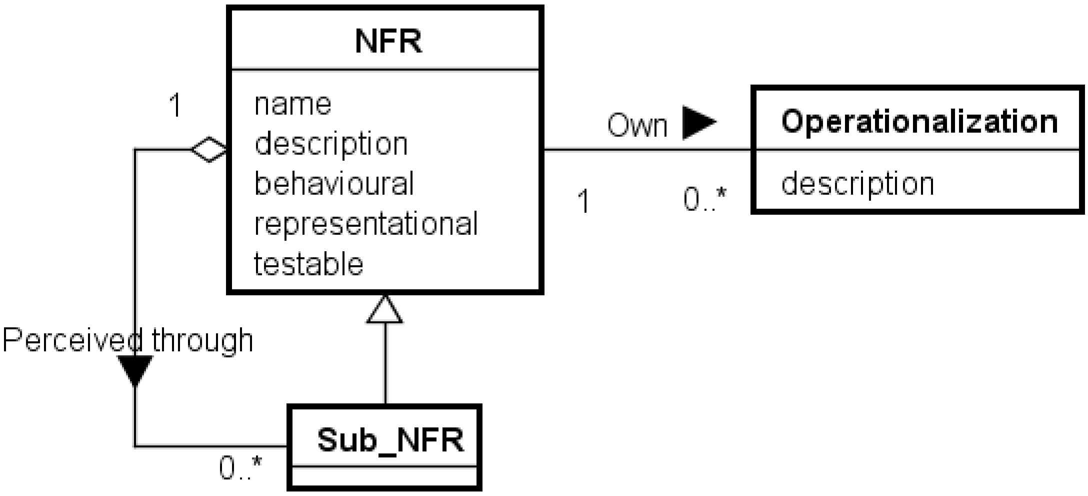


Figure 6 - Hierarchical structure of NFRs

3.1.2 A literature review on the non-functional testing approaches

The second structured literature review (LR2) followed the protocol presented in Appendix B. It aims to identify proposed software testing approaches concerned with NFRs and their testing covering. In this context, testing covering is regarding software testing process phases, levels, and techniques of the proposed approaches. Unlike LR1, LR2 does not look at other literature reviews because previous *ad-hoc* searches do not retrieve that kind of study concerning testing approaches for NFRs. LR2 was performed in March 2016, naturally retrieving papers from 1991 to 2015, and driven by the following research question:

- **What are the software testing approaches used to test non-functional requirements?**

Two parts search string was organized and submitted to the Scopus search engine. The first section of the search string aims to limit the results to software testing approaches, and the second one restricts the search to non-functional requirements.

("software test design" OR "software test suite" OR "software test" OR "software testing" OR "system test design" OR "system test suite" OR "system test" OR "system testing" OR "middleware test" OR "middleware testing" OR "property based software test" OR "property based software testing" OR "fault detection" OR "failure detection" OR "GUI test" OR "Graphical User Interfaces test" OR "test set" OR "non-functional

testing" OR "model based testing" OR "test case" OR "testing infrastructure" OR "testing approach" OR "testing environment")
AND
("non-functional requirements" OR "non-functional software requirement" OR "non-behavioral requirement" OR "non-functional property" OR "quality attribute" OR "quality requirement" OR "software characteristic")

The filtering process followed a similar procedure as in LR1. The inclusion/exclusion are:

- **Inclusion criteria**
  - The paper should present a software testing procedure, technique, or any other type of proposal about non-functional requirements software testing.
- **Exclusion criteria**
  - The paper is not available; AND
  - The paper presents an already included study (duplicity).

Table 4 shows the number of papers of LR2. There were three papers manually included because they were not directly available through the Scopus search engine.

Table 4 - Amount of LR2 papers

Papers found	Excluded	Included	Manual included	Total included
331	287	44	3	47

The 47 papers were analyzed using an extraction form with the following fields:

- **Reference Information:** it aims to identify the paper by title, author, and publisher;
- **Abstract:** it aims to provide an overall idea of the paper subject;
- **System Domain/Type:** it indicates whether the approach is proposed to specific software domain or type, *e.g.*, embedded systems, telecommunication systems;
- **Test Phase:** the coverage of the testing approach regarding testing process phases: Planning, Design, Implementation, Execution, and Analysis;
- **Test Level:** testing granularity, with the options Unit, Integration, System, Acceptance, Not Informed, and Not Applied;

- **Test Technique:** with the options Structural, Functional, Fault Based, Not Informed, and Not Applied;
- **Evaluation:** it represents how the software testing approach has been evaluated, *e.g.*, proof of concept, experiment, case study, simulation, not applied, and not informed. Evaluation values emerged from the subject papers;
- **Non-Functional Requirements Considered:** it represents the list of NFRs considered by the software testing approach with their description.

After data extraction and analysis, the information regarding software testing approaches was included in the NFR-BoK. Thus, besides including the relevant non-functional requirements and their characterization, it also contains information about which testing techniques are suitable to assess each of the NFRs.

3.2 Overview of the main findings and implications

Table 5 presents the main findings of two literature reviews and how they influence the scope of this thesis.

Table 5 - Initial findings and their implications

Observed fact	Implication
Identification of 224 NFRs	(1) Inadequacy to handle all these requirements. (2) Setting the research focus on the most relevant: security and performance.
There are non-testable NFRs	(1) Software testing is not enough to assess NFRs. (2) Setting the research scope to software verification (testing and reviews).
Identification of NFRs not covered by testing approaches	(1) Identification of the need for new approaches to assessing NFRs.
Testing approaches do not cover all testing dimensions (phases and levels)	(1) Identification of the need for the evolution of proposed approaches to assess NFRs.

The significant amount of NFRs identified as relevant threw the warning regarding the inadequacy to address all these requirements in the same research. In this way, this research was directed to the two most quoted NFRs in the papers included in the literature review (Section 3.3). Therefore, this thesis focused on security and performance requirements.

Additionally, the initial findings showed that there are no approaches to evaluate some NFRs, and there are incomplete approaches as they do not cover all testing phases and levels. Such findings were essential to the initial hypothesis about the importance and the need for researches related to NFR verification.

3.3 The relevant non-functional requirements: a literature review

The initial overall objective of this research was to assist practitioners in assessing NFRs. In this way, the first logical step was the identification of the most relevant NFRs for these professionals. Thus, we performed a structured literature review searching for works that had identified which NFRs were relevant to the software industry. Appendix B presents the protocol of this literature review.

The literature review aimed to identify other literature reviews or surveys that have collected the perception of practitioners about which NFRs are relevant to a software system, having the following research question:

- **What are the most relevant non-functional requirements, according to software practitioners?**

It was possible to include 16 papers and identify an amount of 224 different names of NFRs. However, there were inconsistencies within the names of some NFRs and the software property they represent — NFRs with different names having the same definition (synonyms); and NFRs having the same name but presenting a discrepant definition. For instance, *fault tolerance* [Sucipto and Wahono 2015] and *robustness* [Bajpai and Gorthi 2012] have an equivalent definition, representing a single software property. On the other hand, the name *performance* is used to represent software properties related to *resource consumption* [Ameller et al. 2016] or *time behavior* [Montagud et al. 2012].

Thus, merely counting the number of NFRs aggregating them by name would be a careless attitude. For example, even if two papers present the performance as a relevant requirement, it is not possible to conclude that performance has two citations. Maybe one of the papers is talking about resource consumption and the other paper about time behavior.

Within this scenario, it was necessary to analyze the description of every identified NFRs and exclude NFRs without description due to the risk they represent. Therefore, 87 NFRs having a description are considered. Next, the analysis of NFRs descriptions allows the aggregation of some synonyms, resulting in 60 unique NFRs (Table 6).

Table 6 - Amount of identified NFRs

Identified NFRs	224
NFRs **without** description	137
NFRs **with** description	87
Unique NFRs	60

Table 7 presents the set of unique NFRs hierarchically organized by the number of papers citing them.

Table 7 - The set of relevant non-functional requirements

NFR	Sub-NFR	Description
Security	System capability to protects data or resources from people or other systems that do not have access permission, providing correct access level from people or other systems that have access permission. It means that the system must have the capability to continue providing essential services even under attack.	
	Confidentiality	System capability to allow only users with an appropriate access level to access resources (data, print, services, etc.), including data traffic.
	Auditability	The availability of auditing capability of service invocations that can be traced to specific users for logging and repudiation.
	Vulnerability	System capability to prevent attacks. It

		means that the system must have the capability to continue providing essential services even under attack.
Performance	System capacity to provide functionalities using a specified amount of resources and time	
	Scalability	System capability to maintain a required performance when it is used in increasing workload.
	Resource consumption	System capacity to provide functionalities using a specified amount of resources.
	Timeliness	System capability to provide data and functionalities "on time." It means the system's capability to provide data and functionalities within the time constraints specified by the destination organization.
Usability	It comprehends how users can easily operate and control software systems.	
	Understandability	System capacity to provide clear ways to users to understand the goals and data provided.
	Accessibility	Capability to be used by different people with different capabilities, including people with particular necessity.
	Satisfaction	User satisfaction in a specified context of use.
	Learnability	System capability to enable the user to learn how to use it to achieve specified system goals.
	Organization	The way that system works with data organization, visual settings or typographical features (color, text, font, images), and the harmonious combinations of these various components.
	Attractiveness	System capability to be attractive for users.
Reliability	Software capability to operate without failure under specified conditions during a given period and the system capability to deal	

	with failures.	
	Availability	Represents how much time the system is operating on the correct state (functional and non-functional) delivering its services.
	Recoverability	System capacity to return to the operating state after some damage (*e.g.*, network failure or system failure).
	Fault tolerance	System capability to adapt to keep operational and performance level in the presence of environment and software faults.
Maintainability	Represents the capability of the system to be effectively and efficiently modified by the maintenance team, specialized support staff, business, or operational staff, or end-users. The modifications can be changing functionality, bug fixing, system improvement.	
	Analyzability	System capacity to provide ways of discovering deficiencies, causes of failures, or identify parts to be modified.
	Stability	System capability to avoid unexpected effects from system modifications.
Portability	Related to a software infrastructure configuration defining settings in which software should be executed.	
	Adaptability	System capability to adapt itself for different specified environments, situations, or use cases without applying actions provided for this purpose.
Testability	Defines how easy software allows being tested to expose its defects.	
Functional suitability	The capacity of the system to provide specified functions to enable users to achieve specified goals with accuracy and completeness in a specified context.	
	Accuracy	Degree of the proximity of a measured or calculated result to its real result.
	Completeness	Software ability to provide adequate functions and data (breadth, depth, and scope) to do a specified task.
Compatibility	System capability to be operable sharing the same environment with	

	other systems or applications exchanging information with other products, systems, or components.	
	Interoperability	System capability to interact with another system.
Reputation	The opinion of consumers toward a system.	
Data quality	It is related to the quality of input and output data regarding the accuracy, timeliness, data policy (how system behave concerning the data it operates on or produces when it fails), and data integrity.	
	Data traceability	The extent to which data are well-documented, verifiable, and easily attributed to a source.
	Data flexibility	The extent to which data are expandable, adaptable, and easily applied to other needs.
	Data trustability	The extent to which data and their sources are accepted as correct.
	Data expiration	The extent to which the date until which data remain current is known.
	Data novelty	The extent to which data obtained from the system influence knowledge and new decisions.
	Information source	The extent to which information about the author/owner of a web portal is delivered to the data consumers.
	Data specialization	Degree of specificity of data/information contained in and delivered by the system.
	Value-added	The extent to which data are beneficial and provide advantages from their use.
	Data Validity	The extent to which users can judge and comprehend data delivered by the system.
	Data reputation	The extent to which data are trusted or highly regarded in terms of their source or content.
Price	It is the fee that the consumer is expected to pay for a system.	
Collaboration	System capacity to support collaborative work between different users of the system.	
Invisibility	System capability to decrease user sensation of explicit use of	

	computer and exchanging the perception that objects or devices provide services or some kind of “intelligence.”	
Context sensitivity	System capability to collect and use information from the environment where it is being used to adapt dynamically to offered services according to the environment where it is being used, respecting its limitations.	
Certification	Confirmation of specific characteristics of the system such as affiliation, signature, specification, policy, standard, law, regulation.	
	Standards compliance	The specification of what are the standards that the system should be following.
	Jurisdiction	Determines countries/territories for which a service complies with national/territorial regulations.
Configuration management	System capability to provide information about the current configuration and the changes applied to the configuration to every stakeholder.	
	Service versioning	System capability to provide documentation about the nature of changes for a system such as an interface update, backward compatibility, added or removed functionalities, or fixed bugs
Documentation	It is related to the quantity and usefulness of documents.	
Currency	The extent to which the system provides non-obsolete data.	
Customer support	System capability to provide some form of support, e.g., provide support through text, e-mail, telephone.	
Server location	Defines a place in physical space that a system is installed.	
Service omnipresence	System capability to provide ways to allow users to move around with the sensation of carrying computing service with them.	
Function composition	System capability to join services in a way that creates a service required by the user.	

3.3.1 Determining the testability of a non-functional requirement

The improvement of the knowledge about NFRs brought by the literature review shows that some NFRs are not observable at runtime, and then they cannot be assessed through software testing. Thus, NFRs can be classified as *behavioral* or *representational* [Broy 2015], indicating if they can be assessed through software testing:

- **Behavioral NFRs:** these requirements represent behaviors that the system exposes at runtime, during their operation. For instance, "the system services must respond to a request within 1 second (max)." Note that the response time can be observed at run time, so it is a testable NFR.
- **Representational NFRs:** these requirements represent a syntactic or technical system property. The observation of such properties does not require the system execution, and it can be impossible to observe them at runtime. For instance, "The system must be developed using the MySQL database." Note that it may be impossible to assess if the system is using the MySQL database at runtime because there are techniques to hide the technologies used in software development. Therefore, this NFR is non-testable, but it can be assessed through a static technique such as inspection.

3.4 Non-functional testing approaches: a literature review

After identifying the most relevant NFRs, another structured literature review was performed to identify software testing approaches used to assess non-functional requirements. Thus, combining previous literature review results (Section 3.3) with these new results, it is possible to identify what are the relevant NFRs that do not have suitable testing techniques to assess them.

It is important to note that the need to include static verification techniques (reviews) within the scope of this thesis was perceived only after the execution of this literature review. Thus, it focused only on testing approaches.

Appendix C presents its protocol. Besides, it uses the following research question:

- **What are the software testing approaches used to test non-functional requirements?**

It was possible to identify 47 software testing approaches to assess non-functional requirements. Table 8 presents the testing approaches indicating the publication year, system type, type of study used to evaluate, and the NFRs covered by the approach.

Table 8 - Software testing approaches to assess NFRs

#	Year	Domain/System type	Evaluation	Non-functional requirements
1	1999	Real-time distributed	Proof of Concept	Timeliness
2	2001	Web application	Proof of Concept	General Approach
3	2004	Service-Oriented	Proof of Concept	Performance: Throughput, Reliability
4	2005	Distributed component	Proof of Concept	Performance: Latency, Throughput, Scalability
5	2007	Component-based	Experiment	Performance: Time, Efficiency
6	2007	Component-based	Proof of Concept	General Approach
7	2007	General	Case Study	Performance: Resource Consumption, Time, Reliability: Recoverability, Interoperability
8	2008	General	Proof of Concept	Performance: Timeliness, Process capacity, Resource consumption, Reliability
9	2008	Web Services	Proof of Concept	Reliability: Fault Tolerance
10	2008	Web Services	Proof of Concept	General Approach
11	2008	General	Proof of Concept	General Approach
12	2009	General	Not applied	Performance: Execution Time, Quality of Service Security: Vulnerability Usability: Navigability Safety
13	2009	Component-based	Experiment	Performance: Time, Efficiency
14	2009	Embedded real-time	Proof of Concept	Performance: Timeliness: Response Time
15	2009	Distributed component	Experiment	Performance: Response Time, Processing Time
16	2009	Embedded	Case Study	General Approach
17	2009	Component-based	Proof of Concept	Reliability
18	2009	Web application	Proof of Concept	Performance: Scalability, Timeliness, Usability: Navigability

19	2009	Distributed component	Case Study	Performance: Response Time
20	2010	General	Proof of Concept	General Approach
21	2010	General	Proof of Concept	General Approach
22	2010	Web application	Proof of Concept	Security: Vulnerability
23	2010	General	Experiment	General Approach
24	2010	Object-Oriented	Not informed	General Approach
25	2010	Distributed	Not applied	Performance: Timeliness, Scalability, Security: Vulnerability, Correctness: Avoid deadlock, Checking conformance, Reliability
26	2011	General	Experiment	Performance: Scalability
27	2011	Embedded	Proof of Concept	Performance: Response Time
28	2011	Reactive	Proof of Concept	Reliability: Fault Tolerance
29	2011	Embedded	Proof of Concept	Reliability: Fault Tolerance
30	2012	Not Specified	Case Study	Reliability: Fault Tolerance
31	2012	Rich Internet Application	Proof of Concept	Usability: Accessibility
32	2013	General	Proof of Concept	Performance: Response Time, Availability Usability: Organization, Accessibility: Interactive
33	2013	Service-Oriented	Experiment	Security: Confidentiality, Integrity, Authenticity, Repudiation (non-repudiation) Reliability: Fault Tolerance, Availability
34	2013	Embedded real-time	Experiment	Performance: Resource Consumption
35	2013	Elastic Computing	Simulation	Elasticity: Plasticity, Resonance
36	2013	General	Proof of Concept	Performance: Response Time, Throughput
37	2013	Using ASTERIX protocol	Experiment	General Approach
38	2013	Interactive	Proof of Concept	Usability: Effectiveness, Efficiency
39	2013	Using the HTTP protocol	Case Study	Performance: Response Time
40	2013	DB2 Database	Proof of Concept	Performance: Response Time, Execution time
41	2014	General	Proof of Concept	Security
42	2014	Real-time	Case Study	General Approach
43	2014	Embedded	Proof of Concept	Performance: Resource Consumption, Timeliness, Reliability: Fault Tolerance, Security: Vulnerability
44	2014	General	Proof of Concept	Performance: workload, timeliness, think time, Rampup time, Startup delay

				Reliability
45	2015	Embedded	Not informed	Performance: Energy consumption
46	2015	Web application	Proof of Concept	Performance: Response Time, Throughput, Simulated workload Security: vulnerabilities
47	2015	General	Case Study	General Approach

Besides identifying the testing approaches, some of their characteristics were analyzed in order to improve understanding of such proposals. First, it was analyzed the *domain/system type* of each testing approach. It is important because if a testing approach is proposed to assess information systems, then the use of it to assess real-time systems is risky. Figure 7 shows the number of approaches found for each domain/system type. It is possible to observe that most of the approaches are generic, as they are not proposed for a particular type of system. It is important to note that some domains/system type could be grouped. For example, systems *using HTTP protocol* are possibly a *web application.* However, it was decided to maintain the original classification presented in the articles to avoid misunderstandings in future evaluations of the proposed approaches.

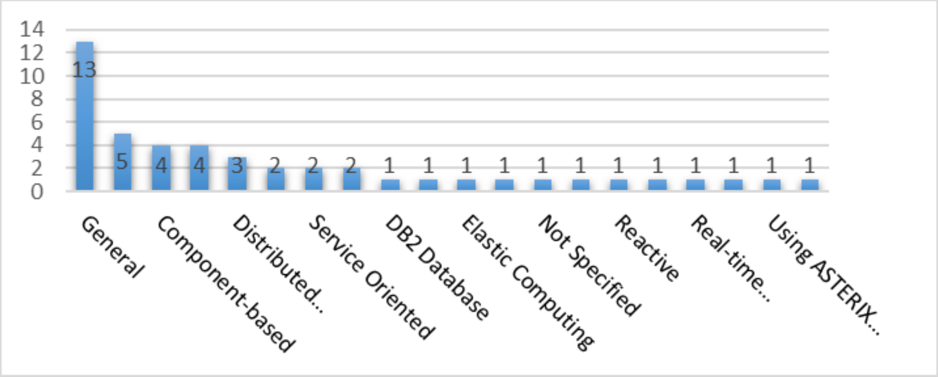


Figure 7 - NFR testing approaches vs. domain/system type

Most of the proposed testing techniques were evaluated through a *proof of concept*. It is a warning as that kind of study does not have strong confidence. In practice, it means the use of such testing techniques in real scenarios is risky, and their benefits are not well-understood. Figure 8 shows the number of techniques and how they were evaluated. It is also important to note that there are four testing approaches with the type of evaluation study as *not informed*.

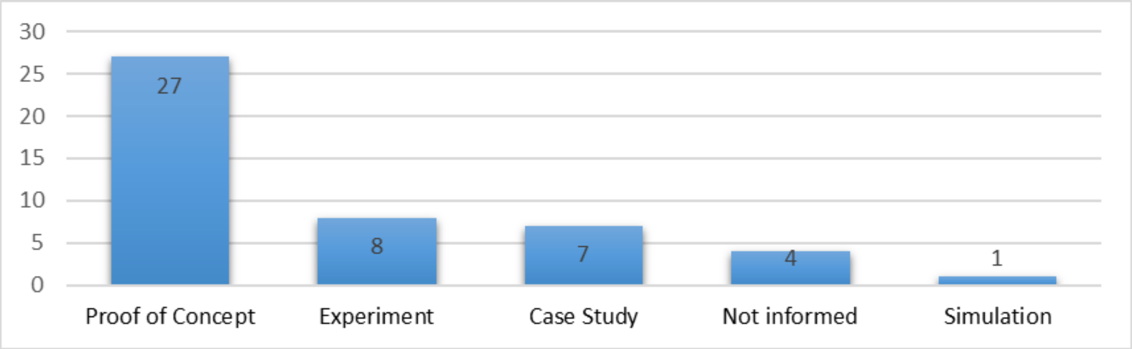


Figure 8 - NFR testing approaches vs. type of evaluation

The effective evaluation of a software system should follow a systematic process, be able to exercise the software in different stages of the development lifecycle, and use appropriate techniques. Thus, the testing approaches were analyzed concerning the phases of the test process, the levels of tests, and the type of technique used.

Regarding the phases of the software testing process (Figure 9), most of the testing approaches target the design and implementation phases, some of them target the execution phase, and a few targets the implementation phase. Besides, there are not testing approaches targeting the planning phase. It is worrisome because testing activities consume most of the software development effort so that the lack of planning can decrease the failure detection rate, resulting in wastage of the resources employed.

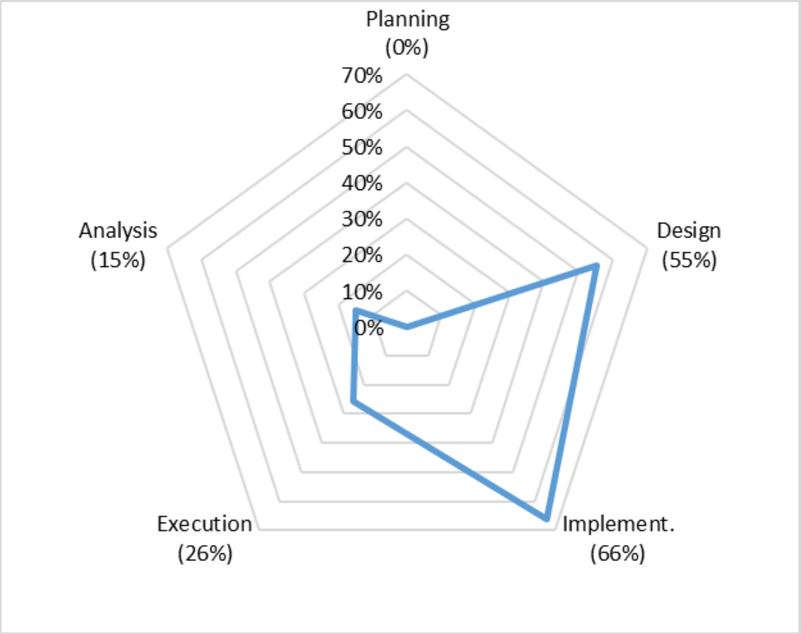


Figure 9 - NFR testing approaches vs. testing process phase

Figure 10 shows the software testing levels covered by identified approaches. Most of them are proposed to solve aspects related to the testing *system level*. It was not possible to identify researches indicating which testing level is suitable for evaluating NFRs. However, each test level is more effective in finding some specific types of failures.

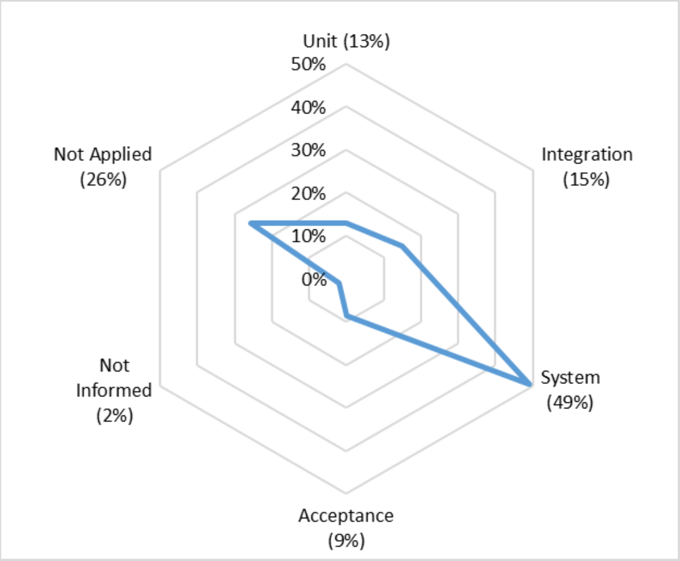


Figure 10 - NFR testing approaches vs. testing level

Regarding the testing technique of the identified approaches, most of them are *requirement-based* (Figure 11). It means the approaches use the software requirements as the base to plan and build software testing. This result is consistent with previous results (*testing level*) because *system-level testing* often uses *requirement-based* techniques. However, this may indicate risk in the application of these approaches since NFRs are rarely adequately described [Matoussi and Laleau 2008].

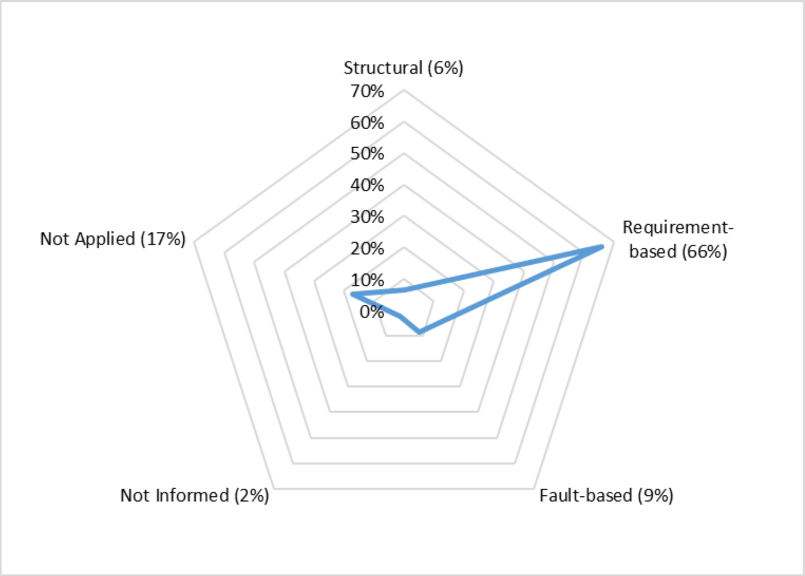


Figure 11 - NFR testing approaches vs. testing technique

3.5 Combining the results of relevant NFRs and testing approaches literature reviews

After finishing the literature reviews, it was possible to make a combination of their findings. The result is the mapping of what are the most relevant NFRs without corresponding software testing techniques. Figure 12 summarizes this matching, where 13 of 87 classified NFRs miss a software testing approach. The first number inside the brackets is the number of papers referencing the NFR, and the second one is the amount of software testing approaches dealing with the NFR.

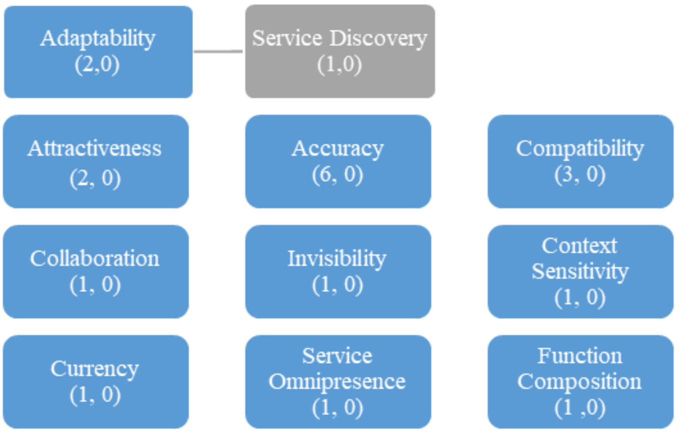


Figure 12 - Relevant NFRs without software approaches

After, the combination was done in the opposite direction to identify the software testing approaches dealing with less frequent NFRs. Figure 13 summarizes this result in which seven approaches are proposed to less relevant NFRs. It is important to note that there are not testing approaches covering *correctness* and *elasticity*, but they are displayed to keep the hierarchical structure of the NFRs.

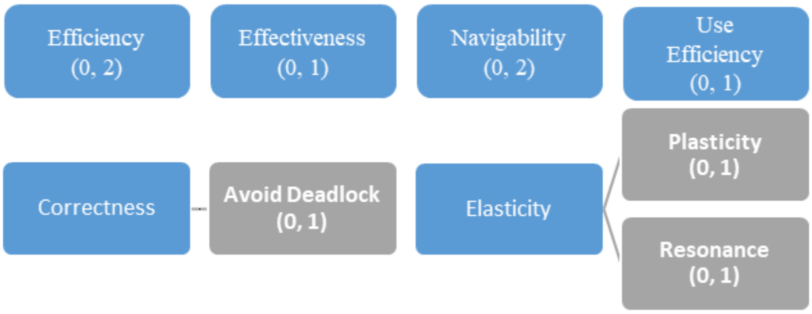
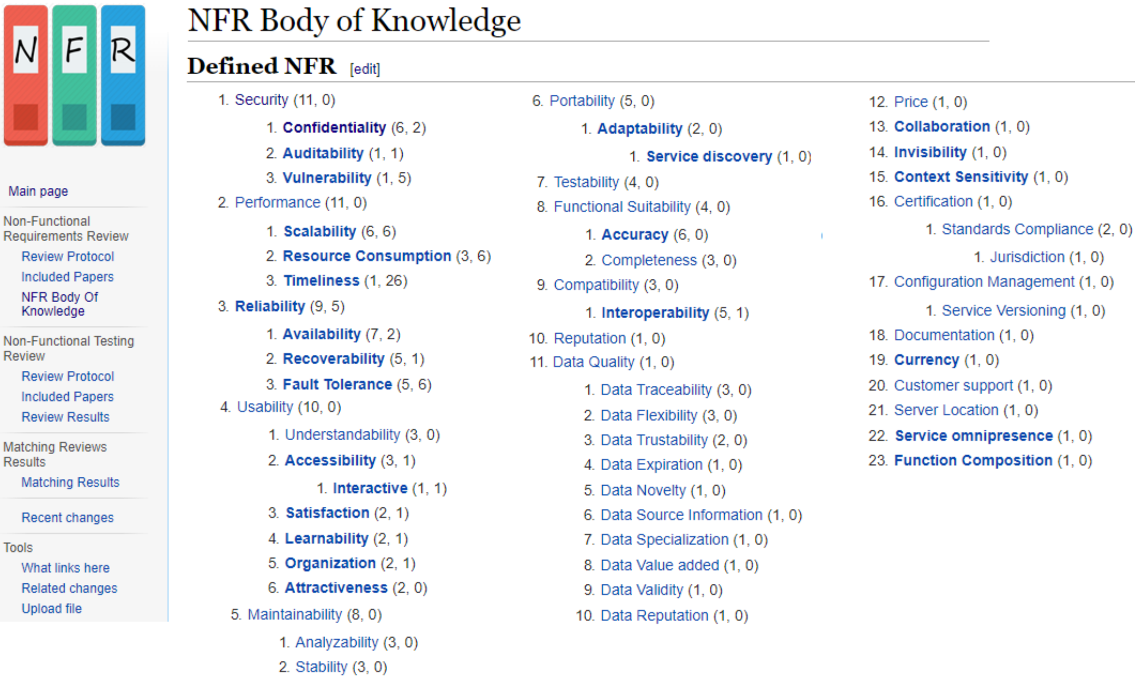


Figure 13 - Testing approaches to assess less relevant NFRs

There may be two explanations for explaining the existence of testing approaches to assess NFRs of little relevance: First, the lack of clear research agenda, causing researchers to focus on NFRs according to their self-interests. It means a waste of resources, so that effort was expended in the creation of testing approaches covering less relevant (low interest) NFRs. Second, these approaches could be proposed for a specific context in which these less relevant NFRs are essential and needed to be verified. However, it is not possible to claim any of them without more comprehensive information from the researchers.

3.6 The body of knowledge of relevant non-functional requirements and their testing techniques

A body of knowledge[4] (NFR-BoK) consolidates the results of the two previous literature reviews. The NFR-BoK presents detailed information about identified NFRs, including the testing approaches that can be used to assess each of them. It is organized as a wiki to facilitate user navigation. Figure 14 presents the relevant NFRs so that by clicking on that the user can view a page of detailed information. The first numerical character inside the brackets represents the number of papers that identify the NFR as relevant, and the second represents the number of testing approaches to assess it. For instance, six papers cite *confidentiality* as a relevant NFR, and there are two testing approaches to assess it.



NFR Body of Knowledge

Defined NFR [edit]

1. Security (11, 0)
    1. **Confidentiality** (6, 2)
    2. **Auditability** (1, 1)
    3. **Vulnerability** (1, 5)
2. Performance (11, 0)
    1. **Scalability** (6, 6)
    2. **Resource Consumption** (3, 6)
    3. **Timeliness** (1, 26)
3. **Reliability** (9, 5)
    1. **Availability** (7, 2)
    2. **Recoverability** (5, 1)
    3. **Fault Tolerance** (5, 6)
4. Usability (10, 0)
    1. Understandability (3, 0)
    2. **Accessibility** (3, 1)
        1. **Interactive** (1, 1)
    3. **Satisfaction** (2, 1)
    4. **Learnability** (2, 1)
    5. **Organization** (2, 1)
    6. **Attractiveness** (2, 0)
5. Maintainability (8, 0)
    1. Analyzability (3, 0)
    2. Stability (3, 0)
6. Portability (5, 0)
    1. **Adaptability** (2, 0)
        1. **Service discovery** (1, 0)
7. Testability (4, 0)
8. Functional Suitability (4, 0)
    1. **Accuracy** (6, 0)
    2. Completeness (3, 0)
9. Compatibility (3, 0)
    1. **Interoperability** (5, 1)
10. Reputation (1, 0)
11. Data Quality (1, 0)
    1. Data Traceability (3, 0)
    2. Data Flexibility (3, 0)
    3. Data Trustability (2, 0)
    4. Data Expiration (1, 0)
    5. Data Novelty (1, 0)
    6. Data Source Information (1, 0)
    7. Data Specialization (1, 0)
    8. Data Value added (1, 0)
    9. Data Validity (1, 0)
    10. Data Reputation (1, 0)
12. Price (1, 0)
13. **Collaboration** (1, 0)
14. **Invisibility** (1, 0)
15. **Context Sensitivity** (1, 0)
16. Certification (1, 0)
    1. Standards Compliance (2, 0)
        1. Jurisdiction (1, 0)
17. Configuration Management (1, 0)
    1. Service Versioning (1, 0)
18. Documentation (1, 0)
19. **Currency** (1, 0)
20. Customer support (1, 0)
21. Server Location (1, 0)
22. **Service omnipresence** (1, 0)
23. **Function Composition** (1, 0)

Figure 14 - NFR-BoK - Relevant non-functional requirements

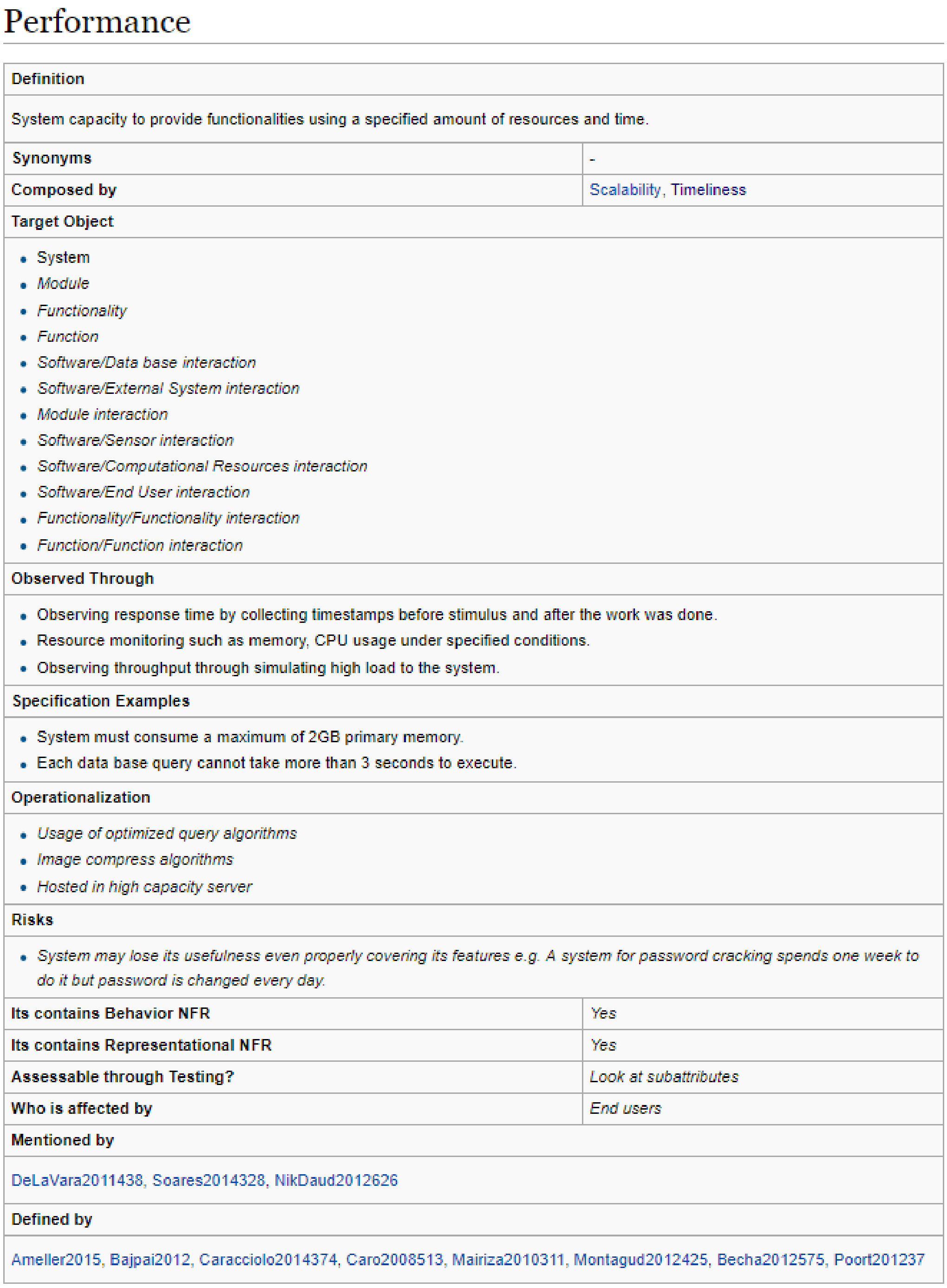
The detailed information about each NFR includes the following attributes:

[4] http://lens.cos.ufrj.br/nfrwiki

- **Definition:** NFR definition. Usually, an NFR description explains some system's capability, *e.g.*, performance: It is the system capacity to provide appropriate use of resources (memory, CPU) needed to perform full functionality under stated conditions;
- **Synonyms:** names having the same meaning, *e.g.*, reliability is presented as a synonym of dependability;
- **Composed by:** other NFRs that are part of the main NFR, *e.g.*, scalability, resource consumption, and timeliness compose performance.;
- **Target object:** system element through which the NFR can be observed. Examples of target objects of the performance NFR: (1) system performance (how the system is using memory during execution), (2) function performance (what is the response time of specific function observing the messages among system functions), (3) interaction with user performance (response time observing user request and time until response);
- **Observed through:** how the NFR can be observed or how the software exposes it. For instance, performance can be observed through resources monitoring or time observation in execution time;
- **Specification examples:** suggest how to specify an NFR, e.g., usability can be observed through user feedback;
- **Operationalization:** describes the mechanisms (system characteristics, properties, or features) used to operationalize the NFR. An example of security operationalization is to store the password encrypted;
- **Risks:** risks related to non-compliance with an NFR. For example, risks of no compliance with availability requirement: loss of business opportunities or slow productivity;
- **It contains behavior NFR:** defines if an NFR represents a software behavior, *e.g.*, "system services must response every request at most one second." Behaviors properties can be observed in execution time, therefore can be tested;
- **It contains representational NFR:** represents syntactical or technical software properties, *e.g.*, "Software must use MySQL database." Representational properties are static properties, and so they cannot be tested. However, it can be assessed through static techniques such as inspections;
- **Assessable through testing:** defines if the NFR is testable. It is *yes* if the NFR represent a system behavior;

- **Who is affected by:** the roles directly affected by the NFR. For example, Internal Stakeholders, Owner, Manager, Software Engineer, Programmer, Final User;
- **Mentioned by:** list of papers identifying the NFR, but not describing it;
- **Defined by:** list of papers identifying and describing the NFR.

Figure 15 presents an example of detailed information about *performance*.

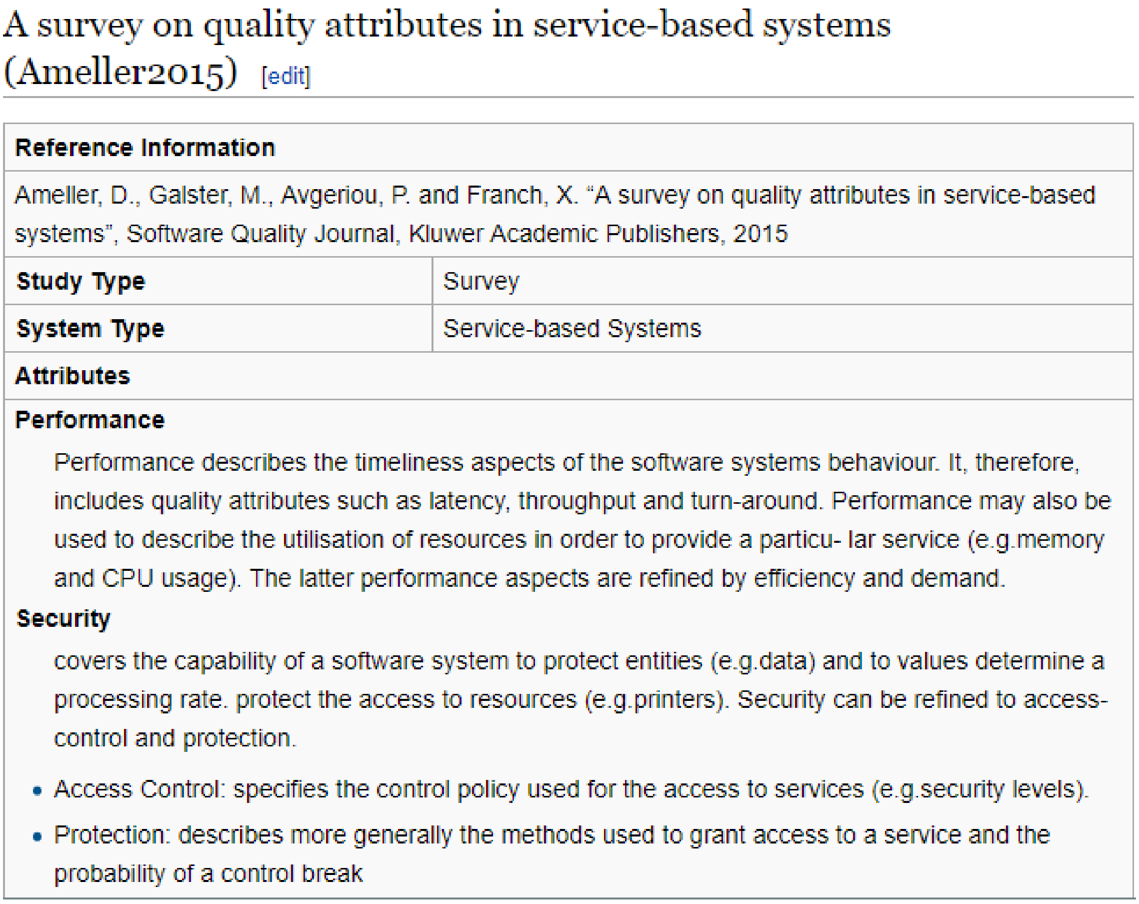


Performance

Definition	
System capacity to provide functionalities using a specified amount of resources and time.	
Synonyms	-
Composed by	Scalability, Timeliness
Target Object	
System *Module* *Functionality* *Function* *Software/Data base interaction* *Software/External System interaction* *Module interaction* *Software/Sensor interaction* *Software/Computational Resources interaction* *Software/End User interaction* *Functionality/Functionality interaction* *Function/Function interaction*	
Observed Through	
Observing response time by collecting timestamps before stimulus and after the work was done. Resource monitoring such as memory, CPU usage under specified conditions. Observing throughput through simulating high load to the system.	
Specification Examples	
System must consume a maximum of 2GB primary memory. Each data base query cannot take more than 3 seconds to execute.	
Operationalization	
Usage of optimized query algorithms *Image compress algorithms* *Hosted in high capacity server*	
Risks	
System may lose its usefulness even properly covering its features e.g. A system for password cracking spends one week to do it but password is changed every day.	
Its contains Behavior NFR	*Yes*
Its contains Representational NFR	*Yes*
Assessable through Testing?	*Look at subattributes*
Who is affected by	*End users*
Mentioned by	
DeLaVara2011438, Soares2014328, NikDaud2012626	
Defined by	
Ameller2015, Bajpai2012, Caracciolo2014374, Caro2008513, Mairiza2010311, Montagud2012425, Becha2012575, Poort201237	

Figure 15 - Performance details in the NFR-BoK

Clicking on a reference to the paper mentioning the NFR, the NFR-BoK presents the information extracted from the paper. For example, Figure 16 presents the information of the paper named as *Ameller2015*.

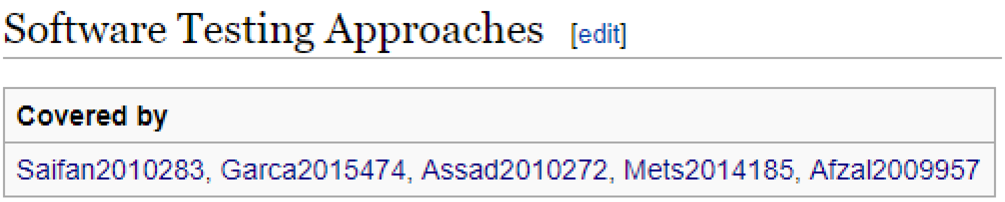


A survey on quality attributes in service-based systems (Ameller2015) [edit]

Reference Information	
Ameller, D., Galster, M., Avgeriou, P. and Franch, X. “A survey on quality attributes in service-based systems”, Software Quality Journal, Kluwer Academic Publishers, 2015	
Study Type	Survey
System Type	Service-based Systems
Attributes	
Performance Performance describes the timeliness aspects of the software systems behaviour. It, therefore, includes quality attributes such as latency, throughput and turn-around. Performance may also be used to describe the utilisation of resources in order to provide a particu- lar service (e.g.memory and CPU usage). The latter performance aspects are refined by efficiency and demand. **Security** covers the capability of a software system to protect entities (e.g.data) and to values determine a processing rate. protect the access to resources (e.g.printers). Security can be refined to access-control and protection. • Access Control: specifies the control policy used for the access to services (e.g.security levels). • Protection: describes more generally the methods used to grant access to a service and the probability of a control break	

Figure 16 - Example of a paper describing non-functional requirements

The software testing approaches covering the NFR are also presented in the NFR detail page (Figure 17).

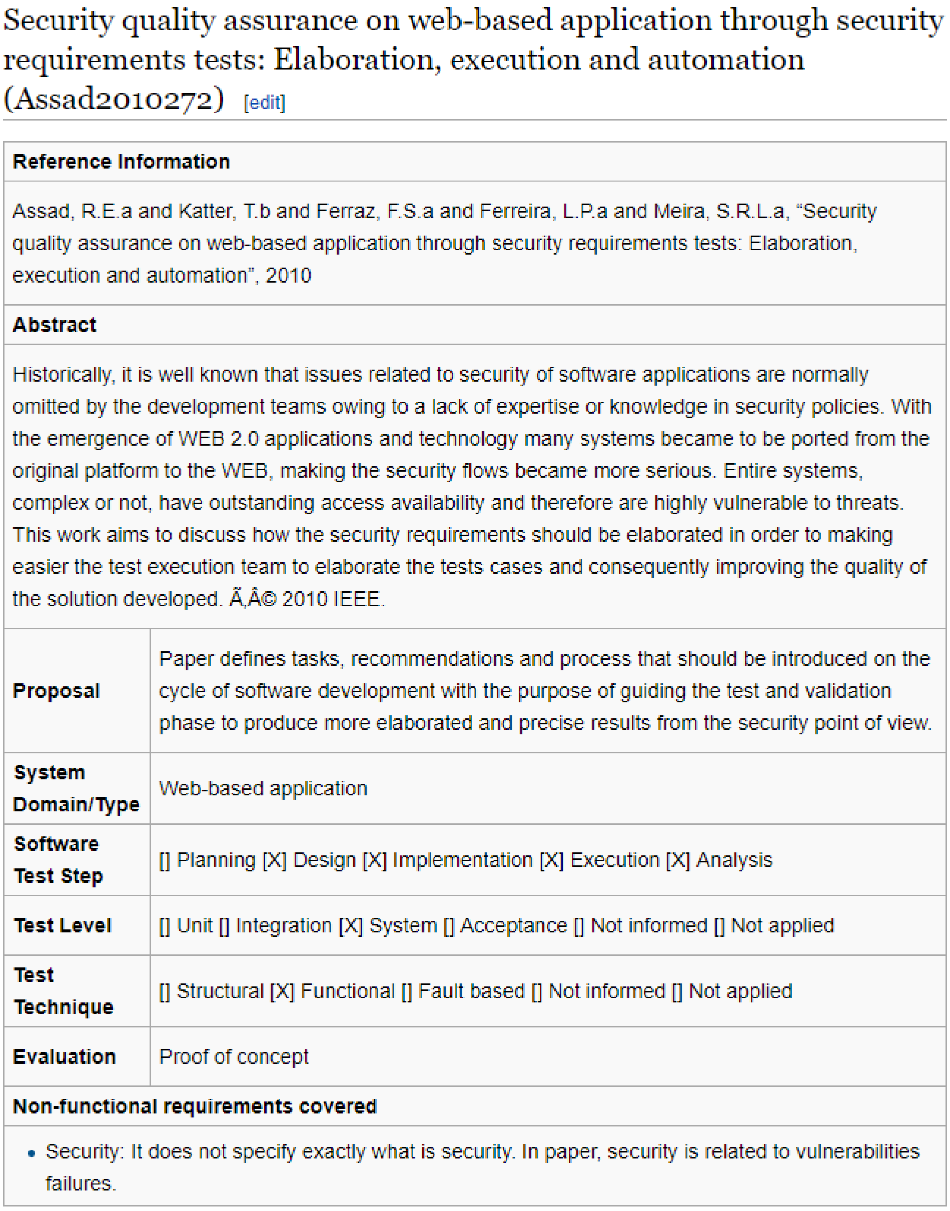


Software Testing Approaches [edit]

Covered by
Saifan2010283, Garca2015474, Assad2010272, Mets2014185, Afzal2009957

Figure 17 - List of testing approaches of an NFR

Clicking on the reference, the NFR-BoK displays detailed information about the testing approach. For instance, Figure 18 presents information about Assad2010272. The fields *Reference information* and Abstract are directly extracted from the paper that proposes the testing approach. The other fields are information extracted from the paper but interpreted the author of this thesis. The *Proposal* is a brief description of the testing approach, and the *System Domain/Type* represents the context the testing approach was proposed. *Software Test Step*, *Test Level*, and *Test Technique* describe the testing dimensions covered by the testing approach, and the *Evaluation* the kind of study was used to evaluate it. Finally, the field *Non-functional requirements covered* presents the NFRs the approach is able to assess.



Security quality assurance on web-based application through security requirements tests: Elaboration, execution and automation (Assad2010272) [edit]

Reference Information

Assad, R.E.a and Katter, T.b and Ferraz, F.S.a and Ferreira, L.P.a and Meira, S.R.L.a, “Security quality assurance on web-based application through security requirements tests: Elaboration, execution and automation”, 2010

Abstract

Historically, it is well known that issues related to security of software applications are normally omitted by the development teams owing to a lack of expertise or knowledge in security policies. With the emergence of WEB 2.0 applications and technology many systems became to be ported from the original platform to the WEB, making the security flows became more serious. Entire systems, complex or not, have outstanding access availability and therefore are highly vulnerable to threats. This work aims to discuss how the security requirements should be elaborated in order to making easier the test execution team to elaborate the tests cases and consequently improving the quality of the solution developed. Ã‚Â© 2010 IEEE.

Proposal	Paper defines tasks, recommendations and process that should be introduced on the cycle of software development with the purpose of guiding the test and validation phase to produce more elaborated and precise results from the security point of view.
System Domain/Type	Web-based application
Software Test Step	[] Planning [X] Design [X] Implementation [X] Execution [X] Analysis
Test Level	[] Unit [] Integration [X] System [] Acceptance [] Not informed [] Not applied
Test Technique	[] Structural [X] Functional [] Fault based [] Not informed [] Not applied
Evaluation	Proof of concept

Non-functional requirements covered

- Security: It does not specify exactly what is security. In paper, security is related to vulnerabilities failures.

Figure 18 - Detailed information on a testing approach

It is important to note that literature reviews were not specifically planned to identify some of the attributes of the body of knowledge. For instance, the aim of the literature review was not to identify the *operationalization* mechanisms of an NFR. Thus, the NFR-BoK is not a wholly finished knowledge repository, but it should evolve as new researches focus on specific points.

3.7 Conclusions from the literature reviews results

These literature reviews are a preliminary initiative aiming to identify and understand relevant NFRs to the success of software systems and to figure out the testing approaches to assess such NFRs. Consequently, through the analysis of the results of the two literature reviews, it was possible to identify gaps between the relevant RNFs and the proposed testing approaches.

Identifying and understanding the most relevant NFRs was an important step in to limit the scope of this thesis to the *security* and *performance* since it would be unfeasible to deal with all relevant NFRs. Besides, the results help to understand the need to expand the thesis scope to software verification (testing and reviews) since there are NFRs that cannot be assessed by testing approaches.

Regarding the software testing approaches, it is a topic that needs research presenting new proposals and evolving the consolidated knowledge since the proposed testing approaches do not fully cover the *testing process phases*, the different *testing levels*, and the *testing techniques*. Moreover, most of the identified software testing approaches are evaluated through experimental methods that offer weak confidence results, such as *proof of concept*. It results in the lack of evidence on the benefits and risks that the use of these approaches can bring. Therefore, the literature reviews emphasize the importance of research related to the evaluation of NFRs.

4 A Perception of the State of the Practice of Security and Performance Verification

This chapter presents the characterization of the security and performance practices employed by software development organizations. It begins by introducing the study context, labeling the organizations and participants. After, it describes the security and performance verification practices performed by such organizations and the decision-making criteria related to these practices.

4.1 The methodology to characterize S&P verification

A case study supports the results presented in this chapter. It is classified as a characterization case study, and it follows the recommendations presented in the guidelines proposed by Runeson and Höst [2009]. Table 9 shows the mains sections of the case study protocol (Appendix E).

Table 9 - Case study research protocol

Objectives
Understand how security and performance verification has been performed in the software development industry.
Scope
Software development organizations that perform Security OR Performance Verification activities.
Research method
• Multiple case studies: one of them including an observational phase and others with only semi-structured interviews and questionnaire data collection phases ○ 1 organization with 1 project as the primary case: including observational, semi-structured, and questionnaires data collection method; ○ 3 organizations with 1 project each as complementary cases: including semi-structured and questionnaires data collection method; • Flexible design ○ Trying to improve the protocol during the study execution • Predominantly qualitative • Criteria for case selection

○ Projects in progress for at least two months
Data sources
Organizations employers, researcher observations, institutional websites
Unit of analysis
Software development projects, including security or performance verification activities.

Besides, the research protocol contains the research questions grouped by two mains subjects. The first aiming to identify the practices used by organizations on the security and performance verification and to characterize such practices. The second one aiming to identify factors influencing the decisions regarding the verification practices:

- **RQ 1 Which are the practices used by the organizations to support the verification of security and performance?**
  - RQ 1.1 What are the standard techniques?
  - RQ 1.2 Which definition of done do they adopt?
  - RQ 1.3 How is the level of automation?
  - RQ 1.4 What are the assets covered?
- **RQ 2 How does the organization define its security and performance verification strategies?**
  - RQ 2.1 What are the factors influencing the decision-making on security and performance verification strategies?
  - RQ 2.2 When does the decision on the verification strategy happen?
  - RQ 2.3 How often the decisions on the verification strategy occur?
  - RQ 2.4 Who is responsible for the decision making on the verification strategy?

After, the organizations were selected by convenience. We tried to identify among our personal contacts people who work in software development organizations. Thus, we selected ten people working in different organizations, and we started the initial contact. However, three organizations did not answer to our request, and three of them said they do not perform security or performance verification. Thereby, we were able to select a set of four organizations as subjects of the study.

4.1.1.1 The data collection process of the case study

A set of artifacts was used to support the study, including the case study protocol (Appendix E) and a presentation letter (Appendix F). Additionally, instruments (Appendices G-N) were used to collect data to answer the research questions directly and to formalize the research agreement and the characterization of the organizations and participants. The author of this thesis filled the instruments when collecting data during observations and interviews. The participants filled out the questionnaire when it was used to collect data. Table 10 presents a summary of the used instruments.

Table 10 - Case study instruments description

ID	Description	Objectives
P1	Case study protocol	The protocol followed by the case study.
C1	Presentation letter	A letter used to make the first contact with the organizations, characterizing the researchers involved and the objectives of the study.
I1	Organization agreement term	After the organization agrees with the research, its representative signs the agreement term. It marks the beginning of the case study.
I2	Participant agreement term	In the first contact with each participant, they must sign the consent term to allow us to collect and use data.
I3	Organization characterization	Data can be gathered from different sources as the participants and organization websites. It was filled in during different moments of the case study execution.
I4	Project characterization	It supports data collection while interviewing participants through a questionnaire.
I5	Participant characterization	It supports data collection after interviewing participants through a questionnaire.
I6	Verification practices identification	Data collected in different stages of case study execution. It was gathered from observation, interviews, and questionnaires.
I7	Identification of decision-making factors	Data collected in different stages of case study execution. It was gathered from observation, interviews, and questionnaires.
I8	Participant opinion	It supports data collection after interviewing participants through a questionnaire.

The first step of the data collection process was to sign the organizational agreement represented by the instrument I1. This is crucial to ensure the integrity of research and organization privacy. In the second step, data regarding organizational characterization (I3), project characterization (I4), and participant characterization (I5) were collected. Furthermore, the participant agreement term is signed because it is the first contact with some of the participants.

The semi-structured interview step used instrument I4 to collect data related to the development process and the environment. This step aims to understand more fully how the organization operates. The instruments I5 and I6 were used to identify the security and performance practices and decision-making factors, respectively. It is important to note that some semi-structured interviews were recorded. In these cases, the instruments were not filled, but they were used as a guide for the interviews.

The observational step aims to confirm previously collected data and to understand how the verification practices were performed in detail. This step was performed in only one of the organizations, and its main output is the researcher's manual notes.

The last step is the application of a questionnaire to collect the opinion of the participant about security and performance verification.

It is essential to mention that the researcher collected data in the form of manual notes at every stage. Figure 19 shows the data collection process, and the instruments were filled.

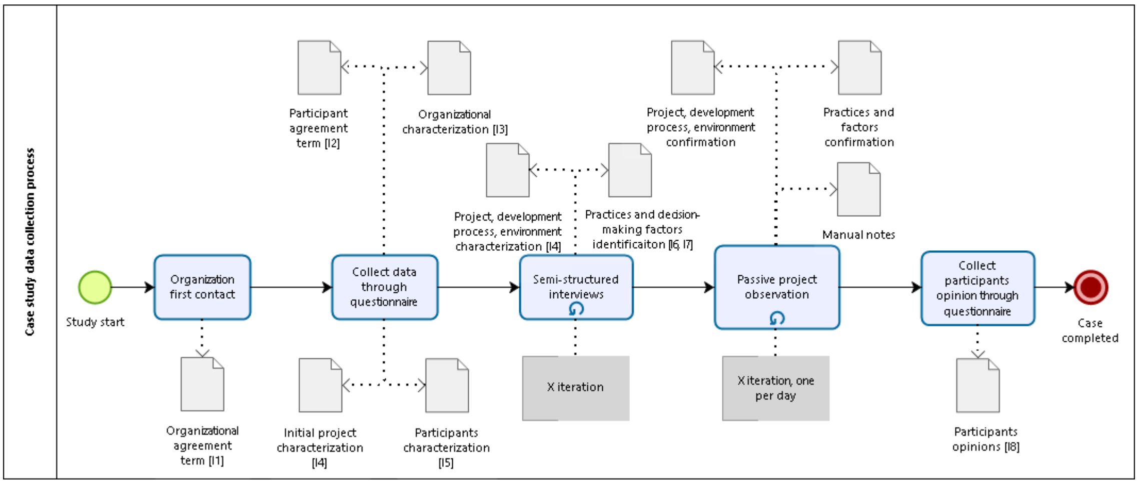

Case study data collection process
Participant agreement term [I2]
Organizational characterization [I3]
Project, development process, environment confirmation
Practices and factors confirmation
Project, development process, environment characterization [I4]
Practices and decision-making factors identificaiton [I6, I7]
Manual notes
Study start
Organization first contact
Collect data through questionnaire
Semi-structured interviews
Passive project observation
Collect participants opinion through questionnaire
Case completed
Organizational agreement term [I1]
Initial project characterization [I4]
Participants characterization [I5]
X iteration
X iteration, one per day
Participants opinions [I8]

Figure 19 - Data collection process

4.1.1.2 *The data analysis process of the case study*

Approximately 47 artifacts were filled out. Then, the author qualitatively analyzed them by following a coding process. The MAXQDA[5] tool, into which all instruments were imported, was used to support the coding process, and 955 artifact excerpts were grouped in 1112 codes.

The coding process was divided into two parts. The first part aimed to answer RQ1 and RQ2. Thus, the author read the artifacts looking for verification practices and practices characteristics. Additionally, the two other researchers revised the generated codes in several meetings sections throughout the process. Figure 20 presents a model representing the structure of codes built during the coding process.

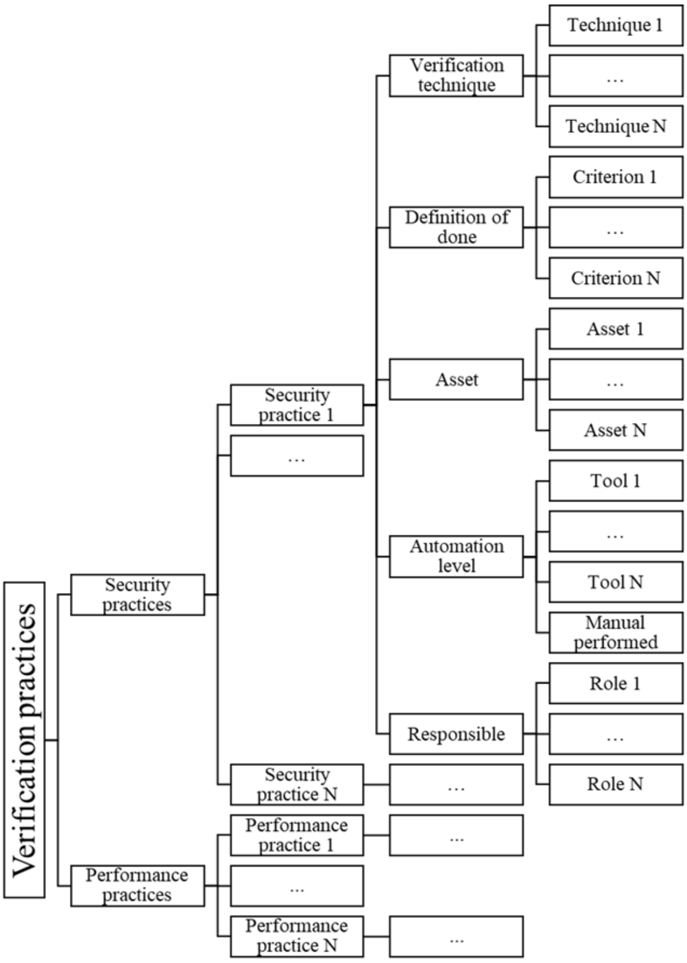


Figure 20 - Structure used to characterize verification practices (RQ 1)

[5] https://www.maxqda.com/

The process to answer RQ 2 was similar to the process to answer RQ 1, but the aim was to identify decision-making factors. Figure 21 represents the structure used to identify decision-making factors (RQ 2) during the coding process.

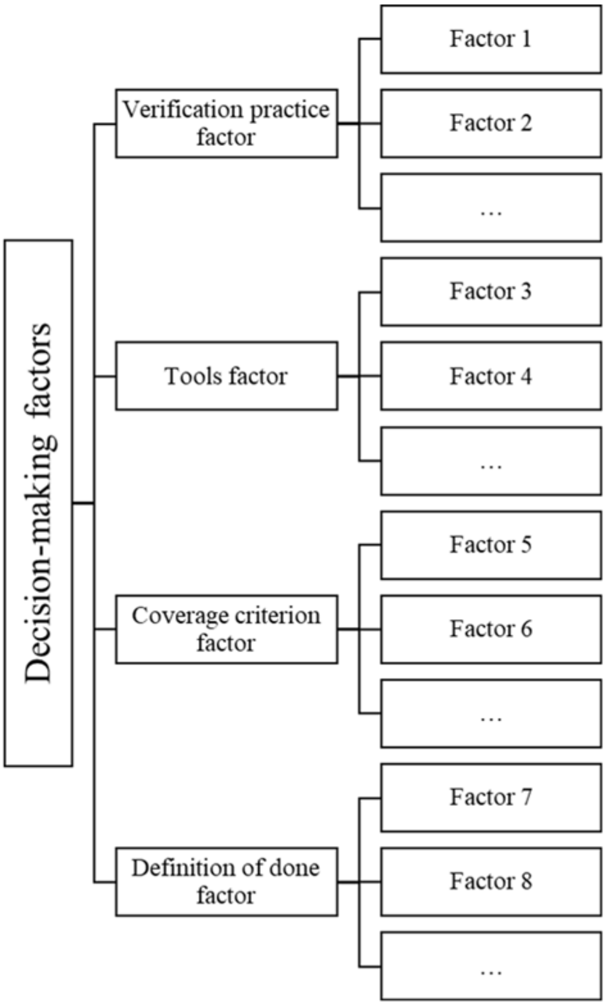


Figure 21 - Structure used to identify decision-making factors (RQ 2)

4.2 The case study context: organizations and participants

It is essential to highlight the characteristics of organizations and participants of the study to allow the application of the results in other contexts. We believe the diversity of the organizations' profiles will facilitate the generalization of the results of this thesis.

4.2.1 Describing the organizations

Table 11 presents an overview of the organizations participating in the case study. The name of organizations is hidden to preserve their privacy.

Table 11 - Organizations' description

Id	Nature	#Employees (#Developers)	#Subjects	Data collection method

Org1	Governmental	~10599
		(Unknown)
			5	Observation;
				Interview;
				Questionnaire

Org2	University tech transfer laboratory	~154 (~132)	2	Interview; Questionnaire
Org3	Private	~250 (150)	2	Interview; Questionnaire
Org4	Military	~80507 (Unknown)	3	Interview; Questionnaire

Org1 is a large governmental organization that provides information technology services to the Brazilian government with 10599 employees (most of them are developers). We performed observations, interviews, and questionnaires with a verification team composed of five employees.

The university tech transfer lab, identified as *Org2*, with about 154 employees (132 are developers), develops technical solutions to the Brazilian government, including the development of software. We performed interviews and questionnaires with two employees that are responsible for security and performance verification.

Org3 is a private company with about 250 employees and 150 developers. The company develops credit card payment systems, and we gathered data from two employees through interviews and questionnaires.

Org4 is a military organization in which it was possible to investigate a big department of software development. *Org4* has about 80 thousand employees, but we couldn't identify how many people have been working directly with software development. In this organization, the data were collected from 3 participants by interviews and questionnaires.

Table 12 presents the agility level of the organizations. It is not exhaustive, but it provides a perception of the practices adopted by them. Thus, we asked the participants about the agile practices used by them. Besides, some participants reported that they failed to implement agile practices. A participant of *Org3* said it is challenging to keep frequent contact with the clients because of client availability. The difficult to implement an agile methodology in *Org4*, a military organization, is related to its rigid employee hierarchy.

Table 12 - Organization agility level

Agile Practice	Org1	Org2	Org3	Org4
Automated testing	X	X	X	X
Continuous integration	X		X	X
Frequent meets with the client		X		
Internal daily meetings	X			
Kanban	X		X	X
Scum based	X	X	X	X
Self-allocation of tasks	X			
Squad-based			X	
Test-driven development		X		

It is possible to note that *automated testing* and *scrum* are practices adopted by all organizations. *Continuous integrations* and *Kanban* are the next most used practices. We advise that this data must not be used to claim that an organization is more agile than another one. For example, it is not possible to claim that *Org1* is more agile then *Org2* because *Org1* implements six agile practices, and *Org2* implements four agile practices. The number of agile practices implemented by an organization could be influenced by the number of observations points (number of participants, described in the next section). Thus, such information aims to enrich the organization's characterization, making it easy to use the study results in similar contexts.

4.2.2 *Describing the participants*

Table 13 shows information about the twelve study participants. It is important to note that the participants have a professional profile without executive power, and the values of participants' experience are not accurate because it is a self-reported experience. The × means the participant did not answer the question.

Regardless of the primary role of the participants (for instance, *P1.4* is a software architect), all participants of Org 1, 2, 3, and *P4.1* directly perform some activity related to S&P verification. Participants *P4.2* and *P4.3* are not directly performing verification activities, but they are involved indirectly by requesting and evaluating the output of such activities.

Participant *P4.2* does not have experience in testing and NFR testing. However, he is part of a project that contemplates S&P verification. Thus, his report is considered essential.

Table 13 - Participants' characterization

Question		Main role	Education level	Software development experience (months)	Software testing experience (months)	NFR testing experience (months)	# of software development projects
Org 1	P1.1	Test analyst	Master D.	144	96	12	20
	P1.2	Develop. analyst	Undergrad.	168	96	12	12
	P1.3	Develop. analyst	Undergrad.	144	6	6	20
	P1.4	Software architect	Lato sensu specializ.	132	24	12	20
	P1.5	Test analyst	Lato sensu specializ.	276	×	×	20
Org 2	P2.1	Security test analyst	Undergrad.	×	84	48	70
	P2.2	Security test analyst	Undergrad.	×	60	60	×
Org 3	P3.1	Programmer	Undergrad.	120	48	×	20
	P3.2	Security test analyst	Lato sensu specializ.	×	×	120	×
Org 4	P4.1	Security test analyst	Master D.	360	12	12	15
	P4.2	Full stack developer	Lato sensu specializ.	120	0	0	20
	P4.3	Software architect	Lato sensu specializ.	180	72	72	10

4.3 Security and performance verification practices

This section presents the S&P verification practices (RQ 1) and their characterization regarding techniques (RQ 1.1), the definition of *done* criteria (RQ 1.2), automation level (RQ 1.3), and assets (RQ 1.4). Figure 22 presents a brief overview of the practices supporting S&P verification, and the next section describes the practices.

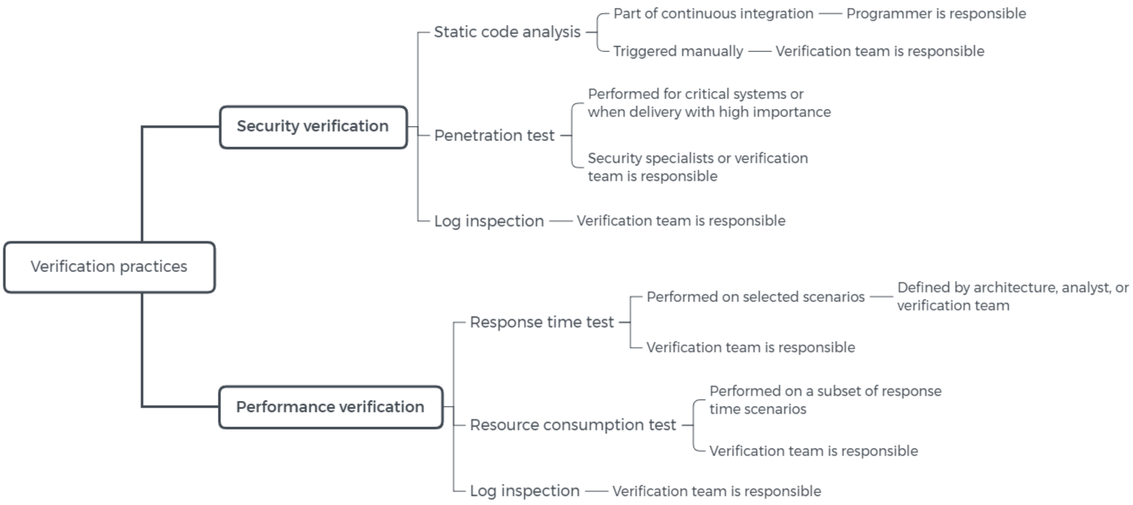

Verification practices
Security verification
Static code analysis
Part of continuous integration
Programmer is responsible
Triggered manually
Verification team is responsible
Penetration test
Performed for critical systems or when delivery with high importance
Security specialists or verification team is responsible
Log inspection
Verification team is responsible
Performance verification
Response time test
Performed on selected scenarios
Defined by architecture, analyst, or verification team
Verification team is responsible
Resource consumption test
Performed on a subset of response time scenarios
Verification team is responsible
Log inspection
Verification team is responsible

Figure 22 - Identified security and performance verification practices

4.3.1 RQ1: Security and performance verification practices

Security *static code analysis* is performed in two ways, either triggered by a tester analyst or embedded in a continuous integration tool (*e.g.*, Jenkins). In the first case, the code inspection depends on human action, and it is the verification team's responsibility to perform this practice. In the second, the code inspections mandatorily happen when the programmer commits the code to the repository.

The *penetration testing* practice is performed at the end of the software development lifecycle. It is usually performed only for critical systems (or a part of them), and if the product owner defines delivery as critical because an attack could harm the organization's reputation. Security specialists or the verification team are who usually perform the penetration tests.

Regarding *log inspection*, we classify it as a verification practice when it is used to identify software failures and as a debugging practice when it is used to identify software faults (failures cause). Further, this classification became comfortable because specialists participating in the case study mentioned such activity as a verification practice. For instance, one interviewee said: "*In a project I participated in, there was one IP accessing the system, trying to identify if it was built in PHP and Wordpress. So, it was a well-known vulnerability they were searching for*". Therefore, the log inspection allows the identification of what could be considered a security failure: the application was configured to show the technologies used.

The verification team performs a *response time test*, *resource consumption*, and *log inspection*. Response time test is the execution of the software aiming to evaluate the amount of time from a request to a response. This practice is performed at the end of a development iteration (*e.g.*, a sprint), and it is not used to assess all system scenarios, but the analysts (*e.g.*, architecture, business) or the verification team selects some system scenarios to be assessed.

The *resource consumption test* is also performed at the end of a development iteration and uses the same test cases regarding *response time tests*. However, the system scenarios evaluated by this practice are a subset of scenarios evaluated by the response time test.

Log inspection is also used as a practice to performance verification, and it is performed aiming to identify significant delays from a request to a response.

Figure 23 presents the characterization of identified security verification practices regarding their *techniques*, *the definition of done*, *automation level*, *and assets*. Figure 24 presents the same information regarding performance practices.

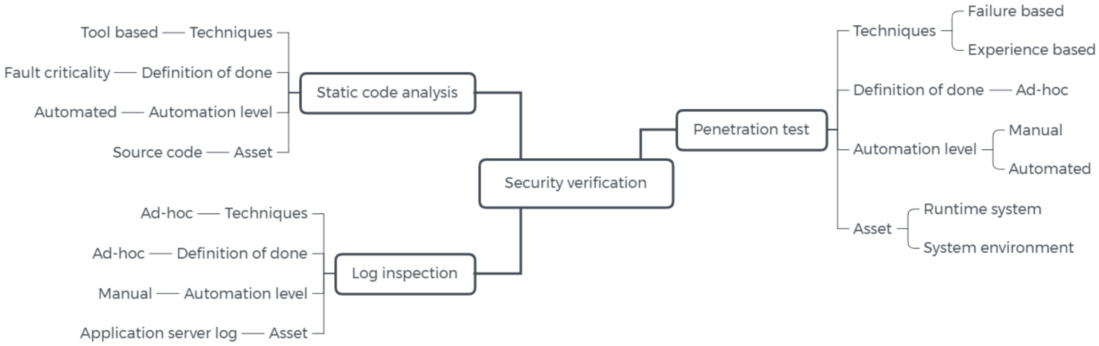


Figure 23 - Software security verification practices details

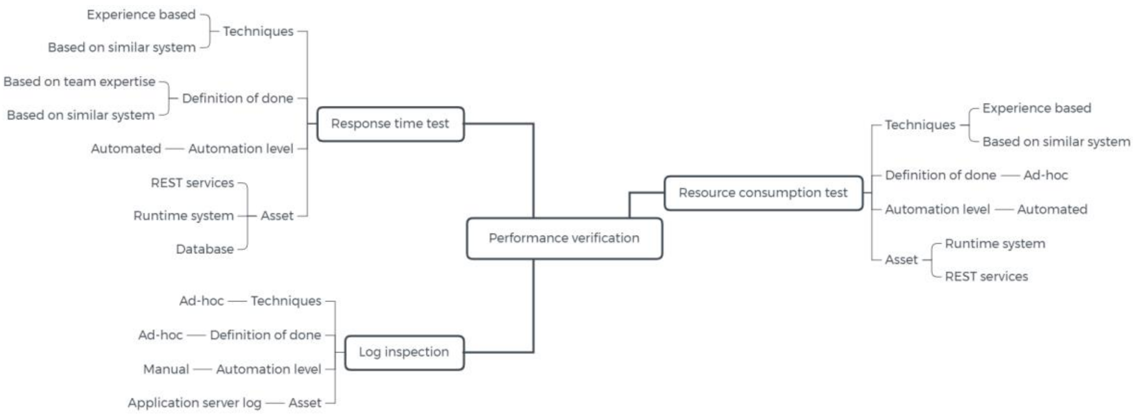


Figure 24 - Software performance verification practices details

4.3.2 RQ 1.1: The techniques supporting security and performance verification practices

The security and performance verification practices are executed using verification techniques, providing a systematic way to build test cases or review procedures, and supporting the definition of coverage and stop criterion. There were different categories of techniques used to evaluate security: *tool-based*, *failure-based*, *experience-based*, *based on a similar system*, and *ad-hoc*.

4.3.2.1 Techniques of security practices

Regarding the techniques of security practices (Figure 23), if it is *tool-based*, the technique is embedded in the tool. *Failure-based* techniques make use of known failures (*e.g.*, common vulnerabilities databases), generating test cases addressed to identify these faults. In *experienced-based techniques*, the verification team makes use of their own experience to generate test cases or perform inspections. If the technique is *ad-hoc,* then the practice is performed in an aleatory and nonsystematic way.

The *Static Code Analysis* is usually performed through a tool (*tool-based*), meaning that a tool is executed, and the verification results are analyzed. An observed issue with the usage of this technique is that in many cases, the verification team is usually not aware of what the tool is verifying and what are the limitations of the tools regarding their fault detection capability.

The *Penetration Test* is usually performed through a *failure-based* technique, and the test cases are built, aiming to explore known defects available on common security vulnerabilities repositories. Another technique used for the *penetration test* is *experience-based* in which a security specialist knowledge plays the role of a malicious user trying to access the system.

The *Log Inspection* can be considered an *ad-hoc* practice because the inspection is based on unknown criteria. A security specialist performs it.

4.3.2.2 Techniques of performance practices

Regarding the techniques used by performance practices (Figure 24), both the *Response Time Test* and *Resource Consumption Test* use techniques based on the tester *experience* or *based on similar systems*. In the second case, the verification team verifies whether the current system can reuse test cases from previous systems of the same company. As for security, the performance Log Inspection is *ad-hoc*, and the verification team performs it.

4.3.3 RQ 1.2: Criteria to define security and performance activities as done

The definition of *done* is the criterion used to define the conclusion of verification activities, signalizing that the software development can move to the next phase, usually delivery. We identified four categories to the definition of *done*, based on *fault criticality*, *team-experience*, a *similar system*, and *ad-hoc*.

On the definition of *done* based on *fault criticality*, the faults identified by verification are classified regarding their criticality level (*e.g.*, low, medium, and high). Verification activities are defined as completed when no more defects of a defined criticality level are identified. The static code analysis makes use of this criterion. However, the minimum criticality level to define code analysis should re-run is not formally defined. One of the participants said: "*The defects (vulnerabilities) reported by Fortify* (security verification tool) *are classified by the tool, according to the criticality level. So, in some situations, we used such a criticality level to define what faults would be fixed. Thus, the most critical faults were fixed. However, no rules were established to limiting what the criticality level determines that faults must be fixed. Such a decision* *was a cross-team agreement."* When based on *team experience*, the verification team meets to decide whether verification activities should be continued, or a new verification battery is needed.

Besides, the verification team may decide to investigate a *similar system* to decide when the verification activities should end. For instance, in the last similar system, the tests were set as concluded after each test ran three times successfully. Thus, the current system tests stop after three tests battery. The response time makes use of these two kinds of the definition of *done*. If the definition of done is *ad-hoc*, there is no systematic way to set the verification as concluded, and it is randomly performed. The penetration test, log inspection, and resource consumption use the *ad-hoc* definition of *done*.

4.3.4 RQ 1.3: The automation level of security and performance verification practices

The evaluation of the *automation level* classifies the verification practice between *manually* and *automated*. If it is *automated,* then we identify the tools supporting the verification practice. Appendix D presents the identified supporting tools.

Static Code Analysis is always *automated*; the most common tools in our case studies were Brakeman and HP Fortify SCA. Additionally, Jenkins, Sonar, and Threadfix are used as auxiliary tools to orchestrate the execution or to visualize results. When using static code analysis tools, participants reported that a problem is a large number of false positives.

The Penetration Test practice is performed *manually* (supported by scripts) or by using a support tool; the most common tools are Arachni, OWASP Zed Attack Proxy (ZAP), XSS ME, SQL Injection ME, Burp Suite, Meta Exploit, NMAP, and Whatweb.

The Response Time and Resource Consumption Tests are always *automated* and make use of similar tools, such as JMeter, Postman, BlazeMeter, Goldeneye, HTTPerf, SoapUI, CA APM, and tools developed by the organization. There are also auxiliary tools such as Jenkins, spreadsheets.

The Log Inspection for both security and performance is manual. Probably, it indicates a lack of log inspection tools. One of the participants informed the performance failures identified by performing log inspection might have many false positives. It is not easy to conclude if a detected anomaly is a regular user behavior or a system bottleneck: *“...when looking at logs, if I identify a 10 minutes delay from a request to another request, the problem is that I do not know if the request took 10 minutes to complete or if the user clicked on one option and then clicked on another only 10 minutes later.*”

4.3.5 RQ 1.4: Assets covered by security and performance verification practices

Here, we aim to identify which part of the system the verification activities are intended to assess.

The assets covered by security verification practices are *source code*, *application server log*, *the system in execution*, and *the environment*. Performance verification practices cover *REST services*, *the system in execution*, *database*, and *application server log*.

It is important to note that verification practices in the case studies do not cover early development phases artifacts.

4.4 Decision-making factors of security and performance verification

Regarding decision-making factors, it was possible to answer RQ 2.1 (*What are the factors influencing the decision-making regarding security and performance verification strategies?*) and 2.4 (*Who is responsible for the decision making regarding the verification strategy?*), but not RQ 2.2 (*When are the decisions on the verification strategy made?*) and 2.3 (*How often are the decisions on the verification strategy made?*). Nevertheless, such findings are an initial set of decision-making factors related to security and performance verification. These results can facilitate future studies on a deeper understanding of how software development organizations have been making decisions regarding security and performance verification.

Figure 25 presents the decision-making factors regarding the choice of tools, verification practices, coverage criteria, and the definition of *done*. Besides, it presents who is responsible for decision-making.

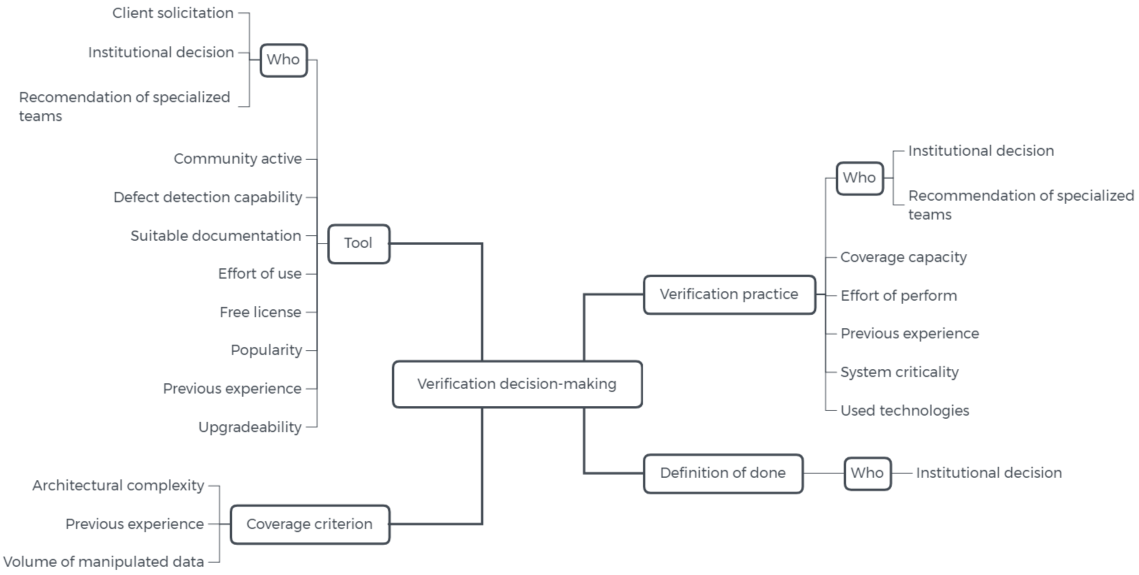


Figure 25 - Decision-making factors regarding security and performance verification

4.4.1 RQ 2.1: The factors influencing the decision-making of security and performance verification

4.4.1.1 Choosing security and performance support tools

Two factors influencing the use of a tool are interrelated: *community active* and *popularity*. It means that the choice of a tool can be influenced by the fact that it has been widely used, that several developers contribute to its development (providing suggestions for improvements and bug fixes), and that it is broadly discussed in blogs, professional websites, and social networks.

Defect detection capability is a self-explanatory decision-making factor, implying that the tool should be efficient. The *effort of use* is an essential factor because the tool can meet all other decision-making factors, but if its use requires additional effort, it becomes inviable owing to economic or time issues.

If the tool has a *free license,* it can be included in the verification process if there is technical staff consensus. Thus, the management staff need not be included in the choice process because it is not necessary to approve the tool purchase order. Therefore, some participants said they often favor to free tools for avoiding the bureaucracy of the buying process.

Another factor weighed in choosing the tools is related to the availability of *suitable documentation*. Such a factor is also essential because it provides the way users can understand how to operate the tool and understand if it meets other decision-making requirements. For instance, the tool documentation should provide information supporting the perception of the *effort of use* and *defect detection capability* factors, as well as explain the license by which the tool is available.

To choose a tool, it is also essential to consider its *upgradeability*. This decision-making factor can be perceived by the periodicity and the date of the last release. Finally, a tool can be selected because a member of the team has *previous experience* using it in other projects.

4.4.1.2 Choosing verification practices

Regarding the decision-making factors used to select the security and performance verification practices, the *coverage capacity* is used to identify which assets are covered by the practice (*e.g.*, source code, design artifacts, and requirements), and the *effort to perform* it is related to its financial and time viability.

System criticality is a decision-making factor because it defines the severity of verification practices. If a system is highly critical, more verification practices are performed. For instance, performing only static code analysis may suffice for a non-critical system. However, for a critical system, other practices, such as threat modeling and penetration testing, should be included.

The *technology used to construct the system* is also an essential factor. There are security and performance verification practices specific to software technology. For example, if the system does not use an SQL database, it is useless to perform practices exploiting SQL-injection vulnerabilities.

Previous experience using a verification practice is a decision-making factor because some members of the verification team tend to use practices of past projects. Thus, it is possible to mitigate the effort and time spent learning the verification practice.

4.4.1.3 Choosing the coverage criteria

Coverage criteria define the percentage that a part of the system (an asset) should be exercised by a test suite or revised by a static technique. For instance, it is used to decide if the static code analysis should be performed considering all source code or a piece of it.

The decision-making factors related to coverage criteria are *architectural complexity*, *previous experience*, and *volume of manipulated data*. It was possible to realize that when a system has a very complex architecture or the expectation that part of the system will manipulate a high data volume, the coverage of the tests is reinforced.

Previous experience is also a decision-making factor used to select coverage criteria. In this case, the verification team looks at other similar systems evaluating the possibility of using the same coverage criterion.

It was not possible to identify decision-making factors related to the selection of the definition of done.

4.4.2 RQ 2.4: The decision-makers

Initially, it was hoped to collect information aiming to answer RQ 2.4 with the roles (*e.g.*, manager, product owner, and architect) responsible for decision making. However, it was possible to identify only the level these decisions are made.

Sometimes, there is a *client solicitation* for the use of a tool. In this case, the verification team does not have the autonomy to make a choice and use their favorite tools. This way of choosing the tools can be harmful to verification as a participant said: "*I would like to use a specific tool, but the customer asked me to use another tool. But after, he regretted it because the tool does not work as he thought…*"

The *institutional decision* means that a higher-level department of the organization is responsible for the decision-making, and the verification teams must follow such decisions. The selection of tools, verification practices, and the definition of done can be decisions made at the *institutional level*.

The tools and verification practices can be select through a *recommendation of specialized teams*. In this case, a team of experts selects tools or verification practices suitable for a project.

Finally, it was not possible to identify who may select a coverage criterion.

4.5 Conclusion

Characterization studies allow the understanding of the state of the practice, providing insights to guide new studies to target real and relevant issues. Thus, there is an evolution of the knowledge of the state of the art, and then, this knowledge can return to the practice.

In this sense, this chapter presented the results of a case study performed in the context of Brazilians software development organizations. The case study aims to characterize the state of the practice of S&P verification activities. Besides, it presents the decision-making factors related to these activities.

It was possible to realize that there is an increasing awareness of the importance of security and performance of software systems, and consequently, the importance of verification activities. However, there is a lack of knowledge about how verification should be accomplished.

Thus, the use of systematic verification techniques is low. For instance, techniques aiming to systematize the test case generation or inspection procedure. It may result in inefficient test cases (low chance to reveal a failure). Such a warning is emphasized by the profile of the professionals who perform the verification. Usually, they do not have suitable experience and training regarding security or performance.

Finally, it was not possible to identify verification activities addressed to artifacts related to the early stages of the software development cycle (*e.g.*, requirements and design diagrams). It contradicts the recommendations of established guidelines [Howard and Lipner 2006] [OWASP 2014].

5 Moderator Factors of Security and Performance Verification

This chapter presents the eight moderator factors influencing security and performance verification, explaining the reasoning behind them. Besides, it presents a set of actions to promote each of the factors. Such factors emerged from the practice (case study), and then their confidence and pertinence were confirmed through the technical literature (rapid reviews) and the opinion of practitioners (survey).

5.1 The methodology used to identify the moderator factors

The case study described in Section 4.1 is part of the research method used to identify the moderator factors. While we were analyzing the collected data through a coding process, it was possible to identify some significant findings for the security and performance verification. Such findings were organized in a set of conjectures (inference formed statements without proof or sufficient evidence [Merriam-Webster 2011]) about S&P verification.

The coding process was performed to identify the conjectures following the principles of the coding phase of grounded theory [Corbin and Strauss 1998]. This methodology includes three coding phases. The open coding phase is an analytical process to identify concepts, their properties, and dimensions. In this phase, the data is fragmented and conceptually labeled in codes. When similarities between codes are found, they are grouped into categories. In axial coding, the categories can be rearranged in subcategories, and new categories can be created. Subsequently, the created categories are unified around a central core category, and relationships are established among them. This last phase was not performed in this study. Finally, step 4 of thematic synthesis [Cruzes and Dyba 2011] was used to translate the codes into themes.

Instantiating the coding process to this work, first, the author of this thesis read every artifact creating codes related to each relevant part of the collected information.

In this step, the researchers were not concerned about grouping codes into categories, but when the excerpts were similar, they were linked to the same code. For instance, when a participant said: "*The problem on using these tools is a large number of false positives*," and another said "*There are organizations that perform security testing only with automated tools, but this generates a large number of false positives*." These two excerpts were linked to the same code "*Automated tools generate a large number of false positives*." After, two other researchers reviewed the first phase of coding, suggesting corrections, mainly in the names assigned to each coding.

In the axial coding phase, the author compared the created codes interactively, for example, by comparing code 1 with code 2, code 3, up to code N. Subsequently, by comparing code 2 with code 3, up to code N (code 1 is ignored because code 1 and code 2 have already been compared) and so on. When similarity was found between two codes, they were grouped into a category. This step required approximately four interactions, and some codes could not be grouped in a category. Then, the categories (as well as the codes in them) were analyzed to concatenate similar categories into one. It also required more than one interaction. Finally, the author iterated through the categories to group them in a more general category, creating the structure: category > subcategory > codes. This phase was not performed in a linear way; the process was interrupted several times so that other researchers could review the coding already done. Thus, they suggested changes such as the inclusion of categories, change of category names, and rearrangement of coding between categories.

Finally, the author iterated through the consolidated structure of categories, subcategories, and codes, translating them into a set of eight high-level themes representing conjectures about security and performance verification. Additionally, it is highly relevant to note that other researchers always iteratively check this phase in several sessions throughout the process.

Additionally, given the importance of these unexpected but essential findings, we performed a set of secondary studies, in the form of *rapid reviews* (RRs) [Tricco et al. 2015] and *Snowballing* [Wohlin 2014], searching for literature to support them. After analyzing the extraction forms of secondary studies through a *coding process* and thematic analysis, it was possible to consider the conjectures as moderator factors that influence the verification of security and performance.

5.1.1 *Rapid reviews and snowballing methodology*

Because the conjectures arose from practice and the participants' opinions, a set of rapid reviews (RR) were performed, consulting the technical literature and searching for confirmations for these conjectures.

RRs are a type of secondary study aiming to deliver evidence to practice promptly with lower effort than a traditional systematic review. To be faster, RR simplifies some steps of systematic reviews. For instance, the database search is limited, the quality appraisal is eliminated, or only one researcher is used to analyze the collected data [Tricco et al. 2015].

Eight RRs were conducted following the same protocol template, but core parts were replaced to guide each RR to target a specific conjecture. The protocols used by each RR are presented in Appendices O-V. Table 14 shows the terms representing each conjecture and the keywords related to them.

Table 14 - Rapid reviews research questions structure

Conjecture	Term	Keywords
C01	suitable environment	Awareness OR recognition OR understanding OR comprehension OR importance OR relevance
C02	cross-functional team	Team* OR Staff* OR "Working Group"
C03	greater precision on the requirements definition	Requirement*
C04	suitable support tools	"support tool"
C05	suitable environment	environment*
C06	suitable methodology	methodolog*
C07	suitable planning	planning OR plan
C08	reuse of artifacts and knowledge	reuse OR reusability OR reusing

The research questions followed the structure presented in Table 14 by replacing *<<CONJECTURE>>* by the term representing the conjecture. For instance, the RR related to the verification environment had *<<CONJECTURE>>* replaced by a "*suitable environment*."

Table 15 - Rapid reviews research questions structure

RR-RQ 1	What are the benefits of a *<<CONJECTURE>>* for the verification of security and performance?
RR-RQ 2	What problems do cause a *<<CONJECTURE>>* for the verification of security and performance?
RR-RQ 3	What are the challenges of creating a *<<CONJECTURE>>* for the verification of security and performance?
RR-RQ 4	What are the strategies to create a *<<CONJECTURE>>* for the verification of security and performance?

The search string followed the same principle of RQs. A search string template was defined and was adapted to target each conjecture. Table 12 presents the search string template used in each RR. For instance, in the search for conjecture C03, *<<KEYWORD>>* was replaced by *Requirement**, for C05 it was replaced for *environment** and so on.

Template	("security verification" OR "performance verification" OR "security testing" OR "performance testing") AND (*<<KEYWORD>>*) AND ("software") AND ("benefit*" OR "problem*" OR "challenge*" OR "strateg*" OR "empirical stud*" OR "experimental stud*" OR "experiment*" OR "case stud*" OR "survey*")

After a search on the Scopus search engine, the following criteria guided the process of paper selection:

- **Inclusion criteria**
  - The paper must be in the context of software engineering; and
  - The paper must be in the context of performance and/or security verification; and
  - The paper must report a study related to *<<CONJECTURE>>* of security or performance verification activities; and
  - The paper must report an evidence-based study grounded in empirical methods (*e.g.*, interviews, surveys, case studies, formal experiment, among others) or a proof of concept; and

- The paper must provide data to answer at least one of the RR research question; and
- The paper must be written in the English language.

After data extraction, it was performed a snowballing to increase the literature coverage. The snowballing was backward with only one interaction, and the starter set was the included papers of the RRs. Table 16 presents the total number of papers founded, included in RR, and included in snowballing.

Table 16 - Number of papers of RR and snowballing

Conjecture	# Founded	# RR	# Snowballing	# Total
C01	63	2	0	2
C02	42	3	0	3
C03	129	6	8	14
C04	185	12	5	17
C05	117	3	1	4
C06	77	4	9	13
C07	41	3	0	3
C08	11	2	0	2

After, we imported the extraction forms in the MAXQDA tool and performed a coding and a thematic analysis process similar to the process described in Section 4.1.1.2. Thus, the output of the RRs is a set of mind maps with high-level themes representing the findings from the technical literature.

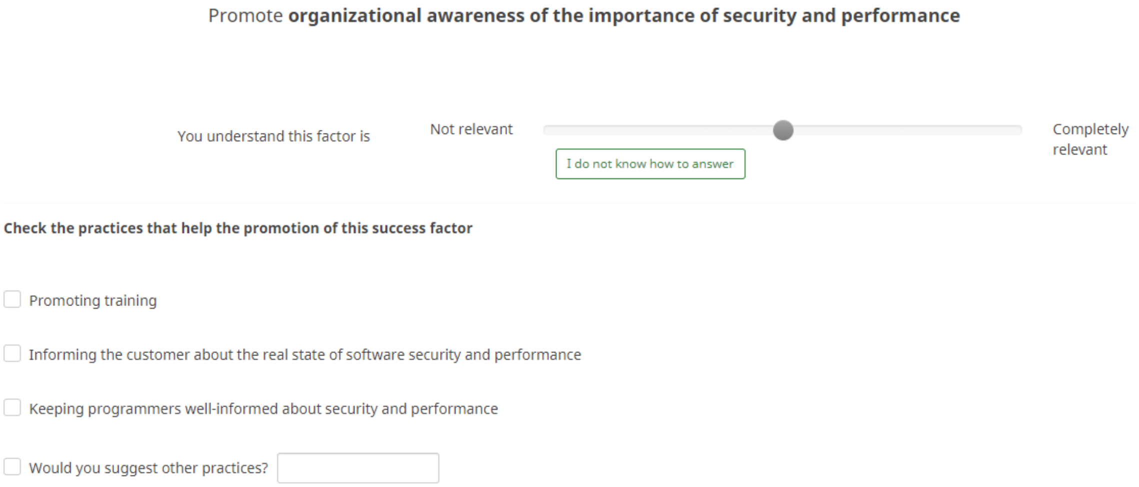
Finally, the themes of the RRs were matched to the themes of the case study. Thereby, the findings that emerged from the state of the practice are supported by the findings extracted from state of the art. Accordingly, credence was lent to the conjectures, turning them into security and performance verification moderator factors.

5.1.2 Survey methodology

Appendix W presents the survey planning protocol, including the followed strategy to recruit participants and the survey questions. The objectives of the survey are twofold, but using the same questions – the first one aims to validate our understanding regarding the information supporting the moderator factors and the actions used to promote them. As this information was collected through the case study, people who already participated in the case study were used as the subjects. In this validation phase, it was possible to get answers from four participants.

The second objective of the survey is to assess the relevance of the moderator factor and the actions used to promote them. Thus, we presented the survey to the software development community using blog posts, social networks, software testing communities, questions-answers services, e-mail groups, and direct private messages. Thus, the survey was available for one month (July 2019), including 37 valid answers from different participants.

The survey used the VAS scale [Wewers and Lowe 1990] to capture participants' opinions regarding the relevance of moderation factors and yes/no questions to capture participants' opinions regarding the relevance of actions. Figure 26 presents an example of a question.



Promote **organizational awareness of the importance of security and performance**

You understand this factor is

Check the practices that help the promotion of this success factor

□ Promoting training

□ Informing the customer about the real state of software security and performance

□ Keeping programmers well-informed about security and performance

□ Would you suggest other practices?

Figure 26 - Example of a question of the survey

5.1.2.1 Characterization of the population

The survey included a characterization section to get information about the participants and organizations they work. Such characterization is vital to understand the context the moderator factors can be applied.

It was possible to obtain responses from 12 countries, but most of them come from Brazil. Besides, 7 participants did not answer about the country they live in (Figure 27).

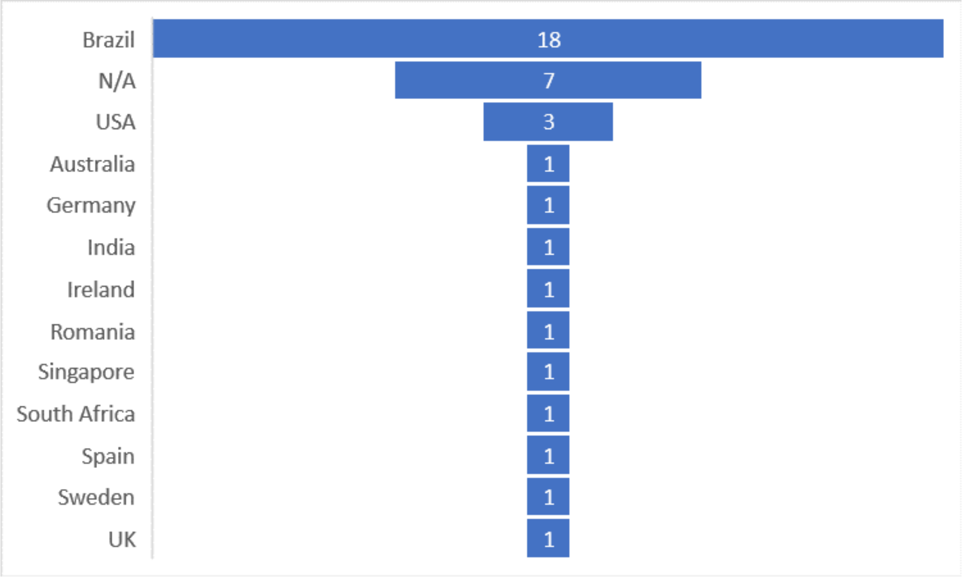


Figure 27 - Survey participants country

We were looking for practitioners who are involved in S&P verification activities, but they could play another role in the organization in which they work. Then we gather information about the primary role played by the participants. As can be seen in Figure 28, most of the participants are programmers, testers, or quality analysts.

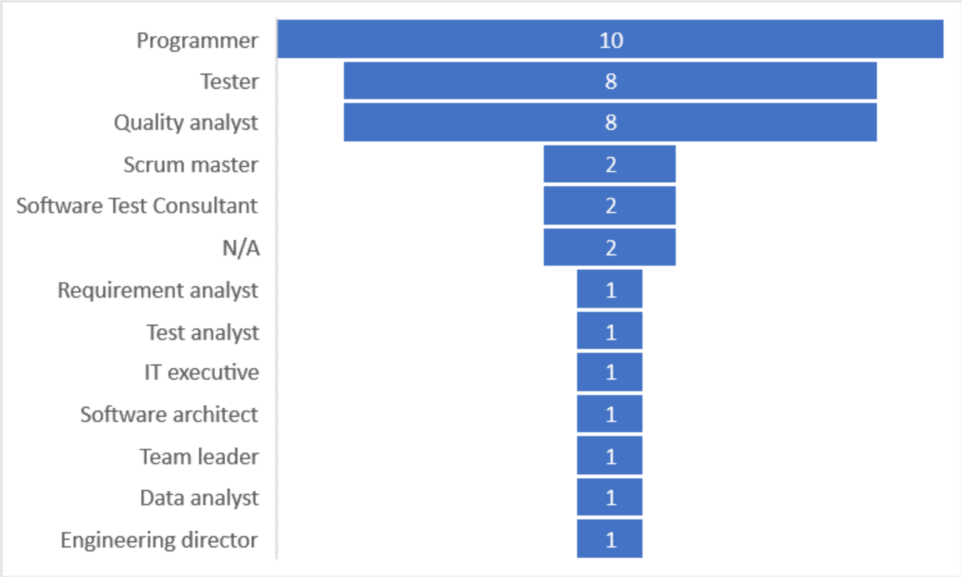


Figure 28 - Survey participants primary role

We also asked the participants for their experience in software development (Figure 29). The results are divided according to the quartiles. Besides, 5 participants did not answer this question.

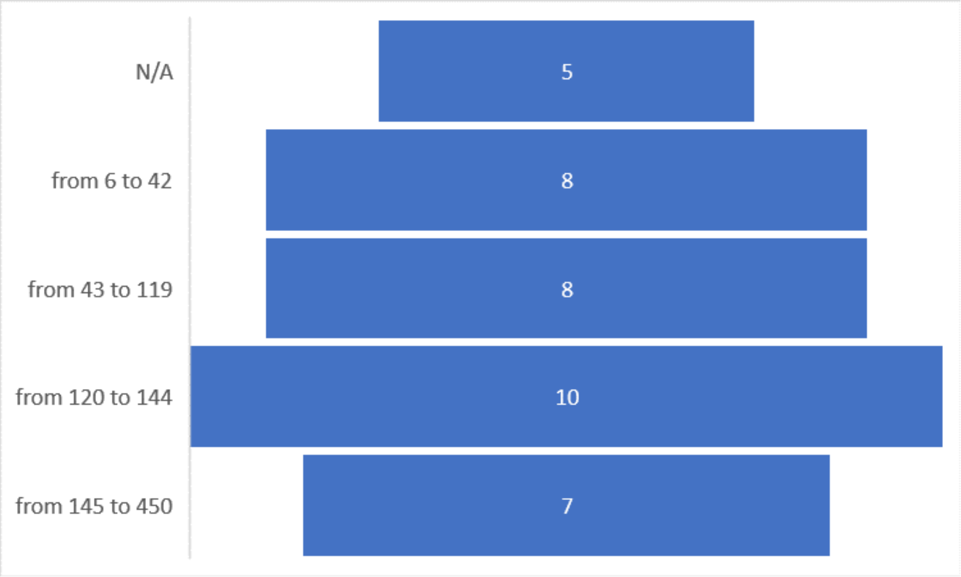


Figure 29 - Survey participants experience (months)

The size of the organizations (the number of employees) is also divided according to the quartiles. Most of the organizations (22) have from 2 to 100 employees. However, 5 participants did not answer the organization size.

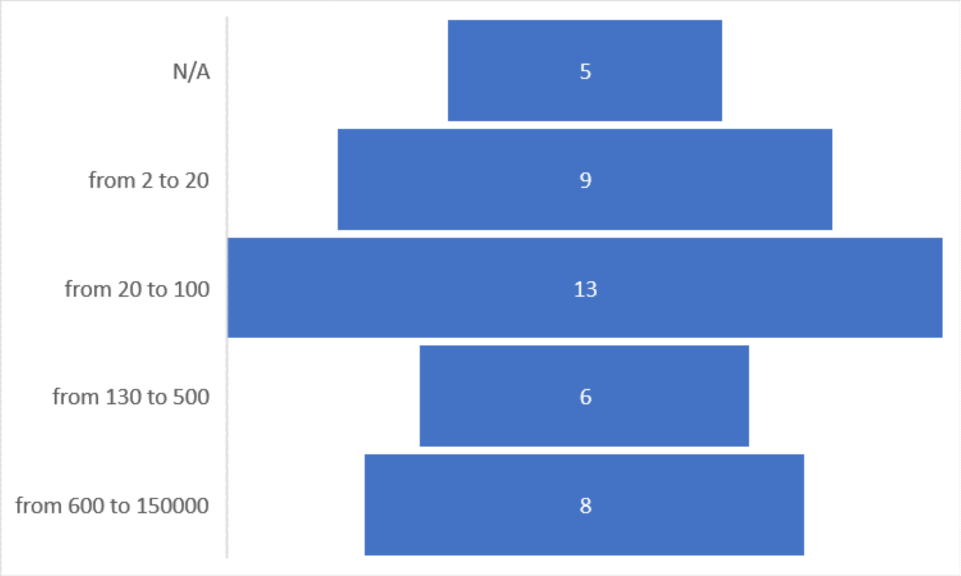


Figure 30 - Organizations size (number of employees)

As shown in Figure 31, it was possible to collect the opinion of participants working in organizations that develop software to distinct domains. The banking domain is the most frequent. It can indicate that the people working in this kind of organization care more about the S&P requirements. As some organizations develop software for different domains, the total is different from 37.

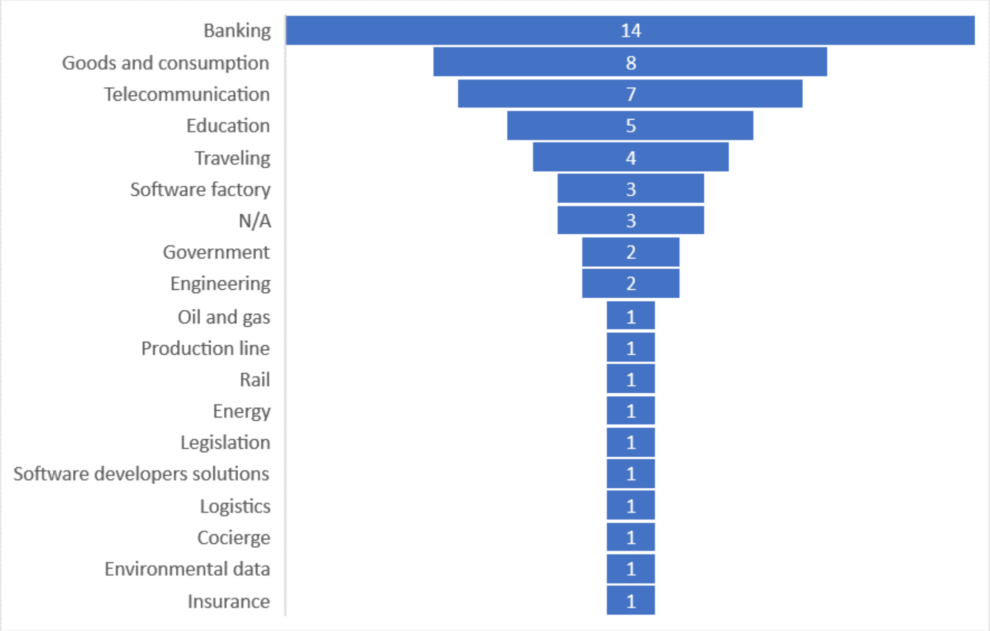


Figure 31 - Organizations domain

Finally, we asked the participants regarding the agile practices used in their organizations. Such information can be used to understand better the context where the participant works (Figure 32).

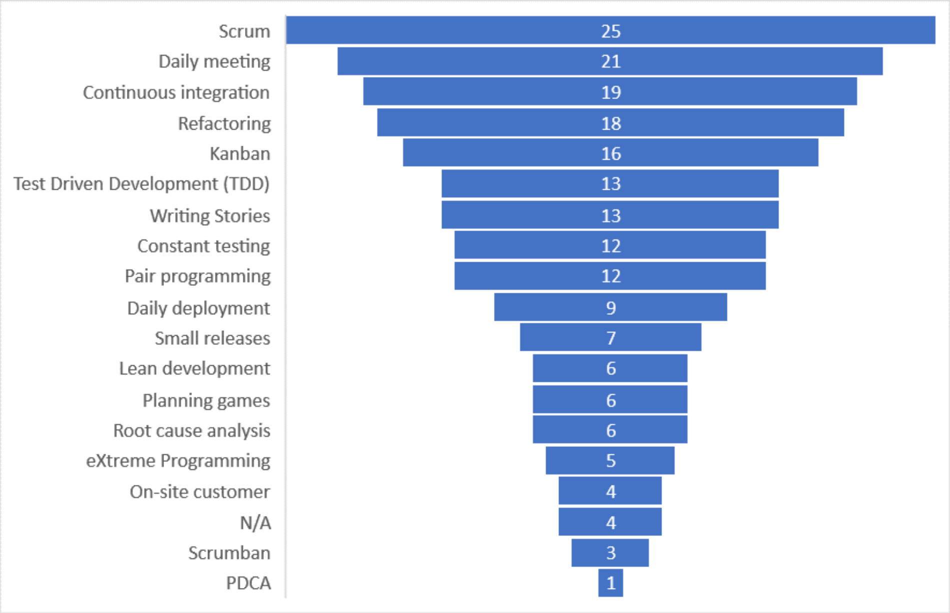


Figure 32 - Level of agility of the organizations

5.2 Introduction

A set of nine conjectures emerged from the observation of the practice (case study). At that time, we realized these conjectures represented significant findings regarding S&P verification, but we did not have proper confidence regarding them. Thus, we decided to perform an in-depth investigation about such conjectures aiming to endorse their confidence. In this way, a set of rapid reviews and snowballing (Section 5.1.1) was performed to identify works supporting these conjectures in the technical literature. Thus, it was possible to rearrange the conjectures and improve their confidence. Therefore, they have presented now as *eight moderator factors of security and performance verification* (MF1-8).

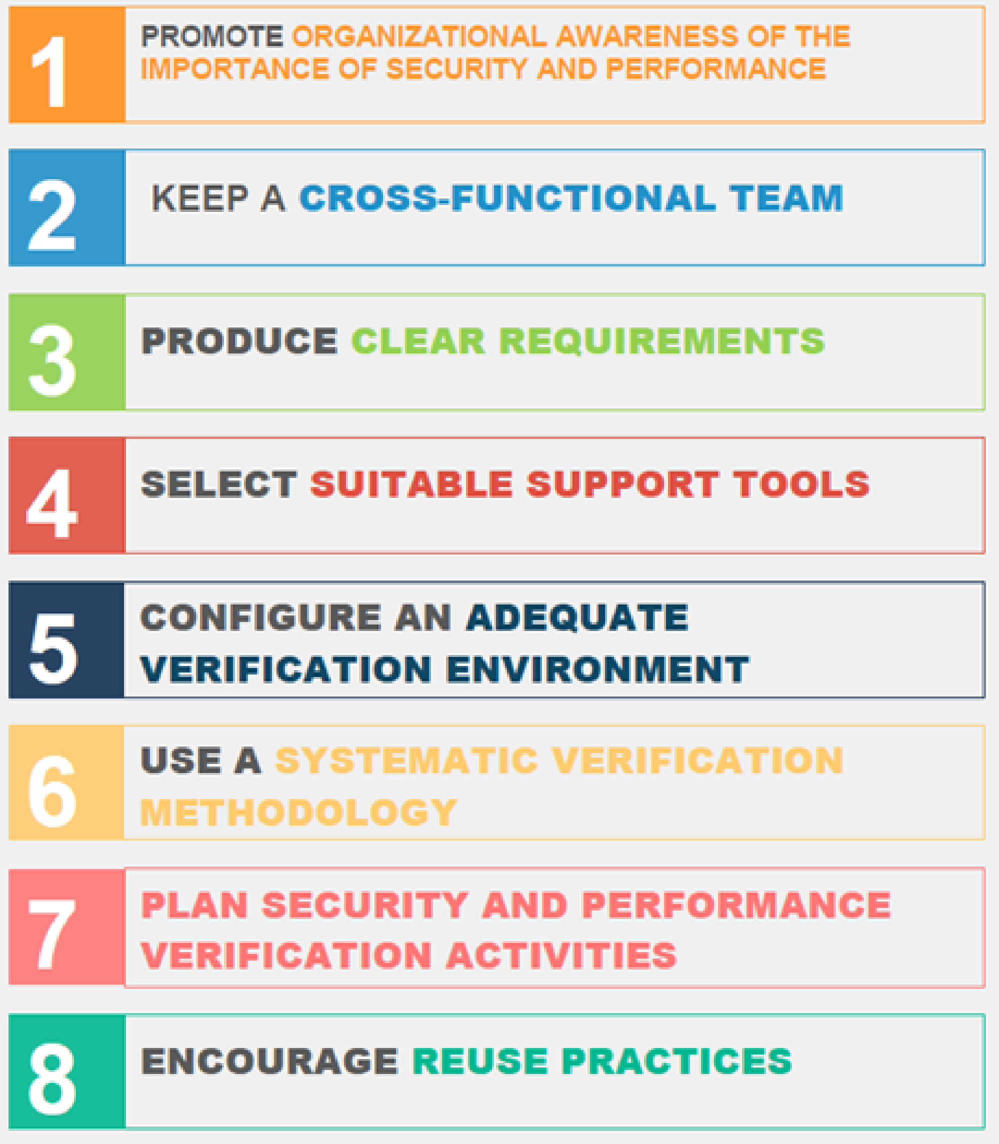
Next, we make use of a survey (Section 5.1.2) to collect the opinion of practitioners regarding the relevance of moderator factors. The participants of the survey were divided into two groups - the control group, composed of people who had already participated in the case study, and another group of external participants. The control group's answers aimed to confirm our understanding and interpretation of the information collected during the case study. The answers of the external participants were used to identify the relevance of the moderator factors and the actions used to promote them.

Besides, the survey allowed us to identify new actions to promote each of the moderator factors. These actions are also presented in this thesis, but they should be further evaluated to increase their understanding and assess their relevance.

It is essential to mention that we use the visual analog scale (VAS) that is used to capture participants' perceptions of a particular event [Wewers and Lowe 1990]. The VAS consists of a horizontal line with two anchor points – the left point indicates the absence of the observed event and the right point the total agreement of the observed event. Appendix W presents the survey plan.

The eight moderator factors (Figure 33) can be seen as confirmed recommendations that a software development organization should meet to perform S&P verification activities successfully. The next sections describe the moderator factors, and the actions can be performed to promote them. It is important to note that the set of actions is not intended to be complete. New actions can be identified through future research focusing on a specific moderating factor.

The moderator factors are also available as a technical report in Portuguese[6] and English[7].



1 PROMOTE ORGANIZATIONAL AWARENESS OF THE IMPORTANCE OF SECURITY AND PERFORMANCE

2 KEEP A CROSS-FUNCTIONAL TEAM

3 PRODUCE CLEAR REQUIREMENTS

4 SELECT SUITABLE SUPPORT TOOLS

5 CONFIGURE AN ADEQUATE VERIFICATION ENVIRONMENT

6 USE A SYSTEMATIC VERIFICATION METHODOLOGY

7 PLAN SECURITY AND PERFORMANCE VERIFICATION ACTIVITIES

8 ENCOURAGE REUSE PRACTICES

Figure 33 - Moderator factors of security and performance verification

[6] http://lens-ese.cos.ufrj.br/spvsurvey/moderators-presentation-ptbr.pdf

[7] http://lens-ese.cos.ufrj.br/spvsurvey/moderators-presentation.pdf

5.3 MF1: Organizational awareness of the importance of security and performance

Figure 34 presents the main points of MF1, and the reasoning regarding this moderator factor is discussed in the following.

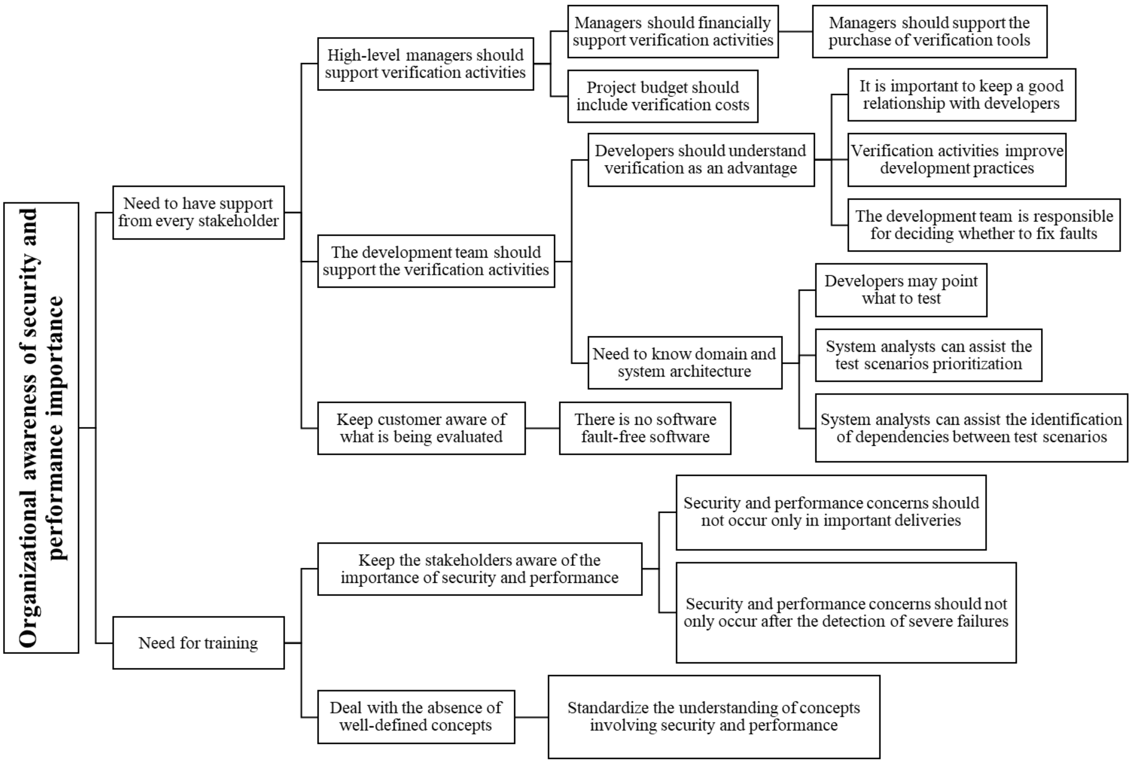

Organizational awareness of security and performance importance
Need to have support from every stakeholder
High-level managers should support verification activities
Managers should financially support verification activities
Managers should support the purchase of verification tools
Project budget should include verification costs
The development team should support the verification activities
Developers should understand verification as an advantage
It is important to keep a good relationship with developers
Verification activities improve development practices
The development team is responsible for deciding whether to fix faults
Need to know domain and system architecture
Developers may point what to test
System analysts can assist the test scenarios prioritization
System analysts can assist the identification of dependencies between test scenarios
Keep customer aware of what is being evaluated
There is no software fault-free software
Need for training
Keep the stakeholders aware of the importance of security and performance
Security and performance concerns should not occur only in important deliveries
Security and performance concerns should not only occur after the detection of severe failures
Deal with the absence of well-defined concepts
Standardize the understanding of concepts involving security and performance

Figure 34 - MF1: Organizational awareness of security and performance importance

Security and performance should not be the responsibility of a separate organization department. The global organizational perception of the importance of these software properties affects verification activities. Thus, S&P verification activities require the *support of every stakeholder*:

- *High-level managers* should financially support S&P activities. For instance, they should support the *purchase of verification tools* and *include the costs of S&P verification in the project budget planning*;
- The support of the *development team* (*e.g.*, programmers, system analysists, and system architects) is also necessary. They should consider security and performance verification an advantage, understanding that if the verification team reports a failure, this is not against the project. Furthermore, the

development team has in-depth knowledge of domain and software architecture. Thus, they can aid decision-making regarding what should be verified, the prioritization of verification scenarios, and the identification of the verification scenario dependencies;

- The *project customer* should also be aware of the importance of security and performance, understanding the importance of S&P verification activities. Moreover, customers should understand that S&P verification activities do not ensure a fully secure system or a system with no performance issues.

Continuous *training* is required to keep stakeholders up to date about the importance of security and the performance of developed software systems. Participants reported that organizations generally do not pay proper attention to system security and performance properties, considering security and performance verification a waste of resources. Usually, organizations are inclined to be *concerned about the security and performance of their systems only after they have a problem*. Another situation where organizations invest more in security and performance assessment is *before a major release* in which a security or performance failure could negatively affect the organization's image. One of the participants said: “*By my own experience working in different places, what I see is that people only care about testing when a problem occurs.*”

Besides, training is also essential to normalize the concepts related to security and performance, furthering the communication between different stakeholders.

5.3.1 Strength of organizational awareness moderator factor

We could observe MF1 in three organizations. It was identified through observation and mentioned by five participants, as shown in Table 17.

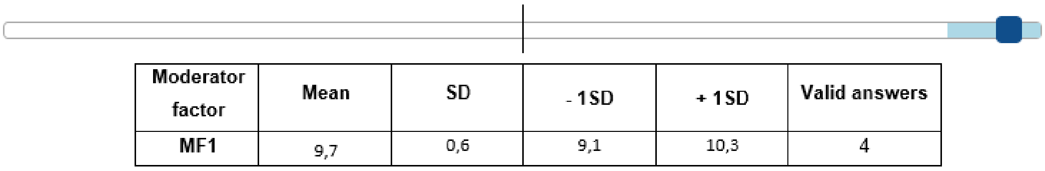
Table 17 - Strength of organizational awareness (MF1)

Organizations	Participants	Quotes
O1	Observation; P4; P5	27
O2	P1; P2	
O4	P1	

Besides, it was possible to identify studies supporting this moderator factor in the technical literature. Horký *et al.* [2015] report an experiment demonstrating that keeping programmers well-informed about performance can decrease the number of bad decisions influencing system performance. Moreover, Ferrell and Oostdyk [2010] emphasize the challenge of security awareness: programmers are not concerned about security because they have a false impression that new development technologies are immune to security problems.

5.3.1.1 Survey results of MF1

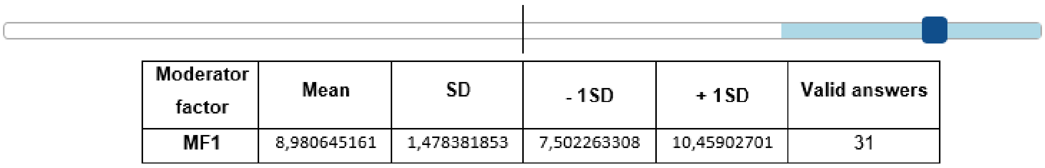
Figure 35 summarizes the opinion of the control group regarding MF1. The dark blue square represents the position of the mean and the area filled with light blue -1 and +1 standard deviation. It is possible to conclude that the control group agrees with our conclusion stating that *organizational awareness of the importance of security and performance* is a factor influencing the security and performance verification.



Moderator factor	Mean	SD	- 1SD	+ 1SD	Valid answers
MF1	9,7	0,6	9,1	10,3	4

Figure 35 - MF1 confirmation (control group)

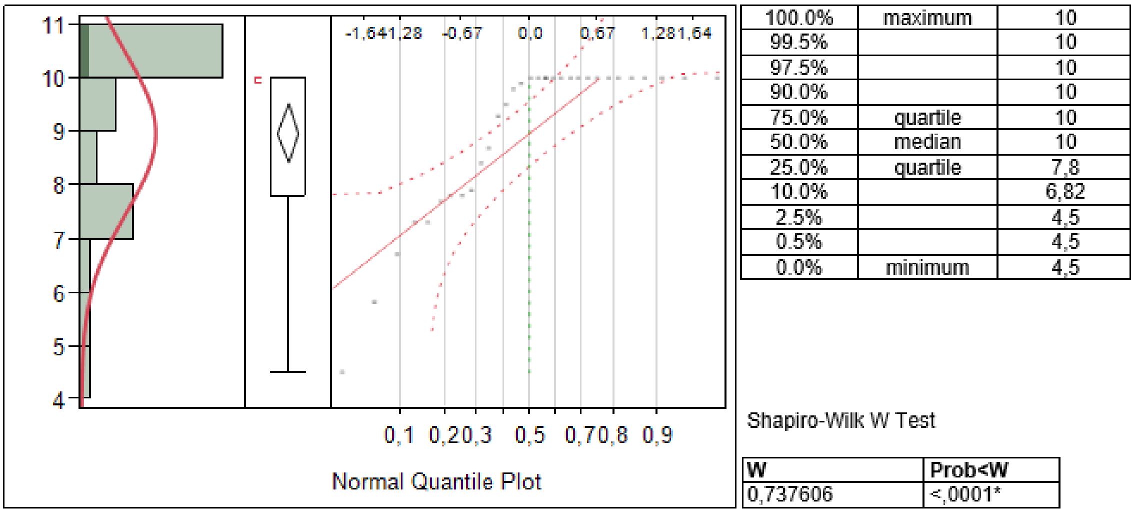
Figure 36 presents the relevance of MF1 according to the opinion of the external participants. The results plotted on the VAS scale graphically show that this moderation factor has high relevance for security and performance verification activities.



Moderator factor	Mean	SD	- 1SD	+ 1SD	Valid answers
MF1	8,980645161	1,478381853	7,502263308	10,45902701	31

Figure 36 - MF1 relevance according to external participants

Besides, the Shapiro-Wilk test (Figure 37) shows the answers do not follow a normal distribution, as most of the responses are not proportionally concentrated around the mean, but at the highest level of the VAS scale. It indicates that most of the participants understand MF1 as relevant. Besides, the median is 10, allowing to state that at least half of the participants understand that this moderation factor has maximum relevance.



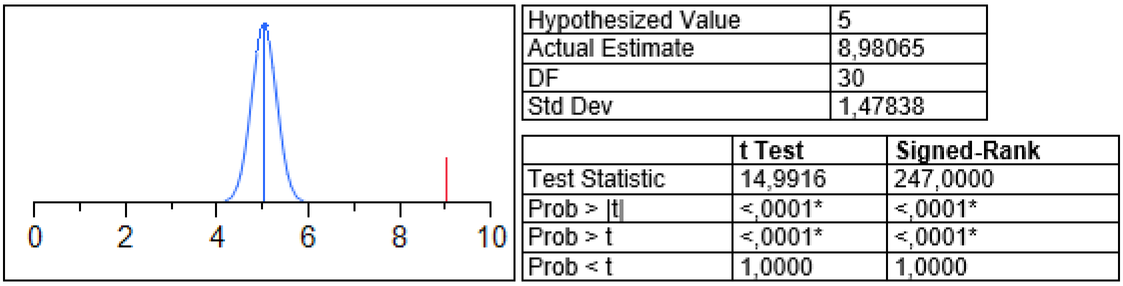
100.0%	maximum	10
99.5%		10
97.5%		10
90.0%		10
75.0%	quartile	10
50.0%	median	10
25.0%	quartile	7,8
10.0%		6,82
2.5%		4,5
0.5%		4,5
0.0%	minimum	4,5

Shapiro-Wilk W Test

W	Prob<W
0,737606	<,0001*

Figure 37 - MF1 distribution

Besides, we tested whether the result was biased owing to the way the questions were presented to the participants. Thus, we performed a mean test hypothesizing a distribution with mean five because the VAS scale was presented to the participants in the central position. The test showed that there was no bias in the responses (Figure 38).



Hypothesized Value	5
Actual Estimate	8,98065
DF	30
Std Dev	1,47838

	t Test	Signed-Rank
Test Statistic	14,9916	247,0000
Prob > |t|	<,0001*	<,0001*
Prob > t	<,0001*	<,0001*
Prob < t	1,0000	1,0000

Figure 38 - MF1 Bias test (mean test)

5.3.2 Actions to promote organizational awareness

Table 18 highlights the actions that can be used to promote the organizational awareness of S&P importance. The relevance given by survey participants orders the actions. Thirty-one participants answered about MF1 and the actions to promote this moderator factor. The ‘#’ and ‘%’ represents the number, and the percentage of participants agree that the action contributes to the promotion of MF1.

Table 18 - Actions to promote MF1

Actions to promote the *organizational awareness of security and performance importance*	**#**	**%**
Keeping programmers well-informed about security and performance	28	90%
Promoting training	25	81%
Informing the customer about the real state of software security and performance	19	61%

It is vital *to keep programmers well-informed about S&P of the software system they are developing. Thus,* they can understand the consequences of the way they are coding and then improve the coding practices to avoid previous mistakes.

It is essential to *promote training* to the technical staff, the managers, and external stakeholders (customers). The programmers should be aware of intrinsic defects of the technologies used in software development and the coding patterns resulting in failures. Thus, they can avoid the use of defective technologies and improve their capability to build failure-free code.

The managers benefit from training as they improve awareness regarding S&P issues during software development. Thus, they understand the importance of S&P verification, including them in the project development life cycle and the project budget.

Customers do not need to be trained in technical stuff. However, they can take advantage of training because they can understand the importance of S&P properties. Therefore, they can agree more easily to the inclusion of S&P verification activities in the project budget.

It is also essential to *keep the customer informed about the real state of software security and performance* — usually, software development organizations ignore the cost of verification activities in the project budget, offering a cheaper software to the customers. Thus, when a failure occurs, the development organization does not notify the customer. Therefore, the customer thinks that it is all right and keep believing that verification activities are a waste of resources. However, the consequences of S&P failures can be catastrophic.

5.3.2.1 New actions to promote MF1

The survey allows the identification of four new actions to promote the MF1. Further investigations are necessary to confirm the relevance of these new actions. The column '#' represents the number of participants mentioned in action.

Table 19 - New actions to promote MF1

New actions to promote the *organizational awareness of security and performance importance*	#
Simulation of security and performance failures and show business impact	1
Regular meetings to discuss security practices	1
External audit to mitigate human problems	1
Having an ethical hacker would be extremely good for security and creating performance indicators	1

5.4 MF2: Cross-functional teams

Verification activities are not performed in isolation by only one team. They require interaction between different teams as well as different skills. In the case studies, a need was identified for a team of *experts in security verification*, *infrastructure*, *legislation,* and *databases* (Figure 39).

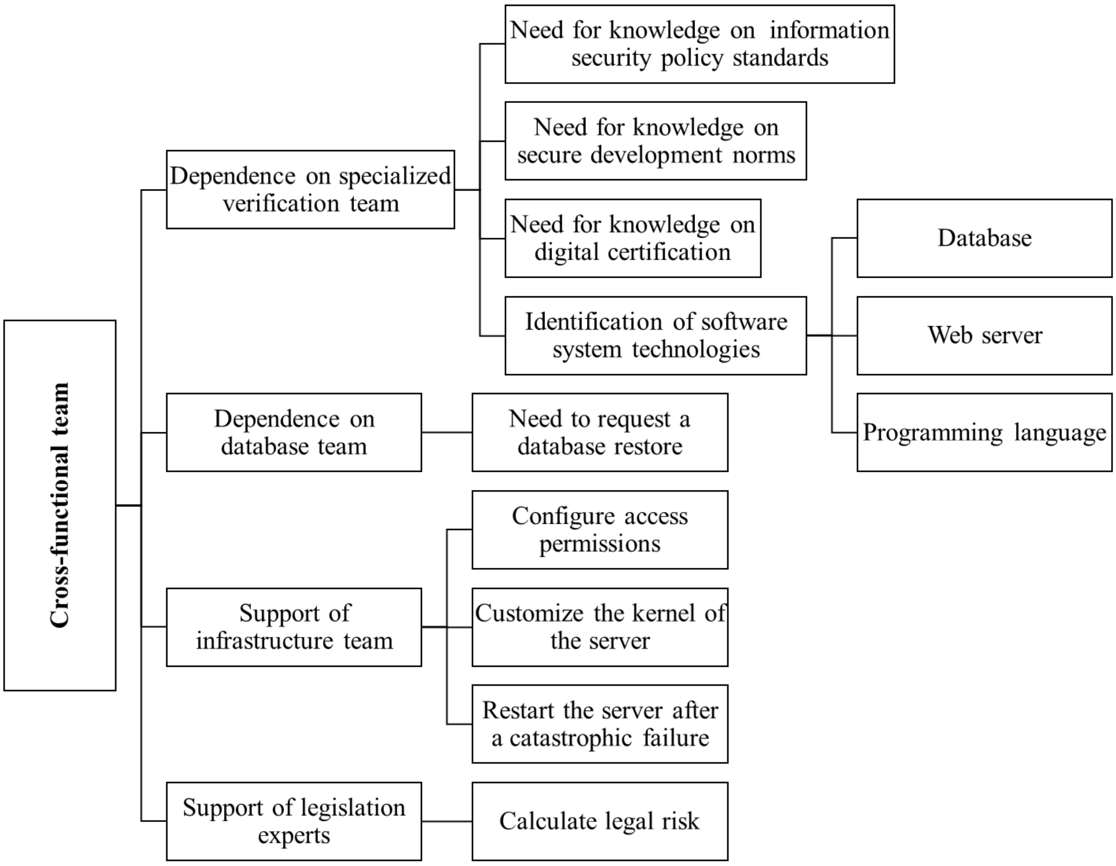

Cross-functional team
Dependence on specialized verification team
Need for knowledge on information security policy standards
Need for knowledge on secure development norms
Need for knowledge on digital certification
Identification of software system technologies
Database
Web server
Programming language
Dependence on database team
Need to request a database restore
Support of infrastructure team
Configure access permissions
Customize the kernel of the server
Restart the server after a catastrophic failure
Support of legislation experts
Calculate legal risk

Figure 39 – MF2: Cross-functional team moderator

Security verification experts are responsible for providing knowledge related to security, such as *information security policies*, *security development standards*, *digital certification*, and cryptography. Furthermore, the security verification team should have the required knowledge to perform the fingerprinting step, *identifying the technologies for developing the software* (database, web server, and programming language).

The support of the *infrastructure team* is outstanding because sometimes the verification teams are unable or do not have the knowledge to take some actions. For instance, it may be necessary to *allow a specific IP* to access the server, *make some changes in the kernel* of the operating system, or *restart the server* after a catastrophic failure. A participant reported a need to interact with an infrastructure specialist: "*...we have to ask to change the (operating system) kernel because it has a limit on the size (of requests) that can be sent...*"

The verification activities may occasionally affect databases irrevocably. In this case, the *database team* may also aid in the verification activities using its knowledge, for instance, to *repair or restore a functional database version*.

A need for *legislation expert* support was also identified. Such an expert can be useful in *assessing legal risks*.

A *cross-team interaction* is also essential to identify the technologies used for software development and how these may influence the verification results. For example, the verification team observes that performance tests of a software scenario result in a decreasing response time. Thus, talking to the development team, they discovered that the system was using the content delivery network (CDN) cache technology.

5.4.1 Strength of cross-functional moderator factor

Issues regarding the need for a cross-functional team were observed in O1, and this was mentioned by four participants from organizations O2 and O4 (Table 20).

Table 20 - Strength of cross-functional team moderator (MF2)

Organizations	Participants	Quotes
O1	Observation	26
O2	P1; P2	
O4	P1; P3	

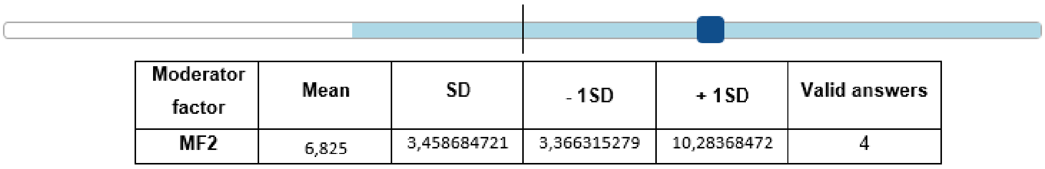
In the technical literature, some studies identify the need for teams with different skills and propose strategies to encourage information exchange between teams [Brucker and Sodan 2014] [Williams et al. 2010] [Johnson et al. 2007].

A card game called protection poker provides security knowledge sharing, involves the entire development team, and increases the awareness of software security needs. Another recommendation is to consider programmers' allies rather than enemies.

Regarding performance, Johnson *et al.* [2007] demonstrate how weekly meetings involving performance architects, domain experts, marketing stakeholders, and developers can improve team interactions.

5.4.1.1 Survey results of MF2

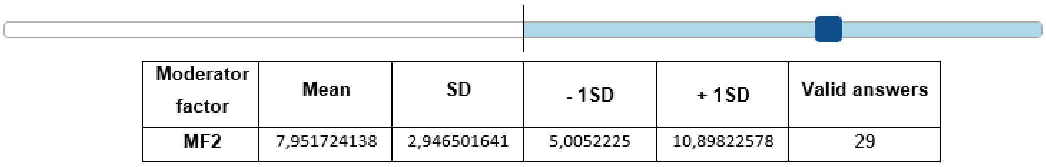
Figure 40 summarizes the opinion of the control group regarding MF2. It is possible to conclude that the control group agrees with our conclusion stating that keeping a cross-functional team is a factor influencing security and performance verification. However, this factor is not unanimous, as the average of the responses is not high, and the standard deviation is wide. It indicates that most participants in the control group agree with MF2, but some participants understand that this factor is not relevant.



Moderator factor	Mean	SD	- 1SD	+ 1SD	Valid answers
MF2	6,825	3,458684721	3,366315279	10,28368472	4

Figure 40 - MF2 confirmation (control group)

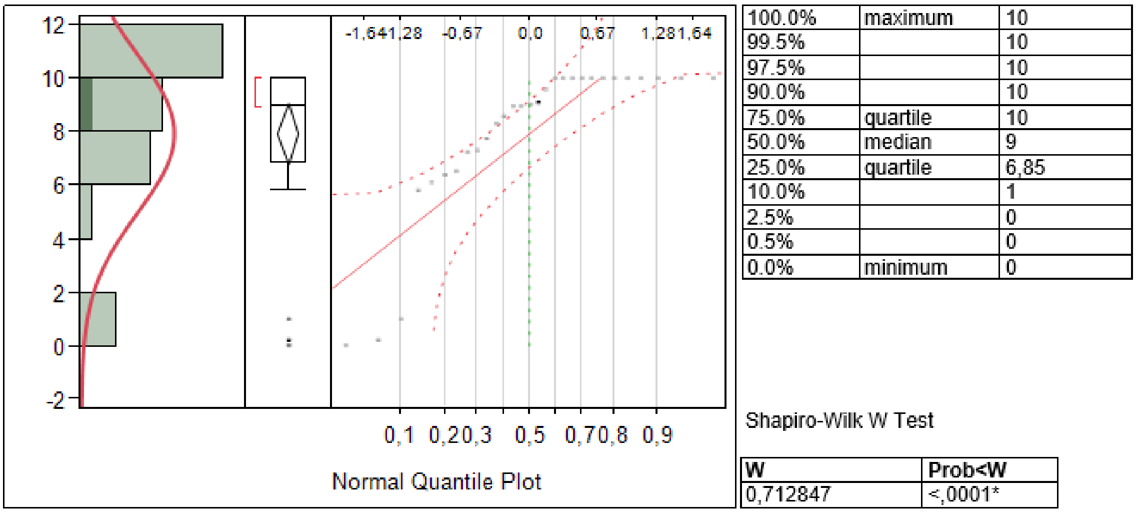
Figure 41 presents the relevance of MF2 according to the opinion of the external participants. The results plotted on the VAS scale graphically show that this moderation factor is relevant for security and performance verification activities. However, the significant standard deviation means that there are practitioners who disagree with the average opinion, understanding that MF2 is more or less relevant compared with the average opinion.



Moderator factor	Mean	SD	- 1SD	+ 1SD	Valid answers
MF2	7,951724138	2,946501641	5,0052225	10,89822578	29

Figure 41 - MF2 relevance according to external participants

Besides, Figure 42 shows the answers does not follow a normal distribution, as most of the responses are not proportionally concentrated around the mean, but at the highest level of the VAS scale. It indicates that most of the participants understand MF2 as relevant.



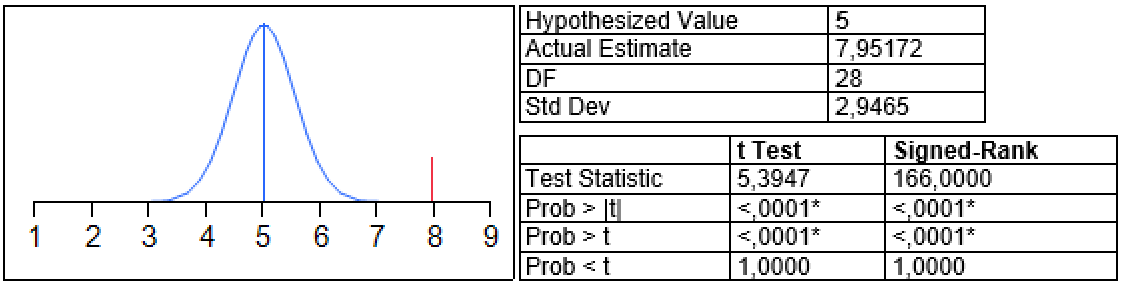
100.0%	maximum	10
99.5%		10
97.5%		10
90.0%		10
75.0%	quartile	10
50.0%	median	9
25.0%	quartile	6,85
10.0%		1
2.5%		0
0.5%		0
0.0%	minimum	0

Shapiro-Wilk W Test

W	Prob<W
0,712847	<,0001*

Figure 42 – MF2 distribution

Also, we tested whether the result was biased owing to the way the questions were presented to the participants. Thus, we performed a mean test hypothesizing a distribution with mean five because the VAS scale was presented to the participants in the central position. As shown in Figure 43, the test confirmed that there was no bias in the responses.



Hypothesized Value	5
Actual Estimate	7,95172
DF	28
Std Dev	2,9465

	t Test	Signed-Rank
Test Statistic	5,3947	166,0000
Prob > |t|	<,0001*	<,0001*
Prob > t	<,0001*	<,0001*
Prob < t	1,0000	1,0000

Figure 43 - MF2 Bias test (mean test)

5.4.2 Actions to promote cross-functional teams

Table 21 presents the actions that can be used to build a *cross-functional team*. The actions are ordered by the relevance, according to the 29 participants that answered about MF2.

Table 21 - Actions to promote MF2

Actions to promote the *build of a cross-functional team*	#	%
Building a team having multiple skills	23	79%

Disseminating the view that the verification team is not the enemy but allied	23	79%
Stimulating interaction between members of different teams	18	62%

Building a team having many skills complements the previous practice. The exchange of knowledge between teams is not so useful if every member has the same set of skills.

Disseminating the view that the verification team is not the enemy helps the verification team work together with the other teams. Otherwise, the verification team could have problems in identifying what should be assessed and requesting failure fixes.

Additionally, stimulating interaction between members of different teams (*i.e.*, programmers, architects, and requirements analysts) is a way of making explicit the capabilities of each team and personal skills. So, when the verification team is faced with a problem that requires specific knowledge to solve, they know whom to ask for help.

5.4.2.1 New actions to promote MF2

It was also possible to identify eight new actions to promote MF2. Table 22 shows the newly identified actions. Column '#' represents the number of participants that mentioned the action.

Table 22 - New actions to promote MF2

New actions to promote the *build of a cross-functional team*	**#**
The team should have leaders swapping places (for example, marketing and development). team leaders can get to know limitations, capabilities, and point of view which can lead to better teamwork and results	1
Highlight the positive results of having a multidisciplinary team	1
Knowing what is problematic in other sectors of the organization	1
Encouraging integration between teams working on similar topics	1
Value verification professionals	1
Select qualified people for the position	1
Invest in the training and qualification of the verification team	1
Apply Scrum	1

5.5 MF3: Suitable requirements

It is crucial to make sure the organization can produce precise S&P requirements. Figure 44 presents the themes of composing MF3.

The *lack of S&P requirements* prevents the verification from fulfilling its original purpose (*i.e.*, assessing whether the software meets its requirements) because, in the *absence of an oracle* it is impossible to know if the verification results are correct.

Moreover, inaccurate requirements result in *teams overloading* (*e.g.*, analysts, architects, and developers) because the verification team must continuously contact them.

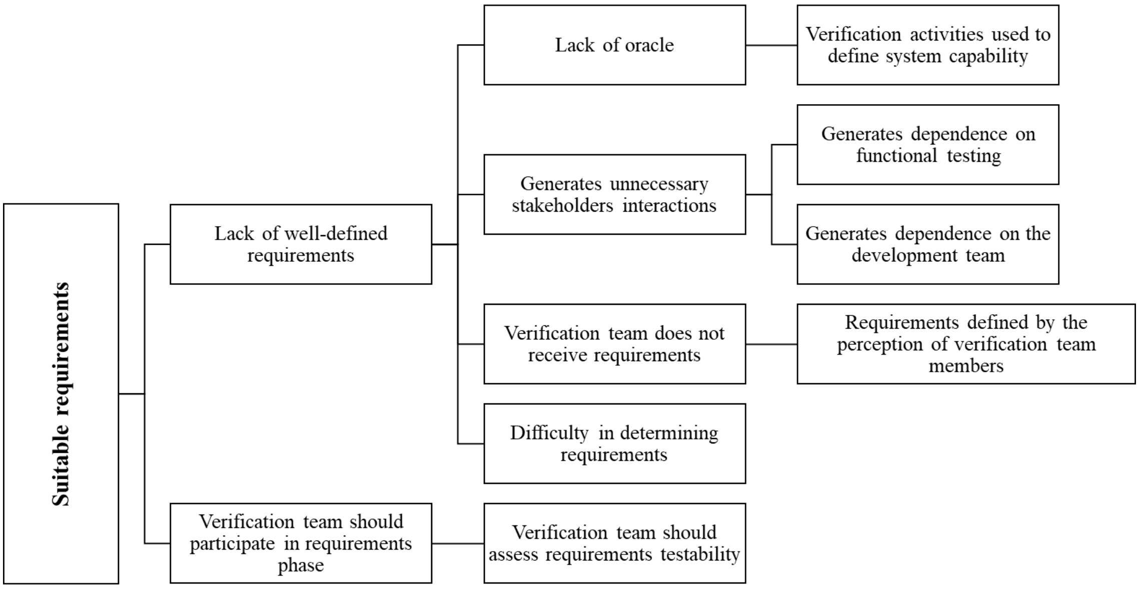

Suitable requirements
Lack of well-defined requirements
Lack of oracle
Verification activities used to define system capability
Generates unnecessary stakeholders interactions
Generates dependence on functional testing
Generates dependence on the development team
Verification team does not receive requirements
Requirements defined by the perception of verification team members
Difficulty in determining requirements
Verification team should participate in requirements phase
Verification team should assess requirements testability

Figure 44 – MF3: Suitable requirements moderator factor

In the organizations used in this study, there were occasionally no written performance requirements that could be used as an oracle. In such cases, the verification activities were not performed to assess whether the software meets its requirements but to *evaluate the capacity of the system*. In other instances, the verification activities were performed based on subjective or imprecise requirements. For example, a participant reported a case where the tests were performed based on a brief description of users' behavior: "*In this system, everyone comes in at 8 in the morning and stays until 10 o'clock. Then they leave the system and come back at lunch*".

The lack of S&P requirements can be dangerous because the *verification team may determine the requirements by their own experience*, which may not reflect customer expectations. Furthermore, it was possible to observe that the verification team has some *difficulties in determining security and performance requirements*.

Participants from two organizations suggest that the verification team must participate in requirement planning, understanding, and evaluating their testability.

5.5.1 Strength of suitable requirements moderator factor

As shown in Table 23, the need for the suitable requirements moderator factor was observed in organization O1, and seven participants from three different organizations mentioned it.

Table 23 - Strength of suitable requirements moderator factor (MF3)

Organizations	Participants	Quotes
O1	Observation; P1; P2; P3	26
O2	P1; P2	
O4	P1; P2	

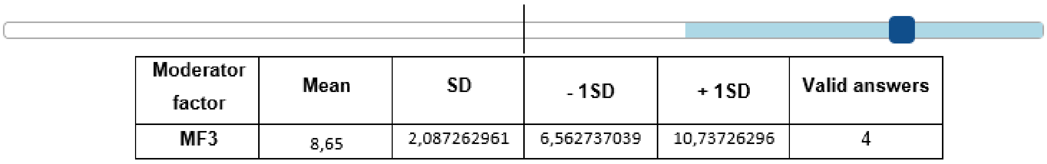
It was possible to identify a set of issues and challenges regarding S&P requirements in the technical literature: lack of support tools and techniques, techniques are not suitable to their users' profile, lacking requirements, and wrong requirement descriptions. These issues make verification impossible, ambiguous, or generic [Harjumaa and Tervonen, 2010] [Tondel et al. 2008] [Stephanow and Khajehmoogahi, 2017] [Weyuker and Vokolos, 2000].

A set of proposed techniques and recommendations to handle security and performance requirements can also be identified: misuse cases, SETAM UMLsec, abuse cases and description of attack patterns [McDermott and Fox 1999] [Harjumaa and Tervonen 2010] [Weyuker and Vokolos 2000] [Sindre and Opdahl 2001] [Hui and Huang 2012] [Jürjens 2002] [Bozic and Wotawa 2014] [Haley et al. 2008] [Bulej et al. 2017].

The availability of the techniques to handle S&P requirements points to the gap between practice and academy, as these techniques are not applied.

5.5.1.1 *Survey results of MF3*

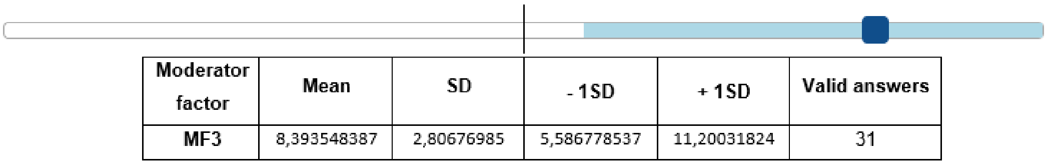
Visualizing the results through the VAS (Figure 45) allows us to confirm our conclusion about the influence of MF3 in the S&P verification. The result shows the agreement of the control group regarding the influence of *suitable requirements* in the S&P verification.



Moderator factor	Mean	SD	- 1SD	+ 1SD	Valid answers
MF3	8,65	2,087262961	6,562737039	10,73726296	4

Figure 45 – MF3 confirmation (control group)

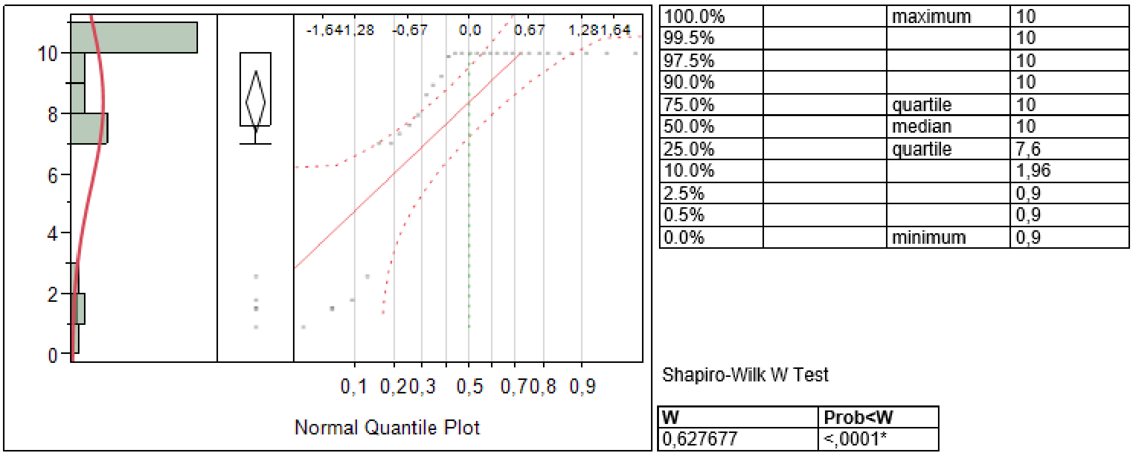
Figure 46 presents the relevance of MF3 according to the opinion of the external participants. The results plotted on the VAS scale graphically show that this moderation factor is relevant for S&P verification activities. Moreover, even having a significant standard deviation, the value -1sd is above the midpoint of the scale, indicating a consensus about the relevance of this factor.



Moderator factor	Mean	SD	- 1SD	+ 1SD	Valid answers
MF3	8,393548387	2,80676985	5,586778537	11,20031824	31

Figure 46 - MF3 relevance according to external participants

Besides, the statistical analysis of the results shows a non-normal distribution (Figure 47), and most of the participants evaluate MF3 with high relevance.



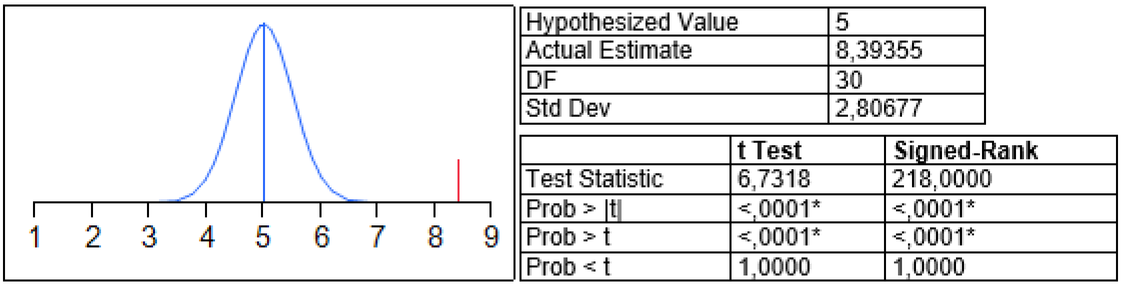
100.0%		maximum	10
99.5%			10
97.5%			10
90.0%			10
75.0%		quartile	10
50.0%		median	10
25.0%		quartile	7,6
10.0%			1,96
2.5%			0,9
0.5%			0,9
0.0%		minimum	0,9

Shapiro-Wilk W Test

W	Prob<W
0,627677	<,0001*

Figure 47 - MF3 distribution

In addition, we tested whether the result was biased owing to the way the questions were presented to the participants. Thus, we performed a mean test hypothesizing a distribution with mean five because the VAS scale was presented to the participants in the central position. As shown in Figure 48, the test confirmed that there was no bias in the responses.



Hypothesized Value	5
Actual Estimate	8,39355
DF	30
Std Dev	2,80677

	t Test	Signed-Rank
Test Statistic	6,7318	218,0000
Prob > |t|	<,0001*	<,0001*
Prob > t	<,0001*	<,0001*
Prob < t	1,0000	1,0000

Figure 48 - MF3 Bias test (mean test)

5.5.2 Actions to promote the building of suitable requirements

Table 24 presents an overview of the actions an organization can perform to promote the building of suitable requirements. We asked the survey participants for the relevance of each action and ordered them by the participant's opinion. Thirty-one participants answered about the relevance of MF3 and the actions to promote it. Column '#' represents the number of participants understanding the action as relevant to promote the MF3 and the column '%' the percentage.

Table 24 - Actions to promote MF3

Actions to promote the building of *suitable requirements*	#	%
Using techniques to handle security and performance requirements	25	81%
Involving the verification team in the requirements phase	24	77%
Stimulating the verification team to assess the testability of requirements	24	77%

Using techniques to identify and represent the S&P requirements is another action the organizations can perform to improve the way they handle S&P requirements. The use of a technique supports the verification team to be more systematic, avoiding misconceptions resulting from subjective decisions. There are different techniques available such as abuse cases, NFR-Framework, Sec-UML.

However, it was possible to realize that these techniques did not reach the software development industry.

Regarding the actions, the organizations can *further the involvement of the verification team in the requirements phase*. Thus, the verification team increases their knowledge in the problem domain, favoring the identification of S&P requirements.

Besides, if the requirements phase includes the verification team members, they can provide criticisms regarding how requirements have been represented, improving the specification, and *assessing the testability of the S&P requirements*.

5.5.2.1 New actions to promote MF3

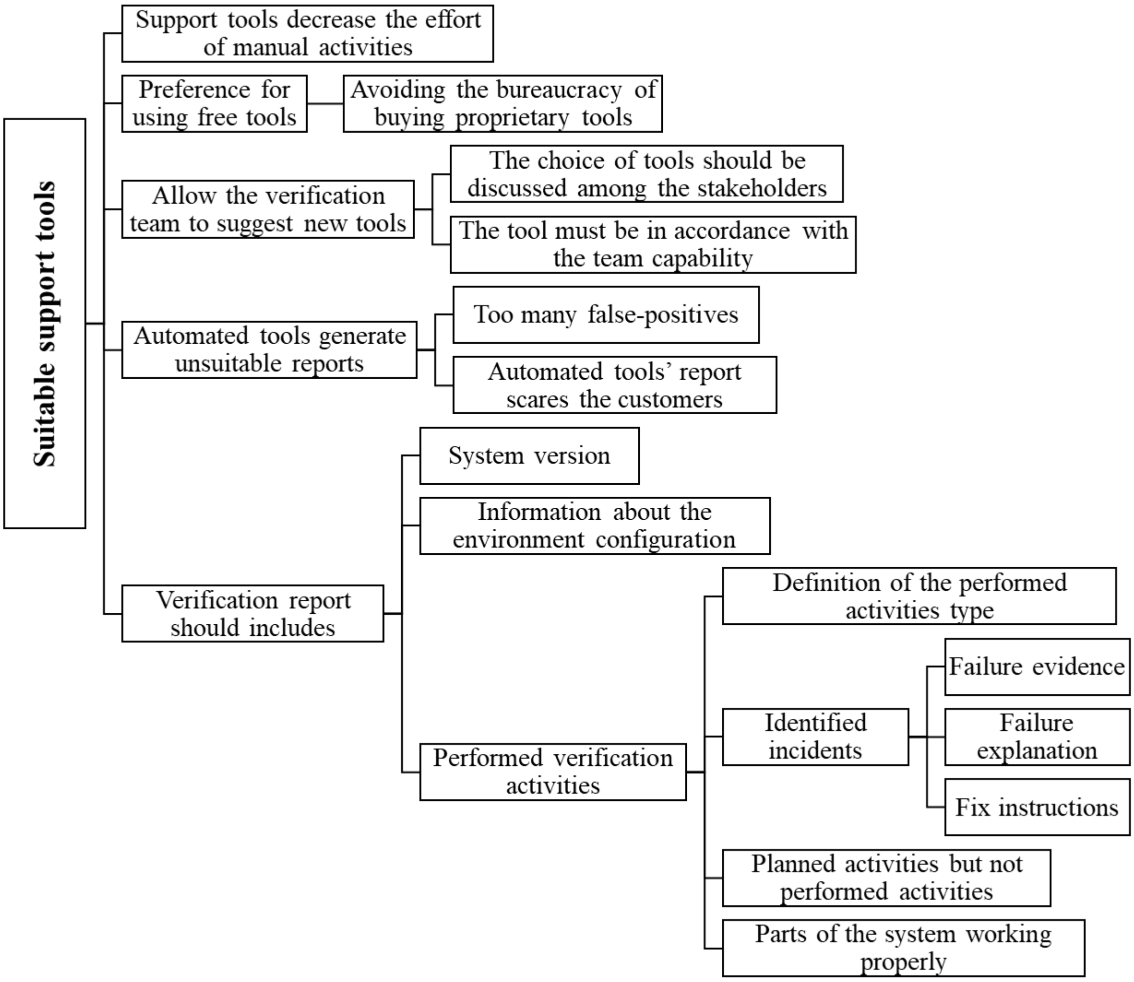
Table 25 presents the actions to promote MF3 that were not previously identified. Initially, we could identify that '*involving the verification team in the requirements phase*' is crucial to produce proper S&P requirements but a participant suggested broader actions stating that it is essential to '*involve the verification team in all phases of the software life cycle.*' Besides, another participant suggested an action with the reverse logic, stating that the *requirements team needs to be involved in the verification activities*. These actions seem to make sense, but they need further researches to improve their understanding and identify their relevance in different contexts.

Table 25 - New actions to promote MF3

New actions to promote the building of *suitable requirements*	**#**
Involving the verification team in all phases of the software life cycle	1
The verification team and Product Owner should discuss the specification in order to identify and adjust any deviations before the specification goes into development.	1
Infrastructure team should assess security and performance	1
Involving the requirements team in verification activities	1

5.6 MF4: Support tools

Figure 49 presents the topics composing the *support tools* moderator (MF4). The use of suitable support tools is essential in S&P verification activities because it can *decrease the effort of manual activities*. It was possible to identify an inclination to choose *free tools* because the *acquisition process is faster*, as it involves the technical team only. In the case of adopting proprietary tools, it is necessary to ask managers for permission, and the price of the tool may hinder or impede the buying process.



Suitable support tools
Support tools decrease the effort of manual activities
Preference for using free tools
Avoiding the bureaucracy of buying proprietary tools
Allow the verification team to suggest new tools
The choice of tools should be discussed among the stakeholders
The tool must be in accordance with the team capability
Automated tools generate unsuitable reports
Too many false-positives
Automated tools' report scares the customers
Verification report should includes
System version
Information about the environment configuration
Performed verification activities
Definition of the performed activities type
Identified incidents
Failure evidence
Failure explanation
Fix instructions
Planned activities but not performed activities
Parts of the system working properly

Figure 49 - MF4: Support tools moderator factor

Moreover, the verification team (technical staff) should have the autonomy to suggest and adopt new support tools. In the choice of these tools, the capacity of the team is an essential decision-criterion as the verification team may not have the necessary knowledge to use advanced features of the tools.

Some findings were identified regarding the tools' reports. The first finding regards the *excessive number of false-positives*. In this case, the development team can ignore the results of verification because it takes substantial effort to analyze each of the reported incidents and classifying them as false-positives or real failures.

Additionally, the tools' report should not be delivered to the customer or the developers immediately. The results can scare people unfamiliar with verification activities. Therefore, these results should be analyzed and processed by the verification team. Thus, a consistent report can be delivered to the customers or the developers. A participant talks about this: "*…tools generate 'cold' reports. My team and I should analyze and consolidate them, making them more understandable for the users, programmers, and managers.*"

It was also possible to identify *vital information that a verification report should contain*. Therefore, a suitable tool should generate reports with this information. First, it should provide the *system version* and *configuration information* because software or environment settings changes may require verification re-execution. It is also necessary to provide information on which *tests have been performed*, defining each of them, and reporting which *incidents were detected*. In the case of incidents, it is necessary to provide *information to replicate the incident*, a *possible explanation,* and *instructions for resolving it*.

Additionally, it is crucial to inform which *tests were planned and not performed* (usually due to deadlines/budget constraints). It makes the customers more aware of the system capability and the possible failures that may occur in the production environment. Finally, it is also essential to make explicit in the *reports the verification activities that did not reveal incidents*. It is psychologically positive for the client or developer to know that the system operates correctly in several aspects.

5.6.1 Strength of suitable support tools moderator factor

Findings supporting this moderator factor were identified in all organizations, as shown in Table 18.

Table 26 - Strength of suitable support tools moderator factor (MF4)

Organizations	Participants	Quotes
O1	Observation; P1; P4; P5	35

O2	P1; P2	
O3	P1	
O4	P1; P2	

In the technical literature, various studies were found stating that the use of support tools is vital for verification activities [Thompson 2003] [Yee 2006] [Johnson et al. 2007]. Some claim that certain types of verification would not be possible without support tools. For instance, long-running tests, significant data volume testing, and concurrent user tests are not feasible without the use of tools [Guo et al. 2010]. Moreover, support tools are essential for specific practices or development methodologies, such as continuous integration [Ferme and Pautasso 2017] and agile software development [Shu and Maurer 2007].

However, it is also important to stress that automated support tools cannot replace manual verification. They are complementary practices because some defects cannot be identified by current support tools [Johnson et al. 2007] [Dukes et al. 2013].

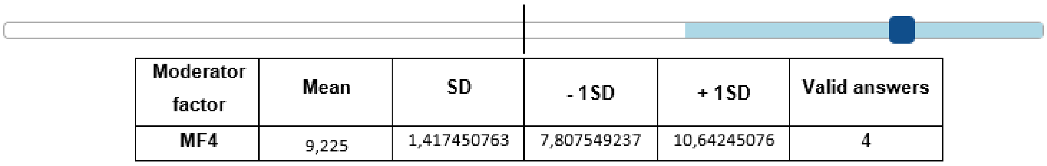
Some studies highlight the lack of suitable support tools, reporting issues related to the need for integration of different tools [Guo et al. 2010] [Barbir et al. 2007], the high cost of proprietary tools [Kabbani et al. 2010], the need of experimental evaluation, the lack of standard-compliant tools [Türpe 2008], and lack of support tools targeting specific technologies [Shu and Maurer 2007] [Barbir et al. 2007] [Kim et al. 2009] [Parveen and Tilley 2008].

Additionally, various studies emphasize the excessive false-positives generated by current tools [Türpe 2008] [Luo and Yang 2014] and consider this a criterion to choose a suitable support tool [Shu and Maurer 2007] [Zhioua et al. 2014].

Finally, Türpe [2008] presents some requirements for useful support tools, also confirming our findings: in line with the team’s capability, idealized site conditions should not be required, and the right problems should be addressed.

5.6.1.1 Survey results of MF4

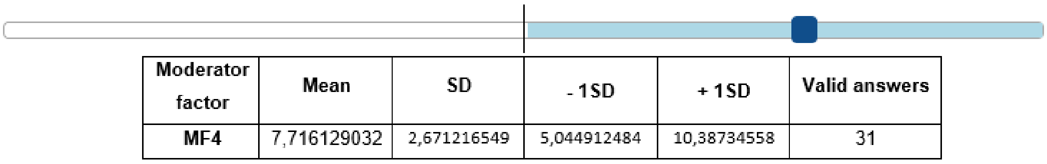
The results of the survey show that the control group agrees with the relevance of the moderator factor support tools (Figure 50). Therefore, we can conclude that our understanding of MF4 is genuine.



Moderator factor	Mean	SD	- 1SD	+ 1SD	Valid answers
MF4	9,225	1,417450763	7,807549237	10,64245076	4

Figure 50 - MF4 confirmation (control group)

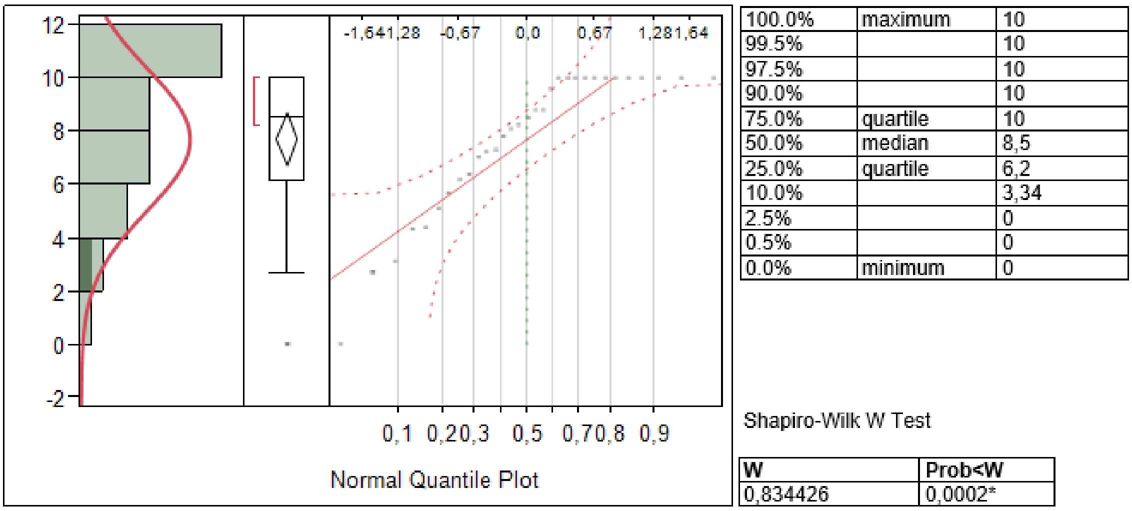
When asking the external participants about the relevance of the *use of suitable tools* in S&P verification activities, the results were also satisfactory. Figure 51 shows the mean is ~7.71, and the VAS scale shows that external participants have the perception that MF4 has high relevance to verification activities.



Moderator factor	Mean	SD	- 1SD	+ 1SD	Valid answers
MF4	7,716129032	2,671216549	5,044912484	10,38734558	31

Figure 51 - MF4 relevance according to external participants

The statistical analysis of the survey answers shows that most participants attributed the maximum value to the relevance of this moderating factor (Figure 52). Thus, it is possible to conclude that the use of suitable tools is relevant to S&P verification in different contexts.



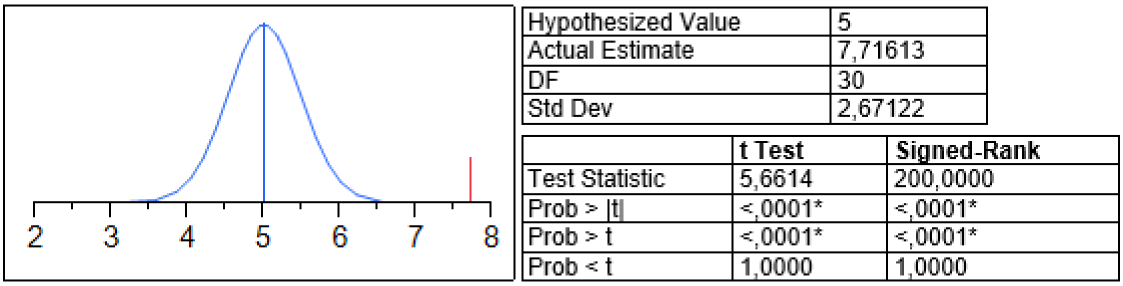
100.0%	maximum	10
99.5%		10
97.5%		10
90.0%		10
75.0%	quartile	10
50.0%	median	8,5
25.0%	quartile	6,2
10.0%		3,34
2.5%		0
0.5%		0
0.0%	minimum	0

Shapiro-Wilk W Test

W	Prob<W
0,834426	0,0002*

Figure 52 - MF4 distribution

Besides, we performed a test aiming to check if the answers were biased. We used the hypothesized value five because the VAS scale was initially displayed at the middle position. However, it did not reveal bias (Figure 53).



Hypothesized Value	5
Actual Estimate	7,71613
DF	30
Std Dev	2,67122

	t Test	Signed-Rank
Test Statistic	5,6614	200,0000
Prob > |t|	<,0001*	<,0001*
Prob > t	<,0001*	<,0001*
Prob < t	1,0000	1,0000

Figure 53 - MF4 Bias test (mean test)

5.6.2 Actions to promote the selection of suitable support tools

A software development organization can use some practices to promote the selection of suitable support tools. Thirty-one participants of the survey answered about MF4, allowing the ordering of the actions by their relevance (Table 27).

Table 27 - Actions to promote MF4

Actions to promote the selection of *suitable support tools*	#	%
Allowing the technical team to suggest and adopt support tools	24	77%
Using tools consistent with the verification team knowledge	22	71%
Supporting the use of free tools	13	42%

It is crucial to *allow the technical team to suggest and adopt support tools*. As intrusion practices evolve constantly, it is necessary to replace the used tools with a new one or a new version. The verification team is composed of people having the skills to evaluate if a tool is outdated and to suggest the use of another.

It is also essential to *use tools consistent with the verification team's knowledge*. An inexperienced verification team using a tool having a lot of sophisticated features get lost and cannot correctly perform the activities. The opposite is also not recommended. A very experienced team using tools that have a limited set of features cannot apply all their knowledge and can be considered a waste of resources.

Besides, the *usage of free tools should be encouraged*. This practice allows the technical team to make decisions regarding the choice of tools without necessarily involving the management team. It makes the process of selecting and changing the used tools more agile because it avoids the bureaucracy of buying a proprietary tool. However, this action had low relevance, according to the survey's participants. It would mean that there is no preference between the use of a free or proprietary tool in the context of some organizations. We can hypothesize that the cost of a verification tool may be irrelevant to organizations with high economic power, or that some organizations have a light purchase tools process, reducing the difficulty of acquiring a proprietary tool.

5.6.2.1 New actions to promote MF4

The survey results revealed new practices that can be used to promote the choice of appropriate tools. It is essential to highlight that five of the participants suggested providing training on the tools adopted by an organization. Thus, this action is stronger than the others.

Table 28 - New actions to promote MF4

New actions to promote the selection of *suitable support tools*	**#**
Providing training to the verification team to enable them to operate the adopted tools	5
Institutionalize the use of tools	1
Using industry best-practice toolsets	1
Support from the tool provider	1

5.7 MF5: Adequate verification environment

A suitable environment is essential for verification. In this context, the environment encompasses both the configurations of the infrastructure responsible for system operation (*e.g.*, application server and database parameters) and the configuration of the system itself (*e.g.*, the data stored in the database while verification activities are performed). Figure 54 presents the themes of composing MF5.

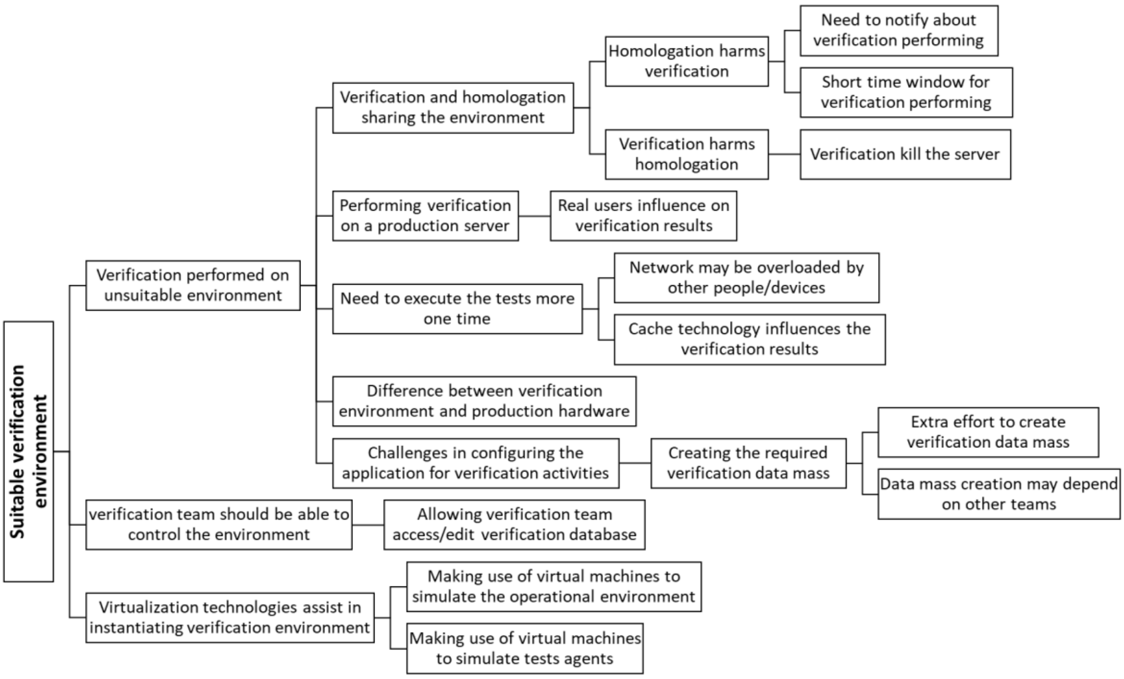

Suitable verification environment
Verification performed on unsuitable environment
Verification and homologation sharing the environment
Homologation harms verification
Need to notify about verification performing
Short time window for verification performing
Verification harms homologation
Verification kill the server
Performing verification on a production server
Real users influence on verification results
Need to execute the tests more one time
Network may be overloaded by other people/devices
Cache technology influences the verification results
Difference between verification environment and production hardware
Challenges in configuring the application for verification activities
Creating the required verification data mass
Extra effort to create verification data mass
Data mass creation may depend on other teams
verification team should be able to control the environment
Allowing verification team access/edit verification database
Virtualization technologies assist in instantiating verification environment
Making use of virtual machines to simulate the operational environment
Making use of virtual machines to simulate tests agents

Figure 54 - MF5: Adequate verification environment moderator factor

It was possible to observe that the *S&P verification occasionally shares the same environment used by other activities*. For example, in one organization, the performance tests were performed on the same server used for user acceptance tests. In this case, there was a bidirectional influence; the performance tests may jeopardize the user acceptance activities because the simulation of a large number of users operating the system causes hardware overload. Furthermore, when the system was used for acceptance testing, the performance tests presented random results (*e.g.*, aleatory response time) because it was not possible to know how the users were using the system. In this case, the organization should appropriately schedule the verification activities (performed by the verification team) and the acceptance tests (performed by end-users) so that these two activities never be performed in parallel.

The *use of a production environment* causes a similar issue to those mentioned above because it is difficult to predict the behavior of the system's real users. Thus, the verification results could be misleading if the system is in actual operation.

The local network (or virtual private network) also influences the performance testing results. For instance, if the machine used for performance tests uses the default organization network, the requests and responses may be delayed due to an overload of the network nodes (*e.g.*, routers and sweets) that route them to the server on which the system is running.

Some *technologies can also influence the results of the verification activities*. For instance, the use of the cache to retrieve data from a database can lead to inaccurate response time test results.

The verification team should resolve network and technology issues by, for example, *performing each test case more than once and at different times* to mitigate external influences on the test results. One of the participants said: "*It is not possible to rely on the response time of only one scenario execution because there may be interference that impairs the operation of the system. Thus, response time analysis should be only performed after the scenario has been successfully executed three times.*"

Another issue regarding the verification environment is the *difference between the hardware configuration used for verification and that used at production time*. In some cases, the hardware used in the production environment is more powerful than the hardware used in verification activities, and this may result in false results regarding system performance.

The *difficulty in configuring the system with suitable data for verification activities* was also an issue that was observed. In this case, some participants stated that to populate the database with suitable data is a difficult task and occasionally requires support from other teams (*e.g.*, database administrators). An alternative to minimizing the dependence on other teams is to allow the verification team to access the database used in the verification activities.

Finally, it was realized that *virtualization technologies* are used for two purposes. First, *to simulate the system execution environment*, trying to obtain an environment more similar to the production environment. Second, *to set up the environment through which tests will trigger*, for example, the creation of several virtual machines to simulate simultaneous access to the system.

5.7.1 Strength of adequate verification environment moderator factor

MF5 was identified in organizations O1, O2, and O4. Moreover, six participants endorsed this moderator factor (Table 29).

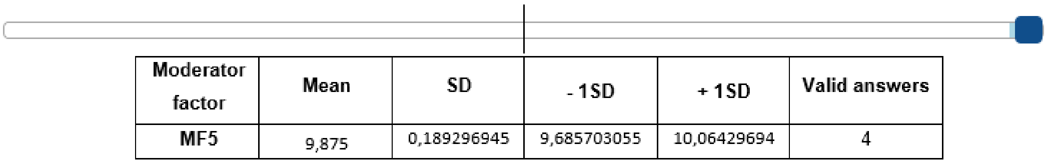
Table 29 - Strength of adequate verification environment moderator factor (MF5)

Organizations	Participants	Quotes
O1	Observation; P1; P2; P3; P5	26
O2	P1	
O4	P1	

There are studies concerned with the verification environment, corroborating the use of virtualization technologies in the context of verification. Netto *et al.* [2011] mention the financial unfeasibility of using physical machines to compose the verification environment, whereas other studies point to virtualization as the most appropriate technology for verification environments. However, some issues should be addressed for the practical use of virtualization technology: the estimation of the number of supported virtual machines, the limit of the number of virtual machines, the instability of test trigger response time and the physical machine overload [Arif et al. 2018] [Kim et al. 2015] [Gaisbauer et al. 2008].

5.7.1.1 Survey results of MF5

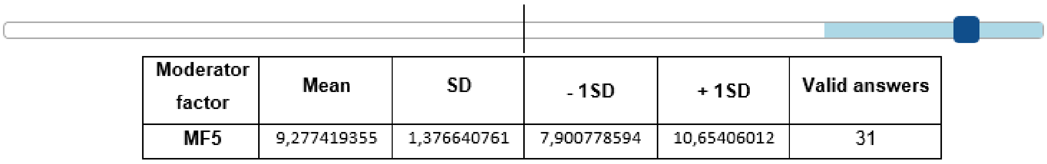
The control group confirmed our findings regarding the relevance of having an appropriate environment to perform security and performance verification activities (Figure 55).



Moderator factor	Mean	SD	- 1SD	+ 1SD	Valid answers
MF5	9,875	0,189296945	9,685703055	10,06429694	4

Figure 55 - MF5 confirmation (control group)

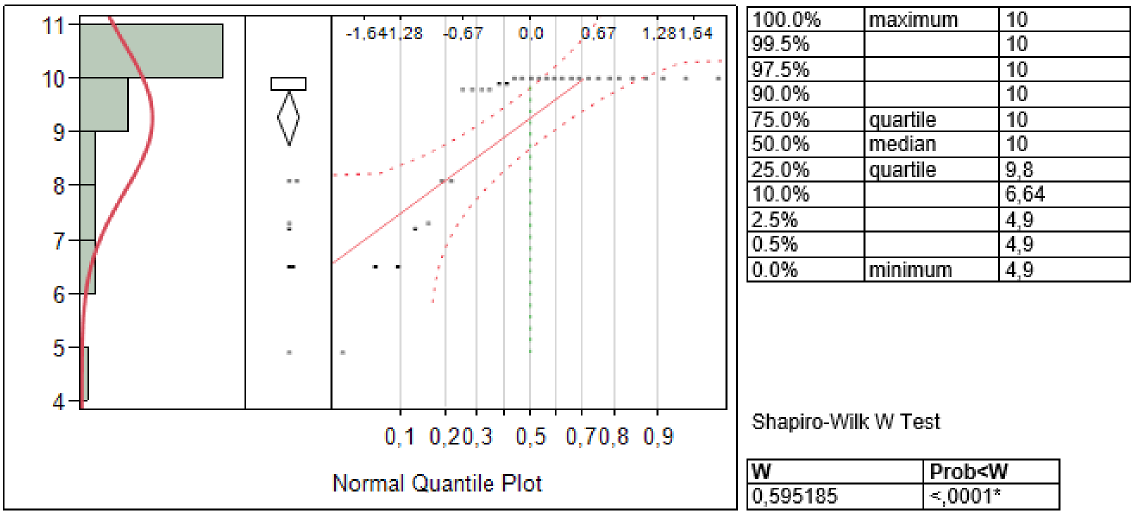
The VAS scale (Figure 56) summarizing the results of the survey shows that the MF5 (adequate verification environment) is a relevant moderator factor according to the opinion of external participants.



Moderator factor	Mean	SD	- 1SD	+ 1SD	Valid answers
MF5	9,277419355	1,376640761	7,900778594	10,65406012	31

Figure 56 - MF5 relevance according to external participants

The statistical analysis of the MF5 responses shows a concentration of the responses at the high values of the VAS scale. Besides, the median of 10 indicates at least 50% of the participants positioned the VAS scale in the highest value. Thus, it can be concluded that most of the external participants understand that an *appropriate environment* is exceptionally relevant to the success of the S&P verification activities.



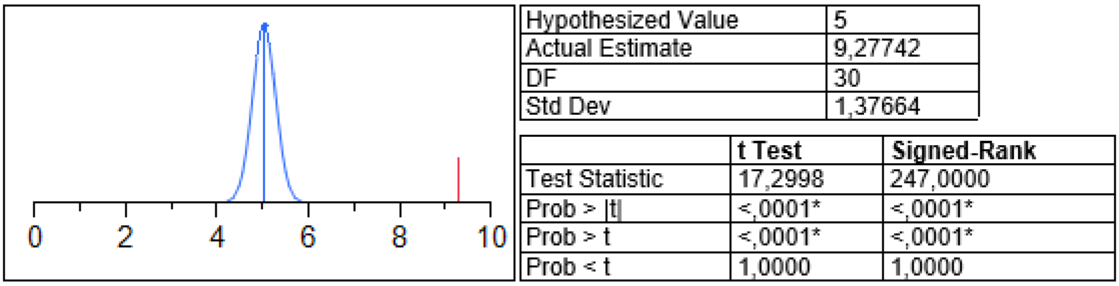
100.0%	maximum	10
99.5%		10
97.5%		10
90.0%		10
75.0%	quartile	10
50.0%	median	10
25.0%	quartile	9,8
10.0%		6,64
2.5%		4,9
0.5%		4,9
0.0%	minimum	4,9

Shapiro-Wilk W Test

W	Prob<W
0,595185	<,0001*

Figure 57 - MF5 distribution

Besides, as shown in Figure 58, the statistical test was unable to show a bias in the sample.



Hypothesized Value	5
Actual Estimate	9,27742
DF	30
Std Dev	1,37664

	t Test	Signed-Rank
Test Statistic	17,2998	247,0000
Prob > |t|	<,0001*	<,0001*
Prob > t	<,0001*	<,0001*
Prob < t	1,0000	1,0000

Figure 58 - MF5 Bias test (mean test)

5.7.2 Actions to promote the configuration of an adequate verification environment

Table 30 presents the actions that can be used to configure an adequate verification environment. Thirty-one survey participants answered about the pertinence of MF5 and the actions used to promote it. Thus, it is possible to order the actions by relevance according to the participants' opinions.

Table 30 - Actions to promote MF5

Actions to promote the *adequate verification environment*	**#**	**%**
Using virtualization technologies to simulate execution environment	26	84%
Keeping the verification team well-informed about used technologies	22	71%
Using virtualization technologies to set up tests agents	19	61%
Performing each test case more than once and at different period of time to mitigate external influences	16	52%
Scheduling the verification activities if it is not possible to instantiate a specific verification environment so that verification should never be performed in parallel with any other activity	15	48%

The participants of the study understand the virtualization technologies as allies of S&P verification. It was also possible to confirm the suitability of such technologies in the technical literature. *Using a virtual environment at verification-time* allows configuring an environment similar to the production environment. Thus, the results of verification become more realistic.

It is also essential to *keep the verification team well-informed about the technologies used during software building* because such technologies can bias the verification results. For instance, the use of cache technologies can result in different response time according to the software state, masking the real performance of the system.

The *virtualization technologies can also be used to simulate testing agents* (*i.e.*, machines from which the tests are trigged). Thus, it is possible to simulate, for example, several users operating the software in parallel, which would be impeditive using real machines.

Different activities use the network infrastructure of the organization (*e.g.*, file transfer, backup routines). These activities can overload the devices (*e.g.*, routers, switches), interfering with the verification results (mostly for performance). Therefore, the *test cases should be performed more than once and at different periods of time to mitigate external influences*.

Finally, If it is not possible to isolate the verification environment using virtualization technologies, the organization should *create a schedule to perform the verification activities*. This action prevents verification from occurring in parallel with other activities (*e.g.*, users performing acceptance testing), avoiding external interferences in the verification results. This is a low relevance action as only less than half (48%) of the survey's participants understand that it can be used to obtain the appropriate verification environment.

5.7.2.1 New actions to promote MF5

Table 31 presents the new actions that emerged through the survey and can be used to *configure an appropriate verification environment*. As this is the first time these actions are emerging, they still need to be investigated in the future.

Table 31 - New actions to promote MF5

New actions to promote the *adequate verification environment*	**#**
Using automated verification	1
Simulating a defined behavior that constitutes real user behavior	1
Using techniques to generate suitable testing data	1

5.8 MF6: Systematic verification methodology

A finding of the case study relates to the existence of a methodology of security and performance verification and recommendations regarding its suitability (Figure 59). When an organization does not follow a suitable methodology, the *verification of S&P is performed in a non-systematic way*, impairing its effectiveness and efficiency. For example, if there is no methodology to guide the verification team, the *test case selection criteria* and *the definition of done are performed informally*, following the tester’s intuition.

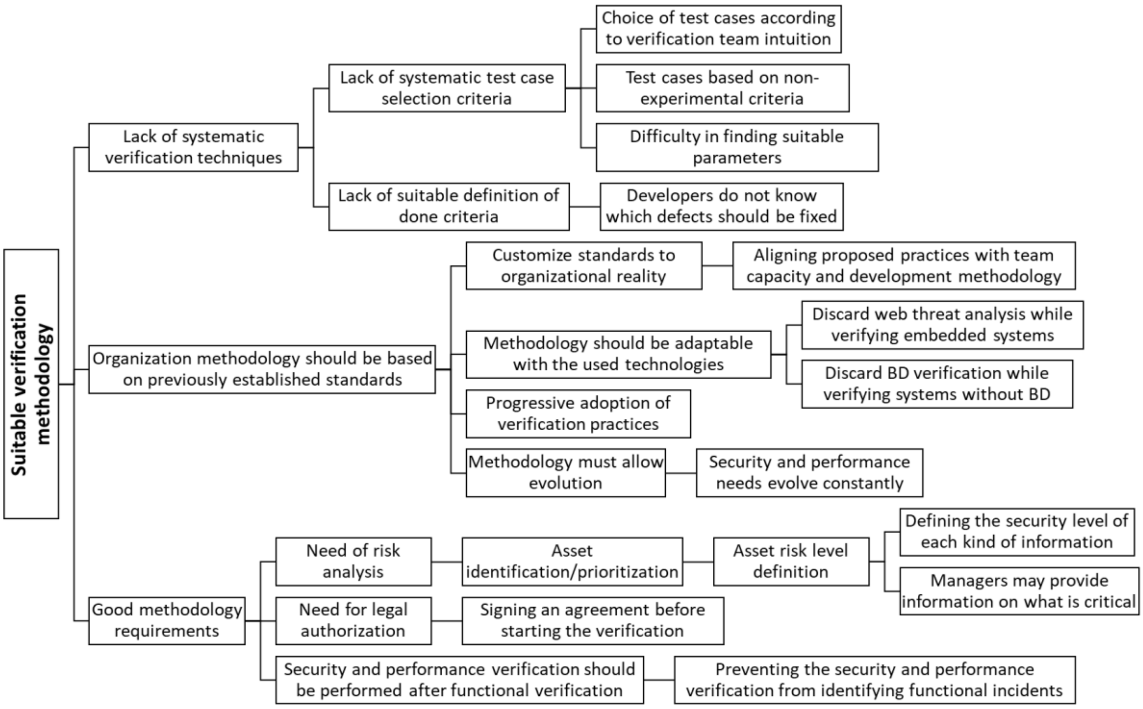

Suitable verification methodology
Lack of systematic verification techniques
Lack of systematic test case selection criteria
Choice of test cases according to verification team intuition
Test cases based on non-experimental criteria
Difficulty in finding suitable parameters
Lack of suitable definition of done criteria
Developers do not know which defects should be fixed
Organization methodology should be based on previously established standards
Customize standards to organizational reality
Aligning proposed practices with team capacity and development methodology
Methodology should be adaptable with the used technologies
Discard web threat analysis while verifying embedded systems
Discard BD verification while verifying systems without BD
Progressive adoption of verification practices
Methodology must allow evolution
Security and performance needs evolve constantly
Good methodology requirements
Need of risk analysis
Asset identification/prioritization
Asset risk level definition
Defining the security level of each kind of information
Managers may provide information on what is critical
Need for legal authorization
Signing an agreement before starting the verification
Security and performance verification should be performed after functional verification
Preventing the security and performance verification from identifying functional incidents

Figure 59 - MF6: Systematic methodology moderator factor

According to the participants of the case study, there are *various publications* (*e.g.*, pre-defined methodologies, norms, and laws) *that can be used as the basis for the definition of a methodology in an organization*. However, it is *not advisable to use these publications verbatim*. It is necessary to understand the recommendations and *adapt them to the context of the organization*, *aligning the proposed practices with the practices already used in the organization and with the team's capability*. A participant said: *"…knowing that our team is small, I have to work according to our ability, performing the tests for which we have the capacity. I took some courses and could apply other verification activities, but I would need an infrastructure that I do not have.*"

A *verification methodology should be adaptable to the technologies used* in the development of a software system. For example, it is useless to perform web vulnerability analysis if the system is embedded, or database verification if the system does not hold any data.

Moreover, *a verification methodology should allow the increasing adoption of the proposed practices*. Thus, the teams can have time to adapt to the new practices. A participant said: "*So, we started using basic open-source tools. Then, we adopted more advanced tools. Thus, using the initial tests, we could understand how we could perform verification and deliver the results to the customers.*"

Moreover, *a methodology should evolve* because system security and performance also evolve with time. Regarding security verification, evolution is mandatory, as new invasion techniques are continuously created.

Appropriate points that the methodology should consider were also identified. The first is a *risk assessment step*, where the *assets should be identified*, and the *criticality level should be assessed*. Furthermore, if the verification activities are performed by a third-party company, the *need for legal authorization* should be considered.

Finally, the verification methodology should make it clear that the *S&P verification activities should be performed after the verification activities targeting the functional requirements*; otherwise, the security and performance verification may identify functional failures, contrary to its real purpose.

5.8.1 Strength of systematic verification methodology moderator factor

Seven participants from three different organizations mentioned the need for a suitable verification methodology (Table 32).

Table 32 - Strength of suitable methodology moderator factor (MF6)

Organizations	Participants	Quotes
O1	Observation; P1; P2; P3; P4	44
O2	P1; P2	
O4	P1	

In the technical literature, Martin and Xie [2007] present the results of an experiment showing the use of a technique increasing the defect detection capability and the coverage of security verification. Furthermore, the use of suitable techniques in different phases of software development (*e.g.*, abuse cases in requirements and modeling, misuse cases and threat trees in design) promotes the identification of defects in early stages of software development [McDermott and Fox 1999] [Alexander 2003] [Woodraska et al. 2011] [Omotunde and Ibrahim 2015]. Some studies also suggest a combination of techniques to increase the ability to detect different types of defects, *e.g.*, complementing the automated tests with manual reviews [Omotunde and Ibrahim 2015] [Ghindici et al. 2006] [Brucker and Sodan 2014]. Thus, a methodology guides the verification team to choose the suitable technique to assess each of the software assets.

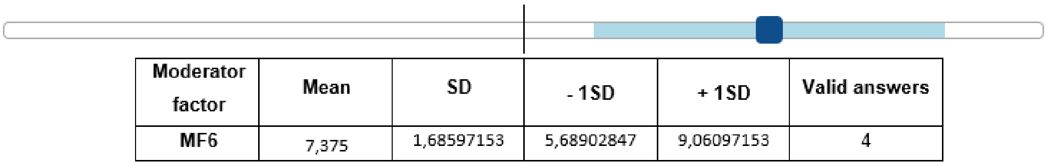
It was confirmed that organizations should not develop an entirely new methodology from scratch. It is more suitable to adapt to an existing methodology [Study 2014] [Choliz et al. 2015].

Some studies discuss the inadequacy of existing methodologies in an agile development process [Ge et al. 2006] [Erdogan et al. 2010] [Sonia and Singhal 2012] [Ayalew et al. 2013] [Wäyrynen et al. 2004] [Siponen et al. 2005] and how such methodologies can be adapted to be more agile. In an agile development process, the lack of documentation [Kongsli 2006] and constant refactoring [Beznosov and Kruchten 2005] can be impeditive characteristics in implementing current methodologies. Furthermore, the demanding activities proposed by real S&P methodologies can hinder development process agility [Keramati and Mirian-Hosseinabadi 2008]. In this sense, there is a proposed metric that can be used to measure the agility of a verification methodology. It can be used to evaluate if the adoption of a methodology will impact the agility of the development process [Keramati and Mirian-Hosseinabadi 2008].

Finally, it was confirmed that considering asset identification and risk analysis is an essential requirement of a sound methodology [Study 2014] [De Win et al. 2009].

5.8.1.1 *Survey results of MF6*

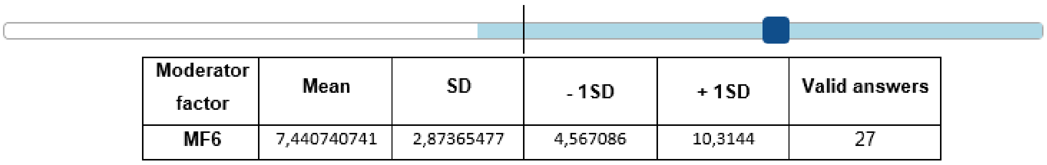
Figure 60 presents the summary of the control group opinions regarding MF6 on the VAS scale. The position of the mean (dark blue square) in the visual scale allows us to conclude that our understanding of the relevance of a *systematic verification methodology* is correct.



Moderator factor	Mean	SD	- 1SD	+ 1SD	Valid answers
MF6	7,375	1,68597153	5,68902847	9,06097153	4

Figure 60 - MF6 confirmation (control group)

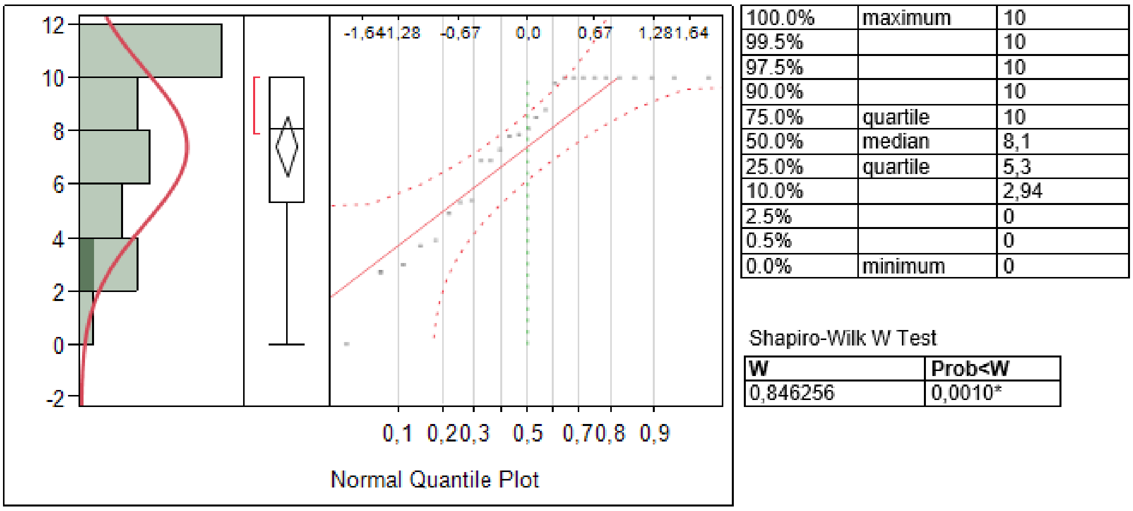
The external participants of the survey also agree regarding the pertinence of MF6 (Figure 61). However, the significant standard deviation indicates that the systematic methodology does not have high relevance in some contexts.



Moderator factor	Mean	SD	- 1SD	+ 1SD	Valid answers
MF6	7,440740741	2,87365477	4,567086	10,3144	27

Figure 61 - MF6 relevance according to external participants

According to the Shapiro-Wilk test, the answers do not follow a normal distribution, and the diagram (Figure 62) shows that the high level of the scale includes most of the answers.



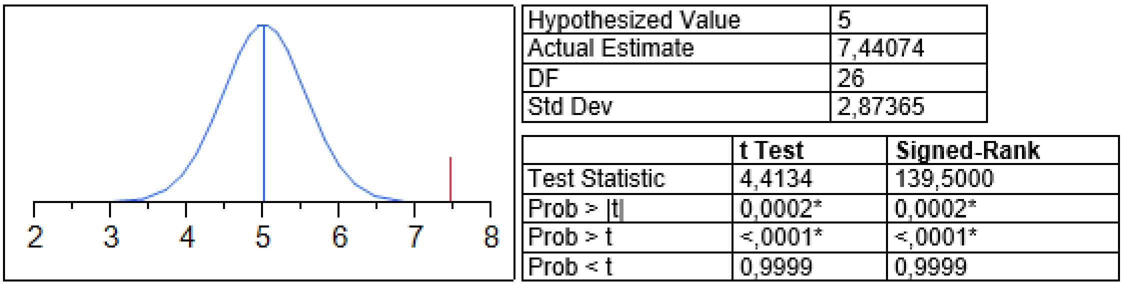
100.0%	maximum	10
99.5%		10
97.5%		10
90.0%		10
75.0%	quartile	10
50.0%	median	8,1
25.0%	quartile	5,3
10.0%		2,94
2.5%		0
0.5%		0
0.0%	minimum	0

Shapiro-Wilk W Test

W	Prob<W
0,846256	0,0010*

Figure 62 - MF6 distribution

The statistical test aiming to identify bias in the response shows it is not biased (Figure 63).



Hypothesized Value	5
Actual Estimate	7,44074
DF	26
Std Dev	2,87365

	t Test	Signed-Rank
Test Statistic	4,4134	139,5000
Prob > |t|	0,0002*	0,0002*
Prob > t	<,0001*	<,0001*
Prob < t	0,9999	0,9999

Figure 63 - MF6 Bias test (mean test)

5.8.2 *Actions to promote the use of a systematic verification methodology*

We could find a single action to promote the use of a systematic methodology (Table 33). This action consists of choosing a methodology already proposed (*e.g.*, OWASP, Microsoft SDL) and adapting it to the context and particular needs of the organization. The survey result shows that 21 out of 27 participants agree that this action can be used to promote MF6.

Table 33 - Actions to promote MF6

Actions to promote the *systematic verification methodology*	#	%
Using a proposed methodology and adapting it to the context of the organization	21	78%

5.8.2.1 *New actions to promote MF6*

As shown in Table 34, the survey allowed the identification of 4 new actions to promote the *use of a systematic verification methodology*.

Table 34 - New actions to promote MF6

New actions to promote the *systematic verification methodology*	#
Modify the company culture at some level by fostering a new methodology	1
Search for a methodology aligned with stakeholders needs	1
Use appropriately trained testers; avoid using a dopey methodology	1
Create processes and revise them according to proposed methodology and company context	1

5.9 MF7: Plan security and performance verification activities

Another issue regarding security and performance verification relates to the planning activity (Figure 64). Usually, verification is not well planned, leading to the need to reprioritize the verification activities, and consequently to the reduction of their coverage.

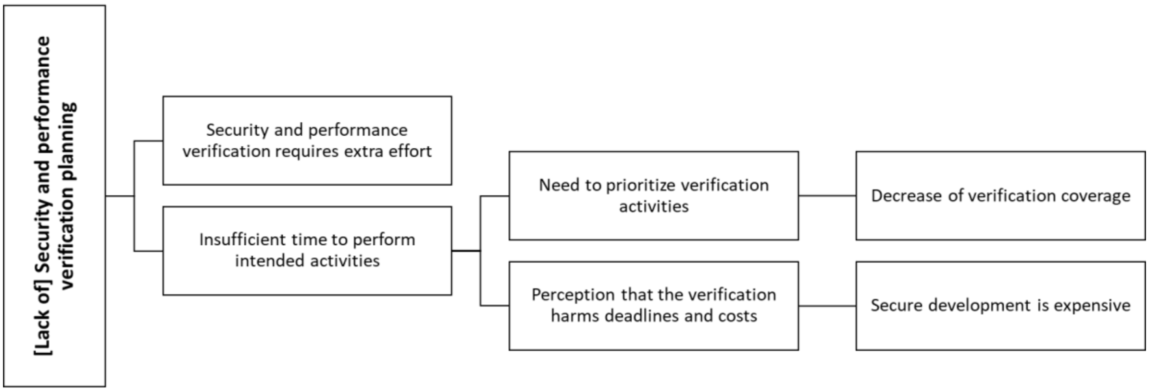

[Lack of] Security and performance verification planning
Security and performance verification requires extra effort
Insufficient time to perform intended activities
Need to prioritize verification activities
Perception that the verification harms deadlines and costs
Decrease of verification coverage
Secure development is expensive

Figure 64 - MF7: Planning security and performance verification moderator factor

The study participants have the perception that *S&P verification activities require additional effort and cost*. The managers neglect these activities, excluding them from verification planning. A participant presented his opinion about why security and performance verification activities are not planned (or were included in the project planning stage): "*… '- How much does it cost to develop a software system?'. '- It costs 300 thousand'. '- And with security?'. '- Well, it depends. So, I should evaluate it. There is a need to have a team performing the verification, and this will have a cost and time impact'. '- So, then leave it for later, for a second version.'…*"

Additionally, while a team of Org1 was performing response time tests, the release time was changed, and some test cases could not be executed. Thus, the team expended more effort reprioritizing the test cases (the activity of planning phase) than executing them.

Moreover, the participants reported that the stakeholders (*e.g.*, managers and customers) have the perception that verification activities can change the delivery time or the cost of a system. However, they do not consider the benefits of these activities. A participant said: "*…every time I talk to someone about testing, about security, or things like that, people always think that it will change the delivery deadline: 'Wow, I need to do it fast.' 'Folks, you are not going to get rework if you do it well.'…*"

5.9.1 Strength of plan security and performance verification activities moderator factor

Table 35 presents the strength of this moderator factor. It is mentioned only six times by two participants.

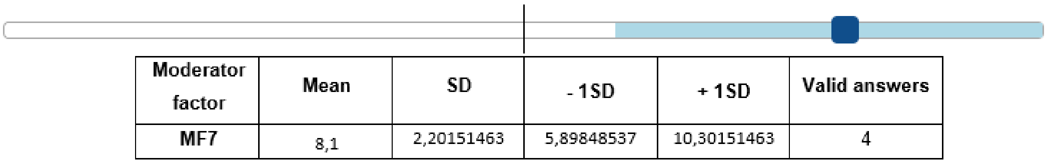
Table 35 - Strength of plan security and performance verification activities moderator factor (MF7)

Organizations	Participants	Quotes
O1	Observation	6
O2	P1	
O4	P1	

In technical literature, it can be identified that successful planning can reduce the number of redundant test cases without losing efficiency [Omotunde et al. 2018]. Furthermore, in a study aiming to identify the skills of good testers, planning ability was recognized, although it was not the most important [Iivonen et al. 2010]. Finally, according to Bozic and Wotawa [2015], the planning phase can be guided by support tools, decreasing the testers' effort.

5.9.1.1 Survey results of MF7

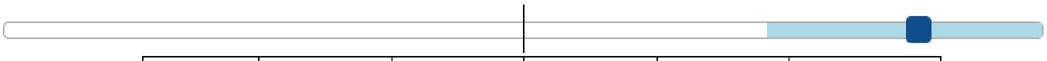
Figure 65 presents the results summarizing the opinion of the control group plotted on the VAS scale. The position of the mean (dark blue square) allows us to conclude that the participants of the control group agree with the pertinence of MF7, revealing that we correctly interpret the information that pointed to this moderator factor.



Moderator factor	Mean	SD	- 1SD	+ 1SD	Valid answers
MF7	8,1	2,20151463	5,89848537	10,30151463	4

Figure 65 - MF7 confirmation (control group)

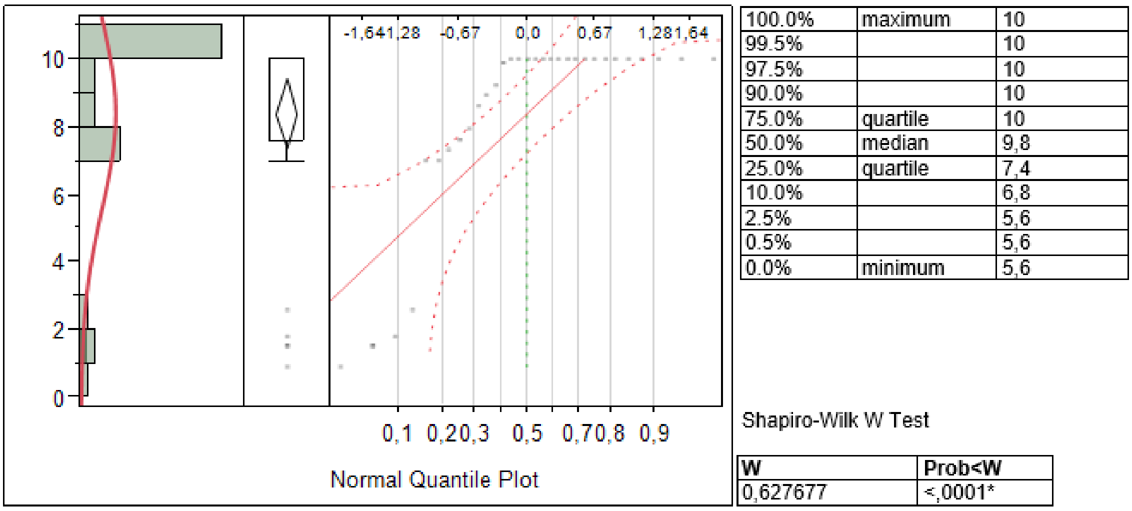
The 29 valid answers show the external participants understand the planning of S&P verification as an essential moderator factor (Figure 66).



Moderator factor	Mean	SD	- 1SD	+ 1SD	Valid answers
MF7	8,817241379	1,461180776	7,356061	10,27842	29

Figure 66 - MF7 relevance according to external participants

Figure 67 presents the graphic and some information about the statistical analysis performed on the answers related to MF7. The graphic shows the answers are majority concentered in the high levels of the VAS scale. Besides, the median of 9.8 indicates that at least 50% of the participants positioned the VAS scale extremely close to its maximum value, manifesting a high agreement regarding the pertinence of the moderator factor.



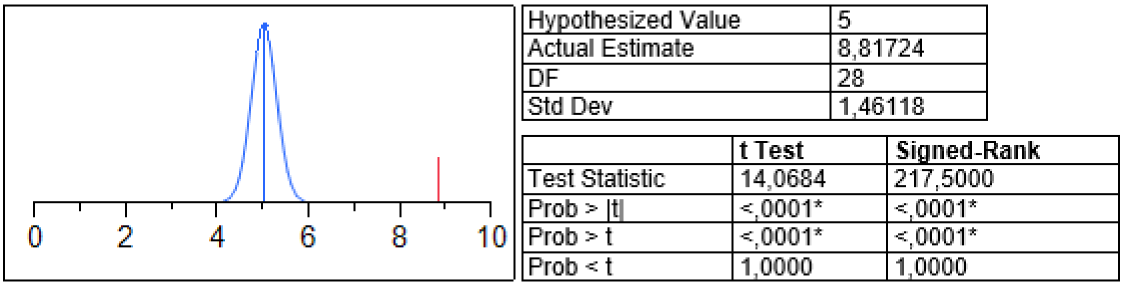
100.0%	maximum	10
99.5%		10
97.5%		10
90.0%		10
75.0%	quartile	10
50.0%	median	9,8
25.0%	quartile	7,4
10.0%		6,8
2.5%		5,6
0.5%		5,6
0.0%	minimum	5,6

Shapiro-Wilk W Test

W	Prob<W
0,627677	<,0001*

Figure 67 - MF7 distribution

As shown in Figure 68, the mean test with the hypothesized value of five does not identify a bias.



Hypothesized Value	5
Actual Estimate	8,81724
DF	28
Std Dev	1,46118

	t Test	Signed-Rank
Test Statistic	14,0684	217,5000
Prob > |t|	<,0001*	<,0001*
Prob > t	<,0001*	<,0001*
Prob < t	1,0000	1,0000

Figure 68 - MF7 Bias test (mean test)

5.9.2 Actions to promote the planning of security and performance verification

Table 36 presents the only practice we can find to foster the creation of a plan for the S&P verification activities. Such action is the *use of tools to guide the planning*, decreasing the effort, and improving the formality of planning activities. 25 out of 29 survey participants agree that this action is relevant to promote the planning of security and performance verification.

Table 36 - Actions to promote MF7

Actions to promote the *planning of security and performance verification*	**#**	**%**
Using a tool to guide the security and performance verification planning	25	86%

5.9.2.1 New actions to promote MF7

Additionally, it was possible to identify two new actions to promote the planning of security and performance verification. Further investigations are required for understanding the relevance of these new practices and the context they could be applied.

Table 37 - New actions to promote MF7

New actions to promote the *planning of security and performance verification*	**#**
Including the security and performance verification activities as part of the development and maintenance cycle	1
Having business knowledge helps prioritize the parts of the system that should be evaluated	1

5.10 MF8: Reuse practices

The *reuse of knowledge and artifacts* was also identified as a recommendation, bringing more agility to security and performance verification activities (Figure 69). The *functional test cases of the system may be used in performance tests* because they represent real usage scenarios. Moreover, it is possible to *adapt the parameters of the test cases of previous systems*, reducing the construction effort and time.

It was possible to observe that *previous similar systems might be used as a basis for the definition of the requirements*. For instance, the required response time of a scenario can be defined based on a similar scenario of a production system. A participant said: "*The number of concurrent users the system should support is defined by a similar system that is already in production.*"

Finally, it is important to *know common defects* (*e.g.*, common vulnerabilities and exposures) and to use pre-defined test cases to identify the failures caused by these faults. While talking about penetration tests, a participant mentioned the use of well know cross-site scripting strings (test cases): "*I have a database with more than 350 XSS queries… it is populated with my own knowledge and aggregating other internet databases… OWASP has a database that we can download. Usually, they are the most frequent attacks... I also keep an eye on Exploitdb and vulnerability monitoring platforms. Usually, when they publish an exploit, they also present the XSS query together. So, these XSS queries are well known, and it is possible to make them more generic to use in other systems*".

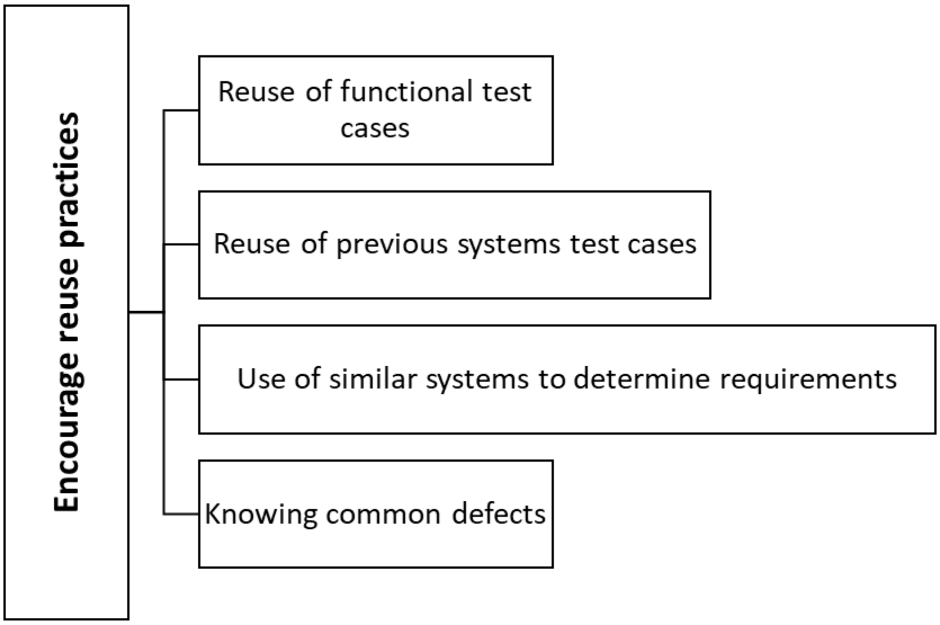

Encourage reuse practices
Reuse of functional test cases
Reuse of previous systems test cases
Use of similar systems to determine requirements
Knowing common defects

Figure 69 - MF8: Reuse practices moderator factor

5.10.1 Strength of reuse practices moderator factor

As shown in Table 38, it was possible to identify MF8 in two organizations. In the first, it was observed and mentioned by two participants. In the second, it was mentioned by one of the participants.

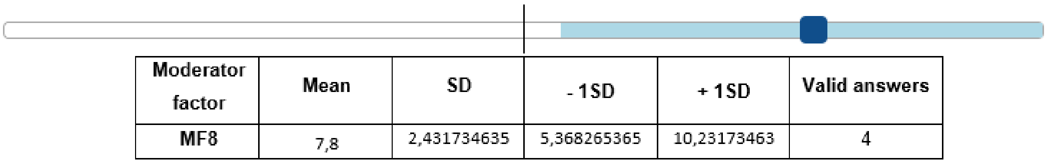
Table 38 - Strength of reuse practices moderator factor (MF8)

Organizations	Participants	Quotes
O1	Observation; P2; P3	6
O2	P2	

There are studies that present the benefits of reusing functional testing as non-functional testing. The functional test cases can be reused both as security and performance test cases, bringing benefits such as an increase in the coverage, improvement of failure detection rate and cost reduction to perform tests and generate suitable testing data set [Dazhi Zhang et al. 2010] [Santos et al. 2011]. Furthermore, Santos et al. [2011] claim that the reuse of functional testing can bring indirect benefits: an increase in the quality of functional testing because the effort saved in performance testing can be used to improve functional testing; an increase in the diffusion of functional testing due to the increased importance of them in the development process.

5.10.1.1 Survey results of MF8

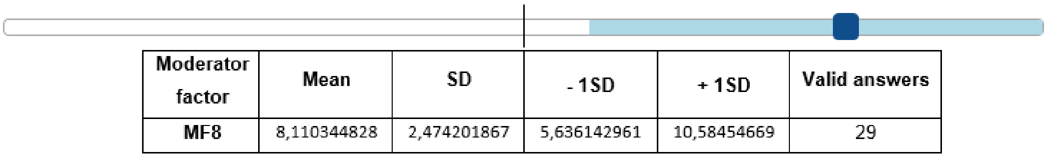
Figure 70 summarizes the opinions of the control group regarding *reuse practices*. As the participants agreed with the moderator factor, we can conclude that we were able to correctly interpret the information that led to the creation of this moderator.



Moderator factor	Mean	SD	- 1SD	+ 1SD	Valid answers
MF8	7,8	2,431734635	5,368265365	10,23173463	4

Figure 70 - MF8 confirmation (control group)

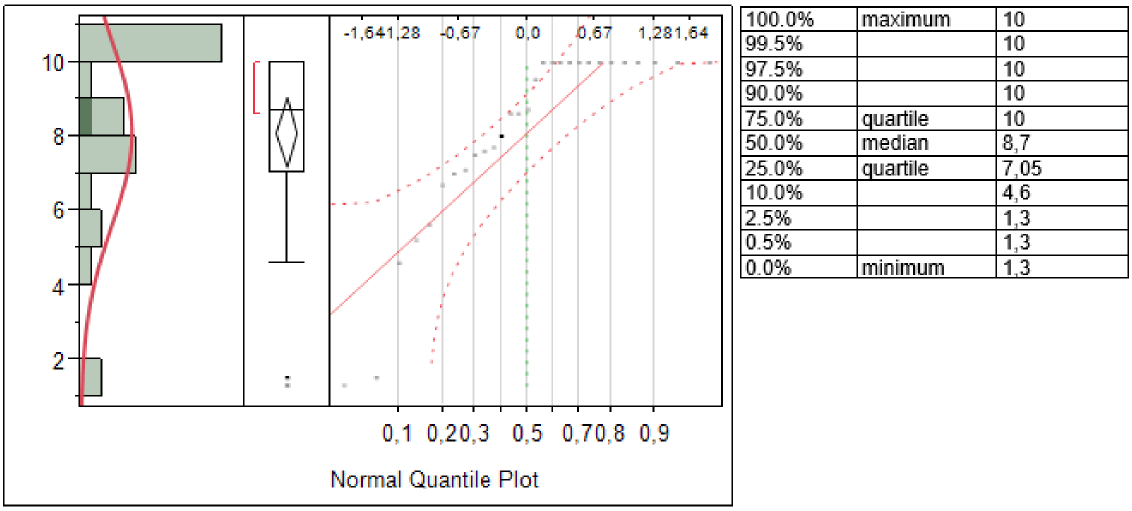
The outside group, composed of 29 participants, understand *reuse practices* as relevant to the S&P verification activities (Figure 71).



Moderator factor	Mean	SD	- 1SD	+ 1SD	Valid answers
MF8	8,110344828	2,474201867	5,636142961	10,58454669	29

Figure 71 - MF8 relevance according to external participants

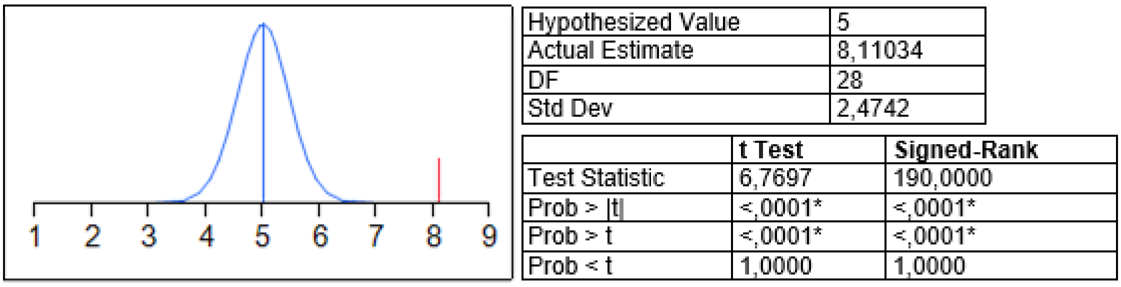
The statistical analysis revealed a distribution similar to the others - most of the participants positioned the VAS scale in a position indicating the moderator factor has high relevance (Figure 72).



100.0%	maximum	10
99.5%		10
97.5%		10
90.0%		10
75.0%	quartile	10
50.0%	median	8,7
25.0%	quartile	7,05
10.0%		4,6
2.5%		1,3
0.5%		1,3
0.0%	minimum	1,3

Figure 72 - MF8 distribution

Finally, the mean test also shows there is no bias (Figure 73).



Hypothesized Value	5
Actual Estimate	8,11034
DF	28
Std Dev	2,4742

	t Test	Signed-Rank
Test Statistic	6,7697	190,0000
Prob > |t|	<,0001*	<,0001*
Prob > t	<,0001*	<,0001*
Prob < t	1,0000	1,0000

Figure 73 - MF8 Bias test (mean test)

5.10.2 Actions to promote reuse of practices

A software development organization can make use of some actions to encourage the reuse of practices (Table 39). The opinions of 29 survey participants were used to order the actions by relevance.

Table 39 - Actions to promote MF8

Actions to promote the *reuse of practices*	**#**	**%**
Knowing common defects (*e.g.*, vulnerabilities) and using pre-defined test cases to identify the failures caused by these defects	25	86%
Reusing the knowledge acquired from other similar systems as a basis for the definition of the requirements	23	79%
Reusing functional test cases as they represent real usage scenarios	19	66%
Reusing test cases from similar systems adapting parameters	13	45%

Different services analyze and disclose common vulnerabilities and exposures (CVE). CVEs are defects related to security. In disclosing these defects, these services also provide instructions on how to detect these defects (test cases). The verification team should regularly consult these services *to be aware of common defects and reuse the available test cases*.

Besides, the verification team can *use similar systems as a basis to define S&P requirements*. As mentioned before, the S&P requirements may not exist. Thus, using the experience with similar systems already in production can be considered because it allows gaining insight into the needs of the new system based on the behavior of actual real users of a similar system.

It is possible to *reuse functional test cases as they represent real usage scenarios*. For instance, it is more appropriate to assess the response time of a software feature using real data as input than using randomly generated data.

Finally, it is also possible to reuse the structure of test cases of similar systems, avoiding building them from scratch. However, this action is not relevant according to the opinion of most survey participants (55%) as only 45% state it is relevant. Thus, it can be considered a low relevance action.

5.10.2.1 New actions to promote MF8

Table 40 presents the new five new actions we could identify through the survey. As the new actions identified to promote the other moderator factors, these actions should be further investigated to increase their understanding and their relevance in different contexts.

Table 40 - New actions to promote MF8

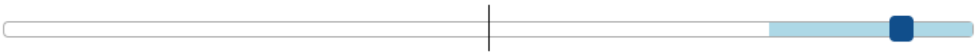
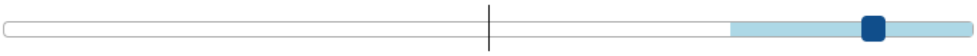
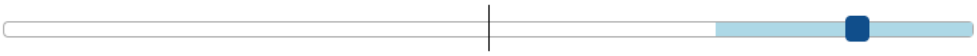
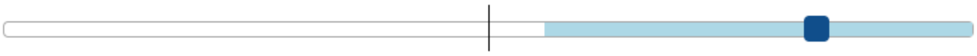
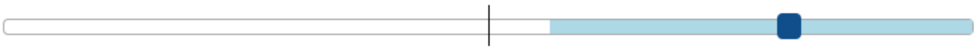
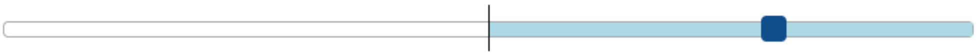
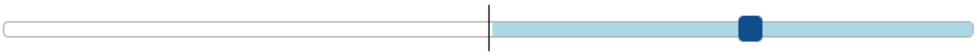
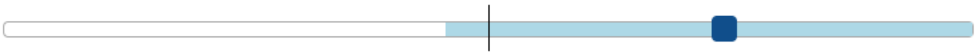
New actions to promote the *reuse of practices*	**#**
Creating a base of knowledge of recurring defects	1
Mapping vulnerability according to the domain to promote the identification of vulnerabilities applicable to specific situations	1
Functional test cases specify what could be added for performance verification	1
Design real-time scenario with production volume data, per hour, per day transaction, Per week, etc	1
Reusing multiple test scenarios is very useful for both professional and runtime scenarios that we can insert in the context of similar new projects	1

5.11 Conclusions

The eight moderator factors presented in this chapter represent topics a software development organization should consider in order to perform the S&P verification activities successfully. These moderator factors arose from a case study, and they were confirmed thought the technical literature and a survey.

As shown in Table 41, the moderator factors can be ordered by their relevance according to the practitioners' opinions (survey).

Table 41 - Moderator factors ordered by relevance

MF5	
MF1	
MF7	
MF3	
MF8	
MF2	
MF4	
MF6	

MF1: Organizational awareness of the importance of security and performance

MF2: Cross-functional team

MF3: Clear requirements

MF4: Suitable support tools

MF5: Adequate verification environment

MF6: Systematic verification methodology

MF7: Plan security and performance verification activities

MF8: Reuse practices

Therefore, through the survey results, it is possible to conclude that every presented moderator factor is relevant to the S&P activities.

Besides, the chapter presents the actions that can be used to promote each of the moderator factors and their relevance according to practitioners' opinions. The relevance level of each of the actions can be used as a selection criterion if it is not possible to implement all actions.

Therefore, this chapter provides a kind of guideline presenting essential topics to the success of S&P verification and how to achieve these topics (actions).

6 Conclusion

This chapter presents the final thesis considerations, highlighting the main contributions as we describe the answers to the research questions. Besides, it outlines the research limitations represented by the threats to validity and the actions we took to mitigate them. Finally, it presents the open questions indicating future works that can arise from the current findings.

6.1 Final considerations

In general, this work can be classified as descriptive research as it presents a characterization of a phenomenon (non-functional verification). This kind of research results in an organized description of a phenomenon that can be used to foment future researches and provide information to support decision-making in practice [Jackson 2012]. Therefore, as descriptive research, the thesis describes the “What” regarding the verification of NFR without emphasizing the “Why.”

The thesis used the results of two structured literature reviews to provide an organized body of knowledge of non-functional requirements and the testing approaches that can be used to assess them (NFR-BoK). NFR-BoK is ordered by relevance according to the number of papers citing them. Thus, software development organizations can use these findings to prioritize the NFRs and to select the software testing approaches that could be used to assess these NFRs.

These initial findings showed that there are some difficulties in testing NFRs because (1) there is no consensus regarding the software properties each NFR represents, (2) there are NFRs that are not covered by a testing approach, (3) the non-functional testing approaches do not cover all testing dimensions, and (4) some testing approaches are not evaluated experimentally. Therefore, it indicates that non-functional testing is a topic that still needs researches to evolve.

After organizing NFR-BoK, we realized it was essential to focus our efforts on understanding the most relevant NFR (security and performance) and open the research scope to software verification (broader than software testing). Thus, following the recommendations of our qualifying committee, we started understanding how S&P verification has been performed in practice.

By using a case study as a research method, this thesis provides a characterization of the S&P verification practices performed by software development organizations. The characterization describes the S&P verification practices regarding the used techniques, definition of *done* criteria, automation level, and the assets the practice covers. Besides, it provides a set of factors used to make decisions regarding S&P verification: the selection of tools, verification practice, coverage criteria, and definition of done.

The set of identified S&P verification practices and decision-making factors did not intend to be exhaustive but indicates they work on practical contexts as the organizations are using them. Besides, it can be used as input to future research.

In addition, we realized that security and performance are not adequately treated by many organizations, as when we were recruiting organizations to participate in the case study, several of them said they did not perform verification of security and performance.

Finally, this thesis provides a set of eight moderator factors influencing the S&P verification activities. The moderator factors represent points a software development organization should be concerned in order to perform the S&P verification activities successfully. Additionally, it also provides a set of actions that software organizations development can use to promote each moderator factor. These findings showed that software development organizations should (1) promote the awareness of the importance of the S&P, (2) keep a cross-functional verification team, (3) produce precise S&P requirements, (4) make use of suitable S&P verification tools, (5) configure an adequate S&P verification environment, (6) use an S&P verification methodology, (7) plan the S&P verification activities, and (8) encourage reuse practices.

The moderator factors and their actions were evaluated using the technical literature and practitioner's opinion using a systematic research methodology. Therefore, we are confident about the use of these findings as practical guidance to introduce or improve the S&P verification.

Therefore, the main outputs of this thesis are (1) the book of knowledge of non-functional requirements and their software testing techniques, (2) the characterization of the S&P verification practices and their decision-making factors, and (3) the set of eight moderator factors influencing the S&P verification and the actions used to promote them.

6.2 Research contributions revisited

The overall objective of this thesis was to characterize the verification of software non-functional requirements. Organizing the knowledge involving this topic allows a better understanding of the area and the identification of appropriate practices (recommendations) as well as the weaknesses (opportunities for evolution).

Thus, we describe the contributions of this research as we present a summary of the answers to the research questions:

- **RQ0.1:** What are the most relevant non-functional requirements, according to software practitioners?
- **RQ0.2:** What are the software testing approaches used to test non-functional requirements?
- **RQ0.3.1:** What are the relevant NFRs that are not covered by testing approaches?
- **RQ0.3.2:** What are the test dimensions met by the test approaches?

We used two structured literature reviews to answer these initial questions. The first literature review looked for papers describing the opinion of practitioners regarding the relevance of NFRs to a software system. Thus, it allowed the identification of the most relevant NFRs and a set of characteristics describing them. Therefore, it was possible to answer RQ0.1 ordering the NFRs by their relevance according to the number of practitioners who cited them. Besides, the description of the NFRs characteristics provides a deep understanding of their meaning. It is essential due to the lack of consensus on the software property each NFR represents.

The second literature review allowed the identification of software testing approaches supporting the assessment of NFRs. Thus, we were able to create a catalog of non-functional testing approaches and their target NFRs, answering RQ0.2.

After, by combining the findings of the literature reviews, it was possible to identify relevant NFRs that do not have a software testing approach supporting their assessment. This result answered RQ0.3.1 and indicated research opportunities.

Besides, we provide information regarding the testing dimensions (phases, levels, and type of technique) each identified testing approach can evaluate (RQ0.3.2). This information is vital because specific testing strategies can only identify some categories of defects.

Finally, these findings were organized into a body of knowledge (Chapter 3) that can be accessed and evolved by the software engineering researchers' community. The compilation of these findings allowed us to realize the need to focus on the most relevant NFRs: security and performance.

- **RQ1:** Which are the *practices* used by organizations to support the verification of security and performance?

  We performed a case study aiming to identify S&P verification practices used by software development organizations. Thus, we identified six S&P verification practices and their characterization regarding used techniques, the definition of done, the automation level, and assets covered (Section 4.3).

  As we used a case study as a research method, it was not feasible to obtain information from a large number of organizations. Thus, the results cannot be used to affirm that these are the standard S&P verification practices used by software organizations. However, these results can be used by future researches to understand, for example, why these practices have been used.
- **RQ2:** How does the organization define its security and performance verification strategies?

  The case study was also used to identify decision-making criteria regarding S&P verification. Thus, we were able to identify factors influencing the choice of S&P verification tools, S&P verification practices, and S&P verification coverage criteria. Besides, it was possible to identify who is responsible for the decision-making regarding the choice of S&P verification tools, S&P verification practices, and definition of *done* of S&P verification (Section 4.4).

  These findings cannot be generalized due to the intrinsic limitations of the case study method (a small number of organizations). However, these results can be used as a starting point for further researches aimed to confirm and identify new factors influencing the decision-making related to S&P verification activities.

- **RQ3:** What are the moderator factors influencing security and performance verification?

This thesis provides a set of eight moderator factors influencing the S&P verification. It also includes actions that can be used to promote each of the moderator factors (Chapter 5).

The moderator factors arose from the data collected during the case study, and then they were improved and confirmed though a set of rapid reviews. In sequence, the relevance of each moderator factor was identified according to practitioners' opinions (survey). Therefore, if it is not possible to address all moderating factors, software development organizations may use relevance as the selection criterion.

6.2.1 *Contributions to the software industry*

The findings presented in this thesis can contribute to the software development industry in different ways. First, the NFR-BoK can support organizations to define and understand the NFRs their software products should meet. Besides, the information provided by the NFR-BoK support organizations to select suitable testing approaches to evaluate NFRs. Additionally, the NFR-BoK increases the awareness of organizations regarding the lack of testing approaches to evaluate some NFRs. Thus, it is possible to estimate the risk of do not verify such NFRs.

This thesis identified a set of security and performance verification practices used by the software industry. Besides, the presented decision-making factors support the organizations to choose the criteria to make decisions regarding S&P verification.

Finally, the moderator factors indicate topics that software development organizations should invest to improve S&P verification activities. Such moderator factors can be addressed by a set of actions also provided in this thesis.

6.2.2 *Contributions to academia*

This thesis contributes to the software engineering research area as it shows how different research methods can be combined into a research project. It Demonstrate how to use structured literature reviews to build a trustworthy body of knowledge, how to use the coding phase of grounded theory to analyze the results of a literature review, present relevant insights on how to use rapid reviews to increase the confidence of case study findings, present example of use of thematic analysis to analyze the data of a case study research, and present an approach on how to use a survey to bring knowledge from industry to academia, validating theoretical results. Therefore, other researches can make use of these mythological steps in their researchers.

Besides, the organization of the NFR-BoK provides initial information regarding the NFRs, which can support further investigations of such NFRs.

Additionally, it evidenced different research opportunities. For example, the opportunity to create new testing approaches to evaluate the NFRs and improve the coverage of the existing testing approaches.

6.3 Threats to validity

The threats to validity are presented according to the research cycle as they are influenced by the research methods performed in each of the cycles.

6.3.1 Threats to validity of cycle 1

Cycle one used structured literature reviews as the research method. Besides, the collected data were analyzed through a qualitative approach (open coding). Thus, the threats to the validity of this cycle are related to subjective evaluations carried out on this phase. For instance, the *open coding* is an interpretative process, and it could have led us to a wrong categorization of non-functional requirements. Moreover, the NFR-BoK is based on NFR descriptions provided by LR1. Thus, further investigation can direct effort on understanding a specific NFR, resulting in the restructuring of the body of knowledge, and it could result in changes in our initial findings.

Furthermore, papers included in LR2 are not clear about test dimensions. Therefore, defining a particular approach coverage (testing phases, levels, and techniques) was an interpretative task. Thus, despite having followed a systematic methodology, the process is failure-prone.

Another threat to validity is related to the use of a single search engine. The strategy of only using Scopus was adopted because, through previous literature reviews, we realized that other search engines have a low contribution to the research coverage.

6.3.2 Threats to validity of cycle 2

As the case study was the core research method of cycle two, we describe here how we tried to mitigate the threats to validity following the recommendations of Cruzes and Othmane [2018] and using the quality criteria (Q1-4) and proposed methods (M1-6) to handle them [Lincoln and Guba 2016] [Maxwell 2012].

Credibility (Q1), representing the quality of being convincing or believable, was addressed using rich data/persistent observations (M1) and through data collection using three methods (observation, interviews, and questionnaires), by making notes about what happened, and by verbatim transcripts of what participants said. Furthermore, quotes from the participants were provided.

The *transferability* (Q2) quality refers to the degree to which the results can be generalized to other contexts or settings. This quality is problematic in the case of studies because it is not possible to have a significant number of subjects, as was in the present case. However, to improve transferability, we used the intensive long-term involvement (M2) method, whereby the research was conducted on-site, making it possible to have a more accurate contextual perception. Thus, it was possible to provide an in-depth description of the organizations' characteristics and the context in which data were collected.

Regarding *dependability* (Q3), data stability and reliability over time and various conditions, the study was conducted in four different organizations with participants of a variety of profiles. Thereby, the results can be triangulated (M3), improving dependability. Additionally, the research protocol is available, making it possible to replicate the study in different contexts.

To avoid researcher bias and improve *confirmability* (Q4), peer debriefing (M4) was used, exposing the main findings to a research group and discussing their coherence. Furthermore, multiple meetings among the thesis author and other researchers were held to discuss the codes. Additionally, a search in the literature was performed to support the conjectures. A survey was performed as an instance of the methods of *respondent validation* (M5) and *member checking* (M6).

Furthermore, the case study was conducted among Brazilian organizations, where Portuguese is the native language. Thus, the participants' quotes reproduced here are translations of what they said. Moreover, the artifacts and codes were initially in Portuguese and translated into English to be presented here. This translation does not affect the results reported, as no sentiment/feelings analysis was performed on the answers.

Finally, investigating two non-functional requirements together can be risky. There were situations in which it was not possible to determine whether a respondent was reporting issues related to security or performance.

6.4 Future work

As a work describing a software engineering topic in its broader scope, there are several opportunities for further investigations of each individual part from a deeper perspective. Such opportunities are described below as future work.

Improvements to NFR-BoK regarding the NFRs characterization – we organized NFR-BoK based on the results of a literature review aimed to identify relevant NFRs. Thus, we were not focused on the understanding of a particular NFR. Therefore, further researches focusing on a particular NFR can identify additional characteristics, resulting in the improvement of NFR-BoK. Besides, the relevance of the NFRs can change and shall be reflected in NFR-BoK.

Work on spreading usage of NFR-BoK – we provided a body of knowledge of relevant NFRs, and the software testing techniques can be used to assess them. However, further investigations can go further by providing technologies that use the NFR-BoK as input to support the prioritization of the NFRs a software should meet and describe how to assess the selected NFRs using the available testing techniques.

It involves the identification of the software characteristics influencing the selection of relevant NFRs. For instance, understanding the relevance of each NFR according to the software domain. Besides, automated tools can support these future technologies aiding the use of the body of knowledge by software development organizations.

Understand the trade-off of handling different NFRs in a project – time and budget pressures prevent software development organizations from handling each NFR separately. Thus, many NFRs must be handled together over the software life cycle. For example, there is no time to assess the system usability, then the system performance, after the system usability, and so on. Therefore, researches investigating how to assess multiple NFRs at the same time is needed.

Researchers have been studying the conflicts between NFRs, but they are concerned with determining how the accomplishment of a particular NFR can prevent the software to achieve other NFRs [Boehm and In 1996] [Mairiza et al. 2013]. However, we are suggesting researches that answer how to handle multiple NFRs during the software development process.

We believe that organizing a body of knowledge about NFRs can improve the understanding of these requirements and be the first step in dealing with trade-offs during the process of developing a software system.

New non-functional testing approaches and evolution of the existing ones – as we could identify NFRs classified as relevant to the success of software systems, and there are no testing approaches to assess these NFRs, the need to create new testing approaches is evident.

Besides, the existing non-functional testing approaches do not fully cover the software testing dimensions (testing level, phases, and techniques). It is warning as some failure categories can be identified only by specific software strategy. Moreover, most of the identified non-functional testing approaches were not formally evaluated, or they were evaluated through a weak experimental method (proof of concept).

Identify new security and performance verification practices – a case study is a restrictive method regarding the size of the investigated population. Thus, we were able to research only four organizations. Therefore, new S&P verification practices can be identified by performing the case study in other contexts or using another research method (*e.g.*, literature review, survey). For the first option, it is possible to reuse our case study planning and artifacts to replicate the investigation.

Identify new factors influencing the decision-making of S&P verification – similar to the S&P verification practices, and the decision-making factors were also identified using a small population. Therefore, further researches can identify new decision-making factors used in distinct contexts.

Improve the understanding regarding the influence of moderator factors in S&P verification – we presented the moderator factors stating that they are relevant to the S&P verification. However, we do not provide information about how they influence the S&P verification. For instance, what are the consequences of producing precise requirements (MF3)? Does it increase the defect identification rate? Does it decrease the S&P verification cost? Does it decrease the effort? Such questions are unanswered. Therefore, further researches can address these questions in the future.

Improve the understanding regarding the applicability of the S&P moderator factors – some of the factors may not be relevant in specific contexts. For instance, it could be unfeasible to have a cross-functional team (MF2) in the context of extra-small organizations (1, 2, or 3 employees). Therefore, further researches can address each of the factors to understand the context in which they can be applied.

It is possible to apply similar reasoning to the actions used to promote each moderator's factors. Thus, there is also a need for further researches investigating the context they could be applied.

Understand why software development organizations do not use the proposed technologies to manage S&P requirements – one of the moderator factors states about the importance of having clear security and performance requirements (MF3). However, despite the existence of different technologies supporting the management of S&P requirements [McDermott and Fox 1999] [Tondel et al. 2008] [Mohammed et al. 2017], we realized that the organizations manage the S&P requirements ad-hoc.

Therefore, future research should evaluate the applicability of the proposed technologies and propose ways to introduce them in the software development industry.

Identify the criteria used to select a suitable methodology – we concluded that it is vital to use a systematic S&P verification methodology. However, we do not provide indications of how to choose a suitable methodology. Therefore, further researches should identify the factors influencing the choice of a suitable methodology according to the context of each software development organization and the software characteristics.

REFERENCES

Afreen, N., Khatoon, A., and Sadiq, M. (2016). A Taxonomy of Software's Non-functional Requirements. In: Satapathy, S. C.; Raju, K. S.; Mandal, J. K.; Bhateja, V.[Eds.]. Philosophy and Rhetoric. Advances in Intelligent Systems and Computing. New Delhi: Springer India. v. 379p. 47–53.

Alexander, I. (Jan 2003). Misuse cases: use cases with hostile intent. IEEE Software, v. 20, n. 1, p. 58–66.

Ameller, D., Ayala, C., Cabot, J. and Franch, X. (Sep 2012). How do software architects consider non-functional requirements: An exploratory study. In 2012 20th IEEE International Requirements Engineering Conference (RE), Chicago, IL, USA. IEEE. http://ieeexplore.ieee.org/document/6345838/.

Ameller, D., Galster, M., Avgeriou, P. and Franch, X. (2013). The Role of Quality Attributes in Service-Based Systems Architecting: A Survey. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). v. 7957 LNCSp. 200–207.

Ameller, D., Galster, M., Avgeriou, P. and Franch, X. (8 Jun 2016). A survey on quality attributes in service-based systems. Software Quality Journal, v. 24, n. 2, p. 271–299.

Arif, M. M., Shang, W., and Shihab, E. (3 Jun 2018). Empirical study on the discrepancy between performance testing results from virtual and physical environments. Empirical Software Engineering, v. 23, n. 3, p. 1490–1518.

Atifi, M., Mamouni, A., and Marzak, A. (2017). A Comparative Study of Software Testing Techniques. In: El Abbadi, A.; Garbinato, B.[Eds.]. Lecture Notes in Computer Science. Cham: Springer International Publishing. v. 10299p. 373–390.

Aurum, A., and Wohlin, C. (2005). Requirements Engineering: Setting the Context. In: Aurum, A.; Wohlin, C.[Eds.]. Engineering and Managing Software Requirements. Berlin/Heidelberg: Springer-Verlag. v. 44p. 1–15.

Ayalew, T., Kidane, T., and Carlsson, B. (2013). Identification and Evaluation of Security Activities in Agile Projects. p. 139–153, Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-41488-6_10

Bajpai, V., and Gorthi, R. P. (Mar 2012). On non-functional requirements: A survey. In 2012 IEEE Students' Conference on Electrical, Electronics, and Computer

Science. IEEE. http://ieeexplore.ieee.org/document/6184810/.

Baltes, S., Moseler, O., Beck, F., and Diehl, S. (Oct 2015). Navigate, Understand, Communicate: How Developers Locate Performance Bugs. In 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). IEEE. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7321208.

Barbir, A., Hobbs, C., Bertino, E., Hirsch, F., and Martino, L. (2007). Challenges of Testing Web Services and Security in SOA Implementations. Test and Analysis of Web Services. Berlin, Heidelberg: Springer Berlin Heidelberg. p. 395–440.

Barney, S., Aurum, A., and Wohlin, C. (Jun 2008). A product management challenge: Creating software product value through requirements selection. Journal of Systems Architecture, v. 54, n. 6, p. 576–593.

Basili, V. R., and Selby, R. W. (Dec 1987). Comparing the Effectiveness of Software Testing Strategies. IEEE Transactions on Software Engineering, v. SE-13, n. 12, p. 1278–1296.

Becha, H. and Amyot, D. (1 Mar 2012). Non-Functional Properties in Service Oriented Architecture – A Consumer’s Perspective. Journal of Software, v. 7, n. 3, p. 575–587.

Berntsson Svensson, R., Gorschek, T., and Regnell, B. (2009). Quality Requirements in Practice: An Interview Study in Requirements Engineering for Embedded Systems. Requirements Engineering: Foundation for Software Quality. v. 4542p. 218–232.

Bertolino, A. (May 2007). Software Testing Research: Achievements, Challenges, Dreams. In Future of Software Engineering (FOSE ’07). IEEE. http://ieeexplore.ieee.org/document/4221614/.

Beznosov, K., and Kruchten, P. (2005). Towards agile security assurance. In Proceedings of the 2004 workshop on New security paradigms - NSPW ’04. ACM Press. http://portal.acm.org/citation.cfm?doid=1065907.1066034.

Boehm, B. and In, H. (1996). Identifying quality-requirement conflicts. In Proceedings of the Second International Conference on Requirements Engineering. IEEE Comput. Soc. Press. http://ieeexplore.ieee.org/document/491448/.

Boehm, B., and Kukreja, N. (Oct 2015). An Initial Ontology for System Qualities. INCOSE International Symposium, v. 25, n. 1, p. 341–356.

Boehm, B. W. (Jan 1984). Software Engineering Economics. IEEE Transactions on Software Engineering, v. SE-10, n. 1, p. 4–21.

Borg, A., Yong, A., Carlshamre, P., and Sandahl, K. (2003). The Bad Conscience of

Requirements Engineering : An Investigation in Real-World Treatment of Non-Functional Requirements. Third Conference on Software Engineering Research and Practice in Sweden (SERPS’03), Lund, n. January 2003, p. 1–8.

Bozic, J., and Wotawa, F. (Mar 2014). Security Testing Based on Attack Patterns. In 2014 IEEE Seventh International Conference on Software Testing, Verification, and Validation Workshops. IEEE. http://ieeexplore.ieee.org/document/6825631/.

Bozic, J., and Wotawa, F. (Aug 2015). PURITY: A Planning-based secURITY Testing Tool. In 2015 IEEE International Conference on Software Quality, Reliability, and Security - Companion. IEEE. http://ieeexplore.ieee.org/document/7322124/.

Broy, M. (May 2015). Rethinking Nonfunctional Software Requirements. Computer, v. 48, n. 5, p. 96–99.

Brucker, A. D., and Sodan, U. (2014). Deploying static application security testing on a large scale. v. P-228p. 91–101.

Bulej, L., Bureš, T., Horký, V., et al. (13 Mar 2017). Unit testing performance with Stochastic Performance Logic. Automated Software Engineering, v. 24, n. 1, p. 139–187.

Camacho, C. R., Marczak, S., and Cruzes, D. S. (Aug 2016). Agile Team Members Perceptions on Non-functional Testing: Influencing Factors from an Empirical Study. In 2016 11th International Conference on Availability, Reliability, and Security (ARES). IEEE. http://ieeexplore.ieee.org/document/7784622/.

Caro, A., Calero, C., Caballero, I., and Piattini, M. (15 Dec 2008). A proposal for a set of attributes relevant to Web portal data quality. Software Quality Journal, v. 16, n. 4, p. 513–542.

Carroll, C., Falessi, D., Forney, V., et al. (Oct 2015). A Mapping Study of Software Causal Factors for Improving Maintenance. In 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). IEEE. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7321183.

Choliz, J., Vilas, J., and Moreira, J. (Aug 2015). Independent Security Testing on Agile Software Development: A Case Study in a Software Company. In the 2015 10th International Conference on Availability, Reliability, and Security. IEEE. http://ieeexplore.ieee.org/document/7299961/.

Chung, L. and Do Prado Leite, J. C. S. (2009). On Non-Functional Requirements in Software Engineering. International Series in Software Engineering. p. 363–379.

Chung, L., Nixon, B. A., Yu, E., and Mylopoulos, J. (2000). The NFR Framework in Action. Non-Functional Requirements in Software Engineering. Boston, MA: Springer US. p. 15–45.

Conradi, R., Amarjit Singh Marjara, Hantho, Ø., Frotveit, T. and Skåtevik, B. (1999). A study of inspections and testing at Ericsson , Norway. In Rewritten version of chapter presented at PROFES'99.

Corbin, J., and Strauss, A. (1998). Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. 2nd. ed. London: SAGE Publications.

Cruzes, D. S., and Dyba, T. (Sep 2011). Recommended Steps for Thematic Synthesis in Software Engineering. In 2011 International Symposium on Empirical Software Engineering and Measurement. IEEE. http://ieeexplore.ieee.org/document/6092576/.

Cruzes, D. S., and Othmane, L. Ben (2018). Threats to validity in empirical software security research. Boca Raton: CRC Press.

Daud, N. M. N., and Kadir, W. M. . W. (2012). Systematic mapping study of quality attributes measurement in service-oriented architecture. In the 8th International Conference on Information Science and Digital Content Technology (ICIDT). http://www.scopus.com/inward/record.url?eid=2-s2.0-84866998707&partnerID=40&md5=a7fad9b2f2d879fe0b8d347d9936099b%5Cn http://www.scopus.com/inward/record.url?eid=2-s2.0-84866998707%7B&%7DpartnerID=40%7B&%7Dmd5=a7fad9b2f2d879fe0b8d347d9936099b.

Dazhi Zhang, Donggang Liu, Yu Lei, et al. (Jun 2010). Detecting vulnerabilities in C programs using trace-based testing. In 2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN). IEEE. http://ieeexplore.ieee.org/document/5544310/.

De Win, B., Scandariato, R., Buyens, K., Grégoire, J. and Joosen, W. (Jul 2009). On the secure software development process: CLASP, SDL, and Touchpoints compared. Information and Software Technology, v. 51, n. 7, p. 1152–1171.

Delamaro, M. eduardo, Maldonado, J. C. and Jino, M. (2007). Introdução ao Teste de Software. Rio de Janeiro: Elsevier.

Dukes, L., Yuan, X. and Akowuah, F. (Apr 2013). A case study on web application security testing with tools and manual testing. In 2013 Proceedings of IEEE Southeastcon. IEEE. http://ieeexplore.ieee.org/document/6567420/.

Ebert, C. (Jan 1998). Putting requirement management into praxis: dealing with nonfunctional requirements. Information and Software Technology, v. 40, n. 3, p. 175–185.

Eckhardt, J., Vogelsang, A., and Fernández, D. M. (2016). Are "non-functional" requirements really non-functional? In Proceedings of the 38th International Conference on Software Engineering - ICSE ’16. ACM Press. http://dl.acm.org/citation.cfm?doid=2884781.2884788.

Erdogan, G., Meland, P. H., and Mathieson, D. (2010). Security Testing in Agile Web Application Development - A Case Study Using the EAST Methodology. Lecture Notes in Business Information Processing. v. 48 LNBIPp. 14–27.

Ermilov, T., Khalili, A. and Auer, S. (Jan 2014). Ubiquitous Semantic Applications. International Journal on Semantic Web and Information Systems, v. 10, n. 1, p. 66–99.

Fagan, M. E. (1976). Design and code inspections to reduce errors in program development. IBM Systems Journal, v. 15, n. 3, p. 182–211.

Felderer, M., Büchler, M., Johns, M., et al. (2016). Security Testing: A Survey. v. 101p. 1–51.

Ferme, V., and Pautasso, C. (2017). Towards Holistic Continuous Software Performance Assessment. In Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion - ICPE ’17 Companion. ACM Press. http://dl.acm.org/citation.cfm?doid=3053600.3053636.

Ferrell, B. and Oostdyk, R. (Mar 2010). Modeling and performance considerations for automated fault isolation in complex systems. In the 2010 IEEE Aerospace Conference. IEEE. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5446817.

Gaisbauer, S., Kirschnick, J., Edwards, N. and Rolia, J. (Sep 2008). VATS: Virtualized-Aware Automated Test Service. In 2008 Fifth International Conference on Quantitative Evaluation of Systems. IEEE. http://ieeexplore.ieee.org/document/4634959/.

Garousi, V., and Felderer, M. (2017). Living in two different worlds: A comparison of industry and academic focus areas in software testing. IEEE Software, n. October, p. 1–1.

Ge, X., Paige, R. F., Polack, F. A. C., Chivers, H., and Brooke, P. J. (2006). Agile development of secure web applications. In Proceedings of the 6th international conference on Web engineering - ICWE ’06. ACM Press. http://portal.acm.org/citation.cfm?doid=1145581.1145641.

Ghindici, D., Grimaud, G., Simplot-Ryl, I., Liu, Y. and Traore, I. (Nov 2006). Integrated Security Verification and Validation: Case Study. In Proceedings. 2006 31st IEEE Conference on Local Computer Networks. IEEE. http://ieeexplore.ieee.org/document/4116692/.

Gilb, T., and Graham, D. (1993). Software Inspection. Addison-Wesley Professional.

Glinz, M. (2005). Rethinking the Notion of Non-Functional Requirements. Proceedings of the Third World Congress for Software Quality, n. September, p. 55–64.

Glinz, M. (Oct 2007). On Non-Functional Requirements. In 15th IEEE International Requirements Engineering Conference (RE 2007). IEEE. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4384163.

Guo, W., Fu, X. and Feng, J. (Dec 2010). A Data-Driven Software Testing Tools Integration System. In 2010 International Conference on Computational Intelligence and Software Engineering. IEEE. http://ieeexplore.ieee.org/document/5676878/.

Haley, C. B., Laney, R., Moffett, J. D., and Nuseibeh, B. (Jan 2008). Security Requirements Engineering: A Framework for Representation and Analysis. IEEE Transactions on Software Engineering, v. 34, n. 1, p. 133–153.

Hammani, F. Z. (May 2014). Survey of Non-Functional Requirements modeling and verification of Software Product Lines. In 2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS). IEEE. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6861085.

Harjumaa, L., and Tervonen, I. (2010). Introducing Mitigation Use Cases to Enhance the Scope of Test Cases. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). v. 6434 LNCSp. 337–353.

Horký, V., Libič, P., Marek, L., Steinhauser, A., and Tůma, P. (2015). Utilizing Performance Unit Tests To Increase Performance Awareness. In Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering - ICPE ’15. ACM Press. http://dl.acm.org/citation.cfm?doid=2668930.2688051.

Howard, M., and Lipner, S. (2006). The Security Development Lifecycle: SDL: A process for Developing Demonstrably More Secure Software. Microsoft Press.

Hui, Z. and Huang, S. (14 Dec 2012). Comparison of SETAM with security use case and security misuse case: A software security testing study. Wuhan University Journal of Natural Sciences, v. 17, n. 6, p. 516–520.

IEEE-610.12 (1990). 610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology. IEEE Computer Society, v. 121990.

Livonen, J., Mäntylä, M. V., and Itkonen, J. (2010). Characteristics of high performing testers. In Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM ’10. ACM Press. http://portal.acm.org/citation.cfm?doid=1852786.1852862.

ISO/IEC 25010 (2011). ISO 25010 - Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. Iso/Iec Fdis 25010:2011, v. 2010, p. 1–34.

ISO 29119-1, (2013). Software and systems engineering - Software testing - Part 1: Concepts and definitions. v. 2013.

ISO 29119-2, (2013). Software and systems engineering - Software testing - Part 2: Test processes. v. 2013.

Jackson, S. L. (2012). Research methods and statistics: A critical thinking approach. 4. ed. Cengage learning.

Johnson, M. J., Ho, C.-W., Maximilien, E. M., and Williams, L. (May 2007). Incorporating Performance Testing in Test-Driven Development. IEEE Software, v. 24, n. 3, p. 67–73.

Joorabchi, M. E., Mesbah, A. and Kruchten, P. (Oct 2013). Real Challenges in Mobile App Development. In 2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement. IEEE. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6681334.

Jürjens, J. (2002). Using UMLsec and goal trees for secure systems development. In Proceedings of the 2002 ACM symposium on Applied computing - SAC ’02. ACM Press. http://portal.acm.org/citation.cfm?doid=508791.508990.

Kabbani, N., Tilley, S., and Pearson, L. (Apr 2010). Towards an evaluation framework for SOA security testing tools. In the 2010 IEEE International Systems Conference. IEEE. http://ieeexplore.ieee.org/document/5482322/.

Kalinowski, M. (2011). Uma abordagem para prevenção de defeitos provenientes de inspeções para apoiar a melhoria dos processos de engenharia do software. Federal University of Rio de Janeiro.

Keramati, H., and Mirian-Hosseinabadi, S.-H. (Mar 2008). Integrating software development security activities with agile methodologies. In 2008 IEEE/ACS International Conference on Computer Systems and Applications. IEEE. http://ieeexplore.ieee.org/document/4493611/.

Kim, G.-H., Kim, Y.-G. and Chung, K.-Y. (14 Oct 2015). Towards virtualized and automated software performance test architecture. Multimedia Tools and Applications, v. 74, n. 20, p. 8745–8759.

Kim, H., Choi, B., and Yoon, S. (2009). Performance testing based on test-driven development for mobile applications. In Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication - ICUIMC ’09. ACM Press. http://www.scopus.com/inward/record.url?eid=2-s2.0-

70349093096&partnerID=40&md5=61b34bacedccc680fec6061fc8c91fbd.

Kongsli, V. (2006). Towards agile security in web applications. In Companion to the 21st ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications - OOPSLA ’06. ACM Press. http://portal.acm.org/citation.cfm?doid=1176617.1176727.

Labs, M. (2016). McAfee Threat Predictions. https://www.mcafee.com/us/resources/reports/rp-threats-predictions-2016.pdf.

Laitenberger, O., and Atkinson, C. (1999). Generalizing perspective-based inspection to handle object-oriented development artifacts. In Proceedings of the 21st international conference on Software engineering - ICSE ’99. ACM Press. http://portal.acm.org/citation.cfm?doid=302405.302680.

Larsson, J., Borg, M. and Olsson, T. (17 Feb 2016). Testing Quality Requirements of a System-of-Systems in the Public Sector - Challenges and Potential Remedies.

Linåker, Johan; Sulaman, Sardar Muhammad; Maiani de Mello, Rafael; Höst, M. (2015). Guidelines for conducting surveys in software engineering v. 1.1. n. May.

Lincoln, Y. and Guba, E. (2016). Naturalistic Inquiry. Encyclopedia of Research Design. 2455 Teller Road, Thousand Oaks California 91320 United States: SAGE Publications, Inc.

Luo, J., and Yang, W. (Jan 2014). A Performance Testing Tool for Source Code. Applied Mechanics and Materials, v. 490–491, p. 1553–1559.

Macdonald, F., and Miller, J. (1995). Modelling Software Inspection Methods for the Application of Tool Support. v. 44, n. 0, p. 1–30.

Mairiza, D., Zowghi, D., and Gervasi, V. (Sep 2013). Conflict characterization and Analysis of Non Functional Requirements: An experimental approach. In 2013 IEEE 12th International Conference on Intelligent Software Methodologies, Tools, and Techniques (SoMeT). IEEE. http://ieeexplore.ieee.org/document/6645645/.

Mairiza, D., Zowghi, D., and Nurmuliani, N. (2010). An investigation into the notion of non-functional requirements. In Proceedings of the 2010 ACM Symposium on Applied Computing - SAC ’10. ACM Press. http://portal.acm.org/citation.cfm?doid=1774088.1774153.

Martin, E., and Xie, T. (May 2007). Automated Test Generation for Access Control Policies via Change-Impact Analysis. In Third International Workshop on Software Engineering for Secure Systems (SESS’07: ICSE Workshops 2007). IEEE. http://ieeexplore.ieee.org/document/4273331/.

Matoussi, A., and Laleau, R. (2008). A Survey of Non-Functional Requirements in Software Development Process. Laboratory of Algorithmic, Complexity, and

Logic. http://lacl.univ-paris12.fr/Rapports/TR/TR-LACL-2008-7.pdf.

Maxwell, J. A. (2012). Qualitative Research Design: An Interactive Approach. 3rd Ed. ed. Sage Publications. v. 41.

McDermott, J., and Fox, C. (1999). Using abuse case models for security requirements analysis. In Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99). IEEE Comput. Soc. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=816013.

McDonald, J., Dowd, M., and Schuh, J. (2006). The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities. Addison-Wesley Professional.

Mcgraw, G. (Mar 2004). Software security. IEEE Security & Privacy Magazine, v. 2, n. 2, p. 80–83.

Meira, J. A., De Almeida, E. C., Kim, D., Filho, E. R. L., and Le Traon, Y. (2016). “Overloaded!” — A Model-Based Approach to Database Stress Testing. p. 207–222.

Merriam-Webster.com (2019a). Property. https://www.merriam-webster.com/dictionary/property, [accessed on Jun 14].

Merriam-Webster.com (2019b). Condition. https://www.merriam-webster.com/dictionary/condition, [accessed on Jun 14].

Merriam-Webster (2011). Conjecture. https://www.merriam-webster.com/dictionary/conjecture.

Mohammed, N. M., Niazi, M., Alshayeb, M., and Mahmood, S. (Feb 2017). Exploring software security approaches in software development lifecycle: A systematic mapping study. Computer Standards & Interfaces, v. 50, n. May 2016, p. 107–115.

Montagud, S., Abrahão, S. and Insfran, E. (24 Sep 2012). A systematic review of quality attributes and measures for software product lines. Software Quality Journal, v. 20, n. 3–4, p. 425–486.

Myers, G. J. (1 Sep 1978). A controlled experiment in program testing and code walkthroughs/inspections. Communications of the ACM, v. 21, n. 9, p. 760–768.

Netto, M. A. S., Menon, S., Vieira, H. V., et al. (May 2011). Evaluating Load Generation in Virtualized Environments for Software Performance Testing. In 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Ph.D. Forum. IEEE. http://ieeexplore.ieee.org/document/6008948/.

Ng, S. P., Murnane, T., Reed, K., Grant, D., and Chen, T. Y. (2004). A preliminary survey on software testing practices in Australia. In 2004 Australian Software

Engineering Conference. Proceedings. IEEE. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1290464%5Cnpapers3://publication/doi/10.1109/ASWEC.2004.1290464.

NIST (2002). The economic impacts of inadequate infrastructure for software testing. National Institute of Standards & Technology.

Omotunde, H., and Ibrahim, R. (2015). A Review of Threat Modelling and Its Hybrid Approaches to Software Security Testing. v. 10, n. 23, p. 17657–17664.

Omotunde, H., Ibrahim, R., and Ahmed, M. (2018). An optimized attack tree model for security test case planning and generation. Journal of Theoretical and Applied Information Technology, v. 96, n. 17, p. 5635–5649.

OWASP (2014). 4.0 Testing Guide. OWASP Foundation, n. Cc, p. 224.

Parveen, T., and Tilley, S. (Apr 2008). A Research Agenda for Testing SOA-Based Systems. In 2008 2nd Annual IEEE Systems Conference. IEEE. http://ieeexplore.ieee.org/document/4519032/.

Rashid, M., Ardito, L., and Torchiano, M. (Oct 2015). Energy Consumption Analysis of Algorithms Implementations. In 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). IEEE. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7321198.

Ribeiro, V. V. (2017). Desafios na Verificação de Segurança de Sistemas de Software. In Workshop de Qualidade de Produto de Software.

Ribeiro, V. V., and Travassos, G. H. (2016). Testing Non-Functional Requirements: Lacking of Technologies or Researching Opportunities. XV Brazilian Symposium on Software Quality.

Ribeiro, V. V. (2017).Tecnologia de apoio à composição de estratégias de verificação de segurança e desempenho, XV Workshop de Teses e Dissertações em Qualidade de Software.

Ribeiro, V. V., Cruzes, D. S., and G. H. Travassos (2018) A Perception of the Practice of Software Security and Performance Verification, in 2018 25th Australasian Software Engineering Conference (ASWEC), pp. 71–80.

Runeson, P. and Höst, M. (19 Apr 2009). Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering, v. 14, n. 2, p. 131–164.

Russell, G. W. (Jan 1991). Experience with inspection in ultra-large-scale development. IEEE Software, v. 8, n. 1, p. 25–31.

Santos, I. de S., Santos, A. R., and Neto, P. de A. dos S. (2011). Reusing Functional Testing in order to Decrease Performance and Stress Testing Costs. In

Proceedings of the 23rd International Conference on Software Engineering & Knowledge Engineering (SEKE 2011), n. January, p. 470–474.

Shu, X. and Maurer, F. (Aug 2007). A Tool for Automated Performance Testing of Java3D Applications in Agile Environments. International Conference on Software Engineering Advances (ICSEA, 2007). IEEE. http://ieeexplore.ieee.org/document/4299917/.

Sindre, G., and Opdahl, A. (2001). Capturing security requirements through misuse cases. NIK, 2001; Norsk Informatikkonferanse, 2001, p. 12.

Siponen, M., Baskerville, R., and Kuivalainen, T. (2005). Integrating Security into Agile Development Methods. In Proceedings of the 38th Annual Hawaii International Conference on System Sciences. IEEE. http://ieeexplore.ieee.org/document/1385609/.

Soares, L. R., Potena, P., Machado, I. do C., Crnkovic, I. and Almeida, E. S. De (Aug 2014). Analysis of Non-functional Properties in Software Product Lines: A Systematic Review. In 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications. IEEE. http://ieeexplore.ieee.org/document/6928831/.

Sommerville, I. (2011). Software Engineering. 9. ed. Pearson.

Sonia and Singhal, A. (Feb 2012). Integration Analysis of Security Activities from the Perspective of Agility. In 2012 Agile India. IEEE. http://ieeexplore.ieee.org/document/6170016/.

Stallings, W., Brown, L., Bauer, M., and Howard, M. (2013). Computer Security: Principles and Practice. 2nd. ed. Willford Press.

Stephanow, P., and Khajehmoogahi, K. (Mar 2017). Towards Continuous Security Certification of Software-as-a-Service Applications Using Web Application Testing Techniques. In 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA). IEEE. http://ieeexplore.ieee.org/document/7921007/.

Study, A. C. (2014). MEFORMA Security Evaluation Methodology - A Case Study. In Proceedings of the 4th International Conference on Pervasive and Embedded Computing and Communication Systems. SCITEPRESS - Science and Technology Publications. http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0004919902670274.

Sucipto, S., and Wahono, R. S. (2015). A Systematic Literature Review of Requirements Engineering for Self-Adaptative Systems. Journal of Software

Engineering, v. 1, n. 1, p. 55–71.

Svensson, R. B., Host, M., and Regnell, B. (Sep 2010). Managing Quality Requirements: A Systematic Review. In 2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications. IEEE. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5598106.

Symantec (2017). Symantec Internet Security Threat Report. https://www.symantec.com/security-center/threat-report.

Thompson, H. H. (Jul, 2003). Why security testing is hard. IEEE Security & Privacy Magazine, v. 1, n. 4, p. 83–86.

Threat, I. B. M. X. and Index, I. (2017). IBM X-Force Threat Intelligence Index. https://www.ibm.com/security/data-breach/threat-intelligence-index.html.

Tondel, I. A., Jaatun, M. G., and Meland, P. H. (Jan 2008). Security Requirements for the Rest of Us: A Survey. IEEE Software, v. 25, n. 1, p. 20–27.

Travassos, G. H., Shull, F., and Carver, J. (2000). A Family of Reading Techniques for OO Design Inspections. In Workshop Qualidade de Software at Brazilian Symposium on Software Engineering. Brazilian Computer Society. http://www.cs.umd.edu/projects/SoftEng/ESEG/papers/pdf/wqs00.pdf.

Tricco, A. C., Antony, J., Zarin, W., et al. (16 Dec 2015). A scoping review of rapid review methods. BMC Medicine, v. 13, n. 1, p. 224.

Türpe, S. (24 Jul 2008). Security Testing: Turning Practice into Theory. In 2008 IEEE International Conference on Software Testing Verification and Validation Workshop. IEEE. http://ieeexplore.ieee.org/document/4567023/.

Ullah, S., Iqbal, M., and Khan, A. M. (Jul 2011). A survey on issues in non-functional requirements elicitation. In International Conference on Computer Networks and Information Technology. IEEE. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6020890.

Vara, J. L. de La, Wnuk, K., Svensson, R. B., Sánchez, J., and Regnell, B. (2011). An Empirical Study on the Importance of Quality Requirements in Industry. In 2011 23rd International Conference on Software Engineering and Knowledge Engineering (SEKE 2011). http://www.ksi.edu/seke/Proceedings/seke11/190_Jose_Luis_de_la_Vara.pdf.

Vaughn, R. B., Henning, R., and Fox, K. (Apr 2002). An empirical study of industrial security-engineering practices. Journal of Systems and Software, v. 61, n. 3, p. 225–232.

Wäyrynen, J., Bodén, M., and Boström, G. (2004). Security Engineering and eXtreme Programming: An Impossible Marriage? p. 117–128.

Wewers, M. E., and Lowe, N. K. (Aug 1990). A critical review of visual analogue scales in the measurement of clinical phenomena. Research in Nursing & Health, v. 13, n. 4, p. 227–236.

Weyuker, E. J., and Vokolos, F. I. (2000). Experience with performance testing of software systems: issues, an approach, and case study. IEEE Transactions on Software Engineering, v. 26, n. 12, p. 1147–1156.

Williams, L., Meneely, A., and Shipley, G. (May 2010). Protection Poker: The New Software Security "Game." IEEE Security & Privacy Magazine, v. 8, n. 3, p. 14–20.

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14. ACM Press. http://dl.acm.org/citation.cfm?doid=2601248.2601268.

Wood, M., Roper, M., Brooks, A., and Miller, J. (1997). Comparing and Combining Software Defect Detection Techniques: A Replicated Empirical Study. Proceedings of the 6th European SOFTWARE ENGINEERING Conference Held Jointly with the 5th ACM SIGSOFT International Symposium on Foundations of Software Engineering, v. 44, n. 0, p. 262–277.

Woodraska, D., Sanford, M., and Xu, D. (2011). Security mutation testing of the FileZilla FTP server. In Proceedings of the 2011 ACM Symposium on Applied Computing - SAC '11. ACM Press. http://portal.acm.org/citation.cfm?doid=1982185.1982493.

Yee, G. (2006). Recent Research in Secure Software. Technical Report.

Zhioua, Z., Short, S., and Roudier, Y. (Jul 2014). Static Code Analysis for Software Security Verification: Problems and Approaches. In 2014 IEEE 38th International Computer Software and Applications Conference Workshops. IEEE. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6903113.

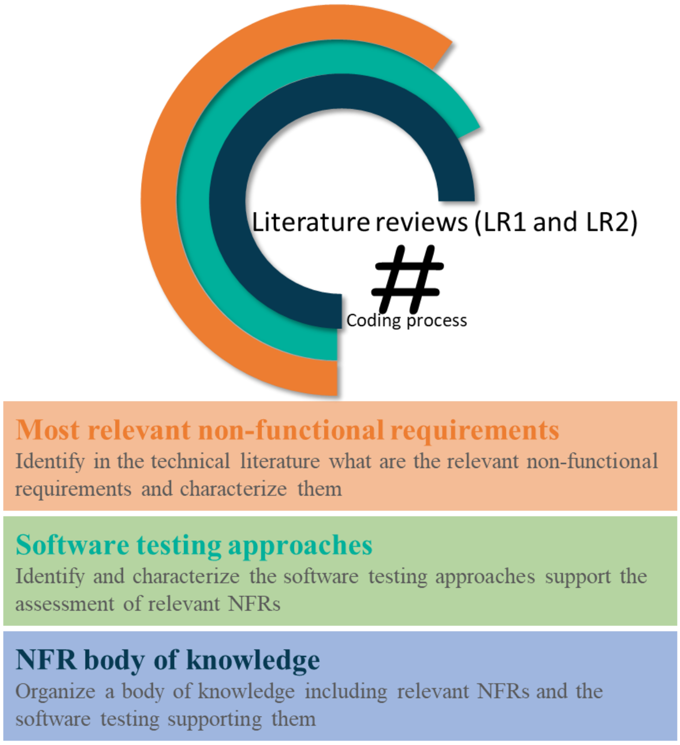
Zhu, H. S., Lin, C., and Liu, Y. D. (May 2015). A Programming Model for Sustainable Software. In 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. IEEE. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7194624.

Appendix A The full research methodology

This appendix presents the methodology followed by this thesis. It describes the methodology divided into two cycles. The first, presenting the methodological issues of two literature reviews. The second presents the methodology of the performed case study and the complementary studies used to confirm its result. Finally, we present the threat to validity.

Thesis methodology overview

The first research cycle was intended to identify and characterize the relevant NFRs and identify the software testing approaches that could be applied to those requirements. As illustrated in Figure 74, two *structured literature reviews* were performed. The first one to identify the relevant non-functional requirements (LR1) and the second one to identify the software testing approaches (LR2). After that, the information resulting from LR1 and LR2 were qualitatively analyzed through a *coding process* and then organized in a body of knowledge (NFR-BoK).



Most relevant non-functional requirements
Identify in the technical literature what are the relevant non-functional requirements and characterize them

Software testing approaches
Identify and characterize the software testing approaches support the assessment of relevant NFRs

NFR body of knowledge
Organize a body of knowledge including relevant NFRs and the software testing supporting them

Figure 74 - Overview of the methodology of cycle 1

Figure 75 presents the methodology of cycle 2. A *case study* was performed aiming to characterize the S&P verification practices performed by software development organizations, identifying the decision-making factors and moderator factors influencing such practices.

Next, we performed a set of *rapid reviews* [Tricco et al. 2015] to improve the confidence of the moderator factors, and, finally, we performed a *survey* to confirm the moderator factors pertinence with practitioners.

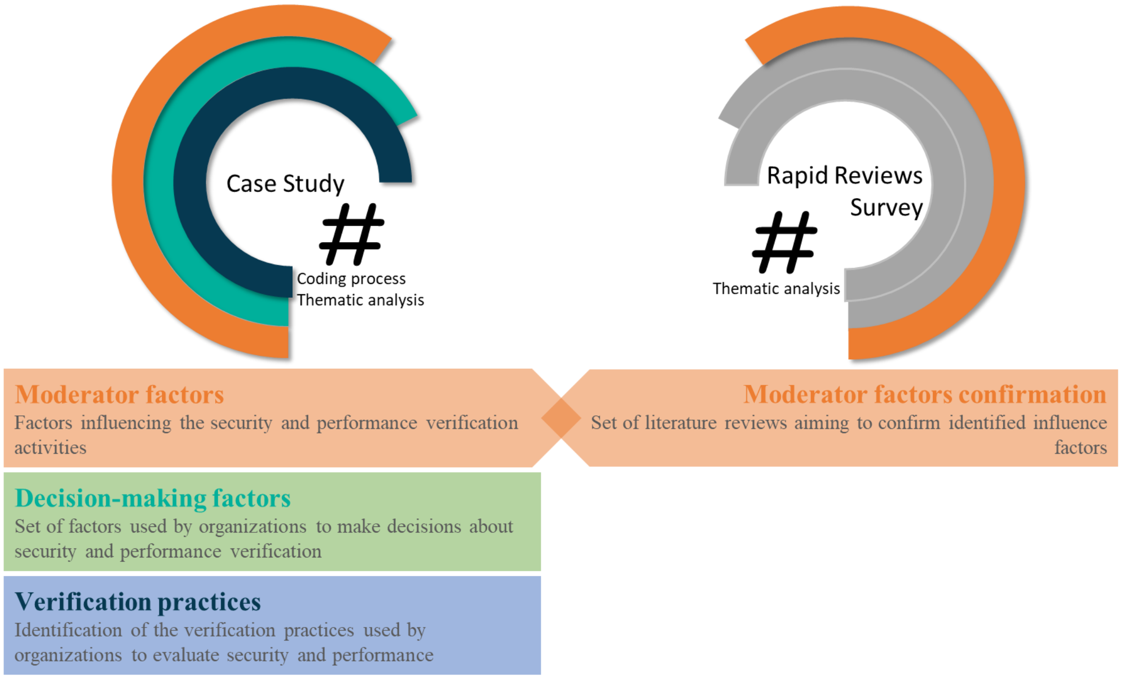

Case Study
#
Coding process
Thematic analysis
Rapid Reviews
Survey
#
Thematic analysis
Moderator factors
Factors influencing the security and performance verification activities
Moderator factors confirmation
Set of literature reviews aiming to confirm identified influence factors
Decision-making factors
Set of factors used by organizations to make decisions about security and performance verification
Verification practices
Identification of the verification practices used by organizations to evaluate security and performance

Figure 75 - Overview of the methodology of cycle 2

Cycle 1: the methodology to build a body of knowledge

A literature review on the relevant non-functional requirements

The LR1 follows the protocol presented in Appendix B. It aimed to identify the relevant NFRs, searching for secondary studies or surveys presenting NFRs mentioned as relevant by practitioners. It was carried out in March 2015, retrieving papers from 1996 to 2015, and driven by the following research question:

- **What are the most relevant non-functional requirements, according to software practitioners?**

Aiming to answer this question, it was built a search string with two parts — the first one to filter systematic reviews or survey researches; the second part to limit the search for non-functional requirements.

("systematic review" OR "systematic literature review" OR "systematic mapping" OR "systematic investigation" OR "systematic analysis" OR "mapping study" OR "structured literature review" OR "evidence-based literature review" OR "survey" OR "review of studies" OR "structured review" OR "systematic review" OR "literature review" OR "systematic literature review" OR "literature analysis" OR "meta-analysis" OR "analysis of research" OR "empirical body of knowledge" OR "overview of existing research" OR "body of published knowledge")

AND

("non-functional requirements" OR "non-functional software requirement" OR "non-behavioral requirement" OR "non-functional property" OR "quality attribute" OR "quality requirement" OR "software characteristic")

The search string was applied to the search engine Scopus so that 266 papers could be found. After the search string execution, the primary author reads the title and abstract of each paper, classifying them on Included or Excluded using the following criteria:

- **Inclusion criteria**
  - The paper must present a systematic literature review, a survey or a similar study; AND
  - The paper must Identify relevant non-functional requirements; AND
  - The paper must express practitioners’ opinions, OR practitioners must.
- **Exclusion criteria**
  - The paper is not available; AND
  - The paper presents an already included study (duplicity).

Then, another author of this thesis analyzed the excluded papers set and reclassified them on Included or kept it out. Table 3 shows the number of papers of LR1. It is important to note that there is a paper manually included because the search engine was not indexing correctly.

Table 42 - Amount of LR1 papers

Papers found	Excluded	Included	Manual included	Total included
266	252	14	1	15

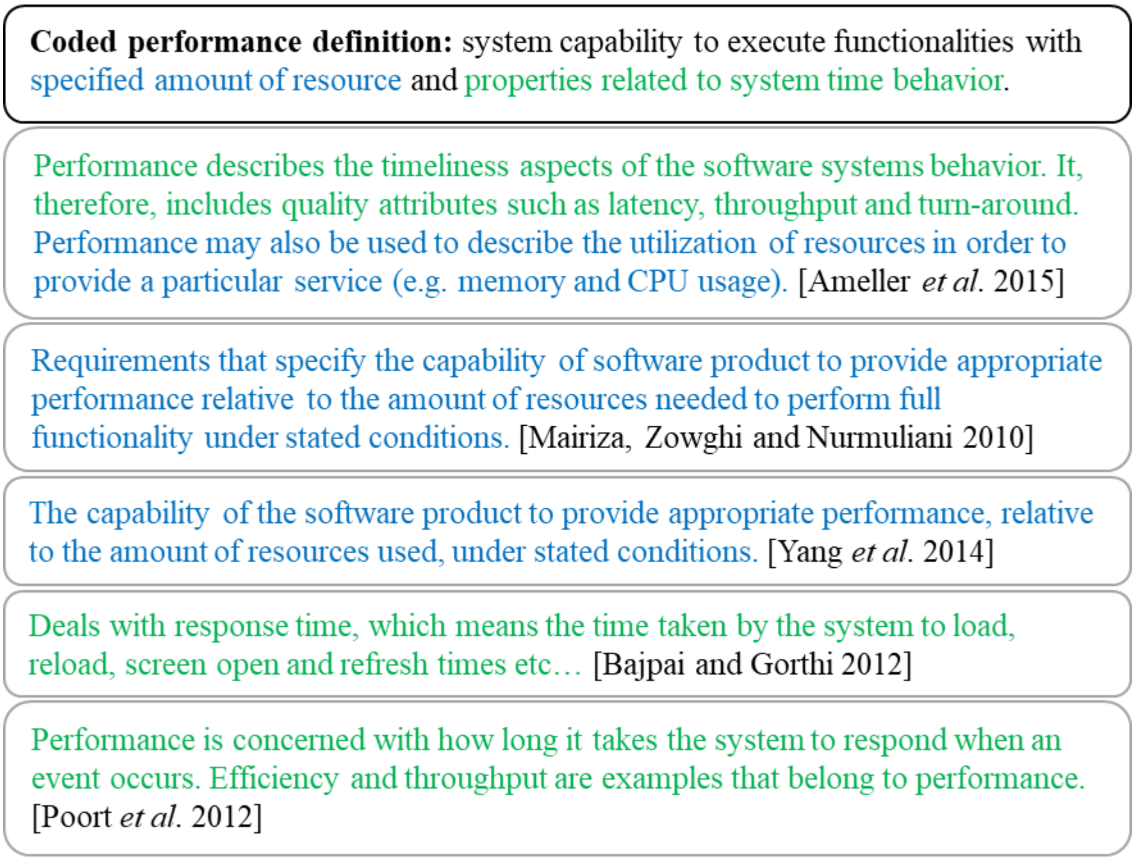
The authors analyzed the 15 included papers and extracted the following information:

- **Reference information:** it aims to identify the paper by title, author, and publisher;
- **Abstract:** it aims to contextualize the research when to query the form;
- **Study type:** it identifies the type of study, *e.g.*, systematic literature review, survey, along with others;
- **System domain/type:** it identifies the system type or domain in which the research has been done;
- **Non-functional requirements:** it identifies the NFRs presented in the paper and their description when presented.

At this point, we had the extraction form of each NFR. Analyzing extracted information was possible to realize that some NFRs did not have a description. Thus, for the sake of comprehensibility, the group of NFRs without description was not considered at this point.

The next followed step is related to the understanding of each NFR to organize them into a body of knowledge. However, it was possible to identify a lack of agreement regarding NFRs' names and descriptions. Thus, to organize all described NFRs, we performed *open coding*, as described in Grounded Theory [Corbin and Strauss 1998].

Figure 5 shows an example of the resulting code on performance definition where the first box is the final performance definition extracted from subject papers. For instance, the highlighted text in blue associate performance on resource consumption and the text of green color to time behavior.



Coded performance definition: system capability to execute functionalities with specified amount of resource and properties related to system time behavior.

Performance describes the timeliness aspects of the software systems behavior. It, therefore, includes quality attributes such as latency, throughput and turn-around. Performance may also be used to describe the utilization of resources in order to provide a particular service (e.g. memory and CPU usage). [Ameller *et al*. 2015]

Requirements that specify the capability of software product to provide appropriate performance relative to the amount of resources needed to perform full functionality under stated conditions. [Mairiza, Zowghi and Nurmuliani 2010]

The capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions. [Yang *et al*. 2014]

Deals with response time, which means the time taken by the system to load, reload, screen open and refresh times etc… [Bajpai and Gorthi 2012]

Performance is concerned with how long it takes the system to respond when an event occurs. Efficiency and throughput are examples that belong to performance. [Poort *et al*. 2012]

Figure 76 - Open coding example

The coding process allowed identifying a hierarchical structure of NFRs. Figure 6 shows this structure in which the class NFR represents high-level abstract system properties such as Usability, Security, and Performance. These properties are perceived through a set of *Sub_NFR*, which are also NFR but, they represent more specific system properties such as Navigability (Usability), Confidentiality (Security), or Resource Consumption (Performance). Moreover, some NFRs may own *Operationalization,* which are features that must be present on the system for it meets the NFR. For instance, the usage of an image compression algorithm is one operationalization of Resource Consumption.

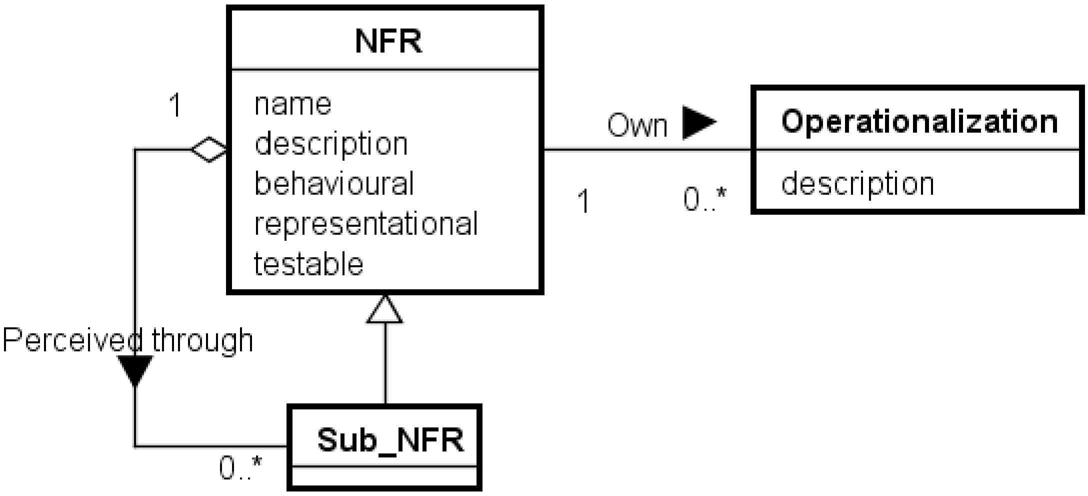


Figure 77 - Hierarchical structure of NFRs

A literature review on the non-functional testing approaches

The second structured literature review (LR2) follows the protocol presented in Appendix C. It aims to identify proposed software testing approaches concerned with NFRs and their testing covering. In this context, testing covering is regarding software testing process phases, levels, and techniques of the proposed approaches. Unlike LR1, LR2 does not look at other literature reviews because previous *ad-hoc* searches do not retrieve that kind of study concerning testing approaches for NFRs. LR2 was performed in March 2016, naturally retrieving papers from 1991 to 2005, and driven by the following research question:

- **What are the software testing approaches used to test non-functional requirements?**

Two parts search string was organized and submitted to the Scopus search engine. The first section of the search string aims to limit the results to software testing approaches, and the second one restricts the search to non-functional requirements.

("software test design" OR "software test suite" OR "software test" OR "software testing" OR "system test design" OR "system test suite" OR "system test" OR "system testing" OR "middleware test" OR "middleware testing" OR "property based software test" OR "property based software testing" OR "fault detection" OR "failure detection" OR "GUI test" OR "Graphical User Interfaces test" OR "test set" OR "non-functional testing" OR "model based testing" OR "test case" OR "testing infrastructure" OR "testing approach" OR "testing environment")

AND

("non-functional requirements" OR "non-functional software requirement" OR "non-behavioral requirement" OR "non-functional property" OR "quality attribute" OR "quality requirement" OR "software characteristic")

The filtering process followed a similar procedure as in LR1. The inclusion/exclusion are:

- **Inclusion criteria**
  - The paper should present a software testing procedure, technique, or any other type of proposal about non-functional requirements software testing.
- **Exclusion criteria**
  - The paper is not available; AND
  - The paper presents an already included study (duplicity).

Table 4 shows the number of papers of LR2. There were three papers manually included because they were not directly available through the Scopus search engine.

Table 43 - Amount of LR2 papers

Papers found	Excluded	Included	Manual included	Total included
331	287	44	3	47

The 47 papers were analyzed using an extraction form with the following fields:

- **Reference Information:** it aims to identify the paper by title, author, and publisher;
- **Abstract:** it aims to give an overall idea of the paper subject;
- **System Domain/Type:** it indicates whether the approach is proposed to specific software domain or type, *e.g.*, embedded systems, telecommunication systems;
- **Test Phase:** the coverage of the testing approach regarding testing process phases: Planning, Design, Implementation, Execution, and Analysis;
- **Test Level:** testing granularity, with the options Unit, Integration, System, Acceptance, Not Informed, and Not Applied;
- **Test Technique:** with the options Structural, Functional, Fault Based, Not Informed, and Not Applied;

- **Evaluation:** it represents how the software testing approach has been evaluated, *e.g.*, proof of concept, experiment, case study, simulation, not applied, and not informed. Evaluation values emerged from the subject papers;
- **Non-Functional Requirements Considered:** it represents the list of NFRs considered by the software testing approach with their description.

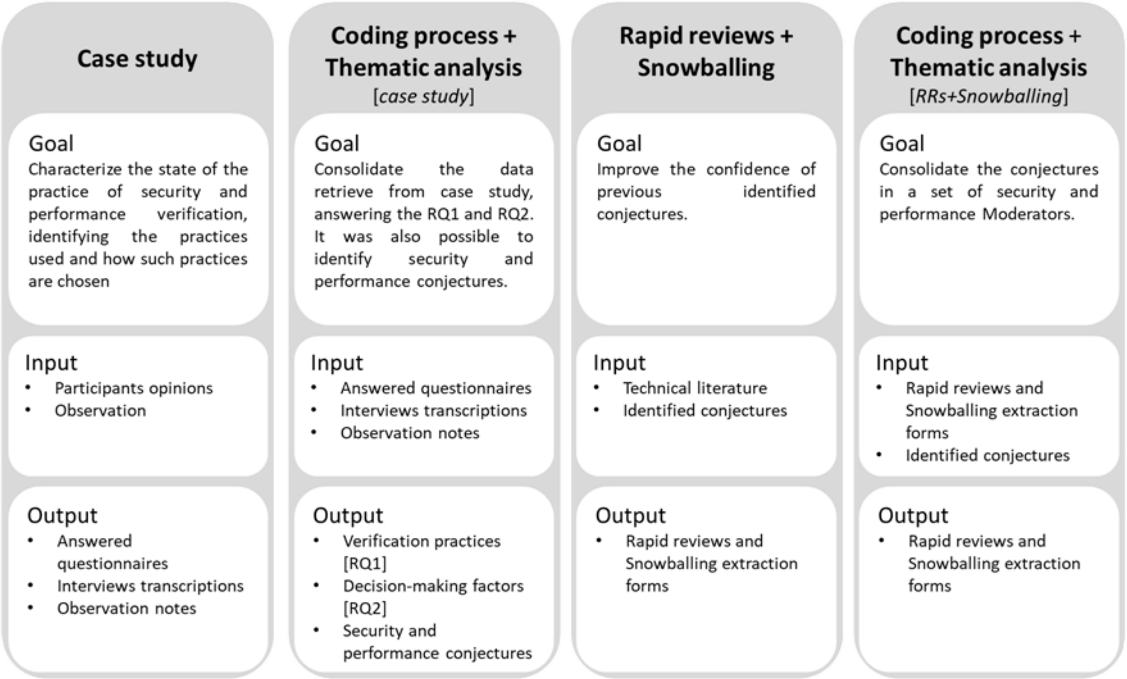
After data extraction and analysis, the information regarding software testing approaches was included in the NFR-BoK. Thus, besides includes the relevant non-functional requirements and their characterization, it also contains information about which testing techniques are suitable to assess each of the NFRs.

Cycle 2: the methodology to characterize security and performance verification

The methodology of cycle 2 was divided into two main parts. First, we executed a *case study* with four cases, on four different organizations, aiming to identify how security and performance verification are performed and how the decision-making regarding security and performance verification is done on software development organizations. Thus, we created research questions addressing the identification of practices and the factors influencing the decision-making of security and performance verification.

However, while analyzing the collected data through a coding process, it was possible to answer the previously defined research questions and also identify some significant findings of the security and performance verification. Such findings were organized in a set of conjectures (inference formed statements without proof or sufficient evidence [Merriam-Webster 2011]) about security and performance verification.

Given the importance of these unexpected but essential findings, we performed a set of secondary studies, in the form of *rapid reviews* (RRs) [Tricco et al. 2015] and *Snowballing* [Wohlin 2014], searching for support literature to confirm them. After analyzing the extraction forms of secondary studies through a *coding process* and thematic analysis, it was possible to consider the conjectures as moderator factors that influence the verification of security and performance. Figure 78 presents a big picture of the methodology, and the next sections describe the methodological details of each kind of study.



	Case study	Coding process + Thematic analysis *[case study]*	Rapid reviews + Snowballing	Coding process + Thematic analysis *[RRs+Snowballing]*
Goal	Characterize the state of the practice of security and performance verification, identifying the practices used and how such practices are chosen	Consolidate the data retrieve from case study, answering the RQ1 and RQ2. It was also possible to identify security and performance conjectures.	Improve the confidence of previous identified conjectures.	Consolidate the conjectures in a set of security and performance Moderators.
Input	• Participants opinions • Observation	• Answered questionnaires • Interviews transcriptions • Observation notes	• Technical literature • Identified conjectures	• Rapid reviews and Snowballing extraction forms • Identified conjectures
Output	• Answered questionnaires • Interviews transcriptions • Observation notes	• Verification practices [RQ1] • Decision-making factors [RQ2] • Security and performance conjectures	• Rapid reviews and Snowballing extraction forms	• Rapid reviews and Snowballing extraction forms

Figure 78 - General research methodology of cycle 2

Case study methodology

This part of the research is classified as a characterization case study, and it follows the recommendations presented in the guidelines proposed by Runeson and Höst [2009]. Table 9 shows the mains sections of the case study protocol (Appendix E).

Table 44 - Case study research protocol

Objectives
Understand how security and performance verification has been performed in the software development industry.
Scope
Software development organizations that perform Security OR Performance Verification activities.
Research method
• Multiple case studies: one of them including an observational phase and others with only semi-structured interviews and questionnaire data collection phases ○ 1 organization with 1 project as the main case: including observational, semi-structured, and questionnaires data collection method; ○ 3 organizations with 1 project per each as complementary cases: including semi-structured and questionnaires data collection method; • Flexible design ○ Trying to improve the protocol during the study execution • Predominantly qualitative

• Criteria for case selection ○ Projects in progress for at least two months.
Data sources
Organizations employers, researcher observations, institutional websites
Unit of analysis
Software development projects, including security or performance verification activities.

Besides, the research protocol contains the research questions grouped by two mains subjects. The first aiming to identify the practices used by organizations on the security and performance verification and to characterize such practices. The second one aiming to identify factors influencing the decisions regarding the verification practices:

- **RQ 1 Which are the practices used by the organizations to support the verification of security and performance?**
  - RQ 1.1 What are the standard techniques?
  - RQ 1.2 Which definition of done do they adopt?
  - RQ 1.3 How is the level of automation?
  - RQ 1.4 What are the assets covered?
- **RQ 2 How does the organization define its security and performance verification strategies?**
  - RQ 2.1 What are the factors influencing the decision-making on security and performance verification strategies?
  - RQ 2.2 When does the decision on the verification strategy happen?
  - RQ 2.3 How often the decisions on the verification strategy occur?
  - RQ 2.4 Who is responsible for the decision making on the verification strategy?

After, the organizations were selected by convenience. We tried to identify among our personal contacts people who work in software development organizations. Thus, we selected ten people working in different organizations, and we start the initial contact. However, three organizations do not answer to our request, and three of them said they do not perform security or performance verification. Thereby, we were able to select a set of four organizations as subjects of the study.

The data collection process of the case study

A set of artifacts was used to support the study, including the case study protocol (Appendix E) and a presentation letter (Appendix F). Additionally, instruments were used to collect data to answer the research questions directly and to formalize the research agreement and the characterization of the organizations and participants. The first author of this thesis filled the instruments when collecting data during observations and interviews. The participants filled out the questionnaire when it was used to collect data. Table 10 presents a summary of the used instruments.

Table 45 - Case study instruments description

ID	Description	Objectives
P1	Case study protocol	The protocol followed by the case study.
C1	Presentation letter	A letter used to make the first contact with the organizations, characterizing the researchers involved and the objectives of the study.
I1	Organization agreement term	After the organization agrees with the research, its representative signs the agreement term. It marks the beginning of the case study.
I2	Participant agreement term	In the first contact with each participant, they must sign the consent term to allow us to collect and use data.
I3	Organization characterization	Data can be gathered from different sources as the participants and organization websites. It was filled in during different moments of the case study execution.
I4	Project characterization	It supports data collection while interviewing participants through a questionnaire.
I5	Participant characterization	It supports data collection after interviewing participants through a questionnaire.
I6	Verification practices identification	Data collected in different stages of case study execution. It was gathered from observation, interviews, and questionnaires.
I7	Identification of decision-making factors	Data collected in different stages of case study execution. It was gathered from observation, interviews, and questionnaires.
I8	Participant opinion	It supports data collection after interviewing participants through a questionnaire.

The first step of the data collection process was to sign the organizational agreement represented by the instrument I1. This step is crucial to ensure the integrity of research and organization privacy. In the second step, data regarding organizational characterization (I3), project characterization (I4), and participant characterization (I5) were collected. Furthermore, the participant agreement term is signed because it is the first contact with some of the participants.

The semi-structured interview step used instrument I4 to collect data related to the development process and the environment. This step aims to understand more fully how the organization operates. The instruments I5 and I6 were used to identify the security and performance practices and decision-making factors, respectively. It is important to note that some semi-structured interviews were recorded. In these cases, the instruments were not filled, but they were used as a guide for the interviews.

The observational step aims to confirm previously collected data and to understand how the verification practices were performed in detail. This step was performed in only one of the organizations, and its main output is the researcher's manual notes.

The last step is the application of a questionnaire to collect the opinion of the participant about security and performance verification.

It is essential to mention that the researcher collected data in the form of manual notes at every stage. Figure 19 shows the data collection process, and the instruments were filled.

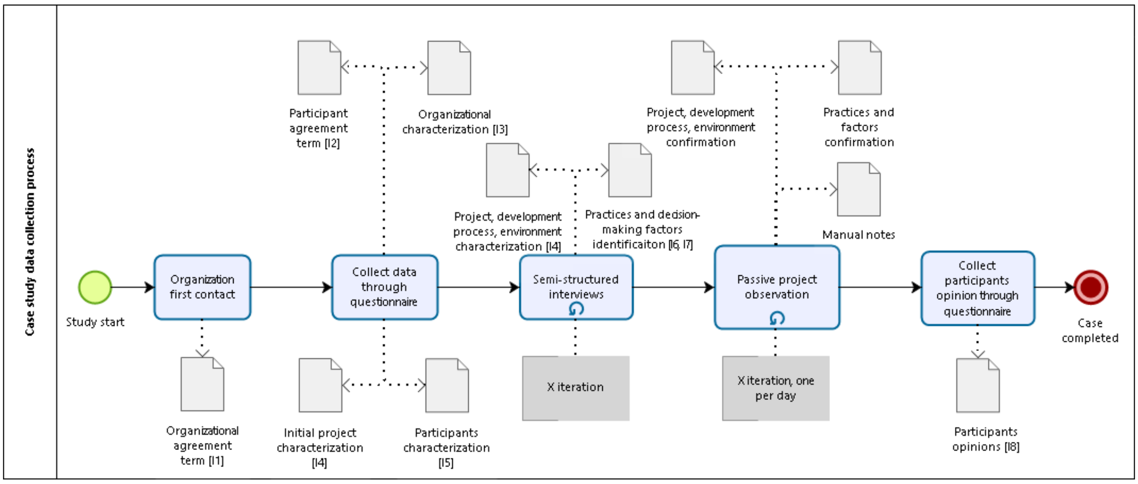

Case study data collection process
Study start
Organization first contact
Organizational agreement term [I1]
Participant agreement term [I2]
Organizational characterization [I3]
Collect data through questionnaire
Initial project characterization [I4]
Participants characterization [I5]
Project, development process, environment characterization [I4]
Practices and decision-making factors identificaiton [I6, I7]
Semi-structured interviews
X iteration
Project, development process, environment confirmation
Practices and factors confirmation
Manual notes
Passive project observation
X iteration, one per day
Collect participants opinion through questionnaire
Participants opinions [I8]
Case completed

Figure 79 - Data collection process

Approximately 47 artifacts were filled out. Then, the first author qualitatively analyzed them by following a coding process. The MAXQDA[8] tool, into which all instruments were imported, was used to support the coding process, and 955 artifact excerpts were grouped in 1112 codes.

The coding process was divided into two parts. The first part aimed to answer RQ1 and RQ2. Thus, the first author read the artifacts looking for verification practices and practices characteristics. Additionally, other researchers revised the generated codes in several meetings sections throughout the process. Figure 20 presents a model representing the structure of codes built during the coding process.

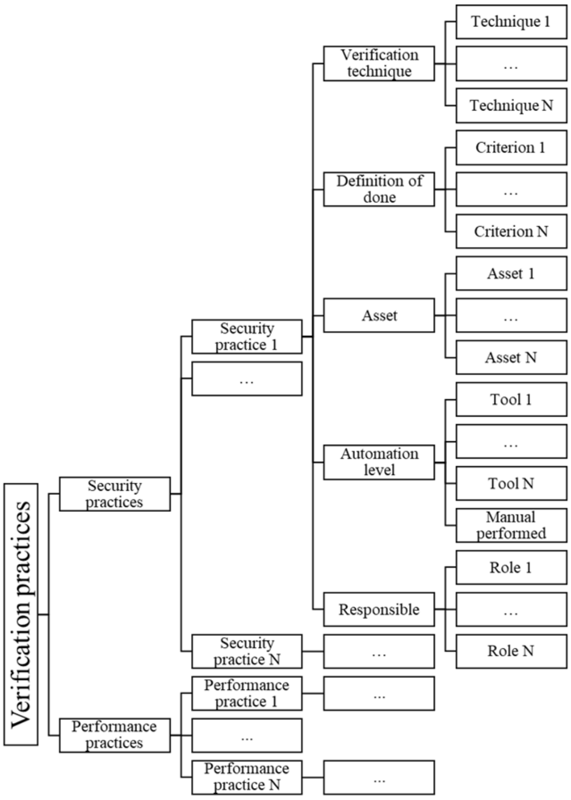


Figure 80 - Structure used to characterize verification practices (RQ 1)

The process to answer RQ 2 was similar to the process to answer RQ 1, but the aiming was to identify decision-making factors. Figure 20 represents the structure used to identify decision-making factors (RQ 2) during the coding process.

[8] https://www.maxqda.com/

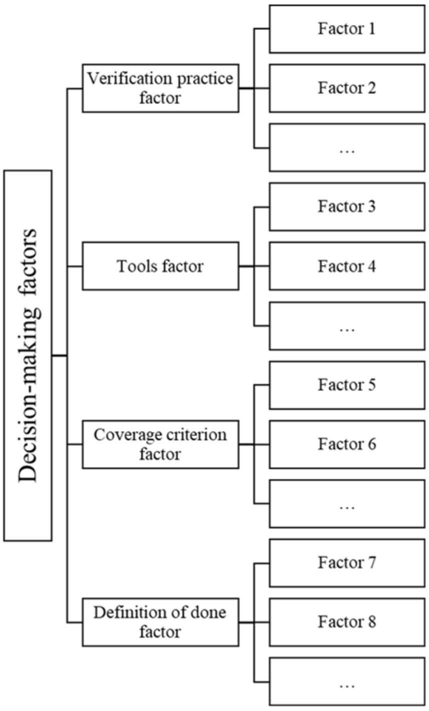


Figure 81 - Structure used to identify decision-making factors (RQ 2)

Furthermore, in the first step of the coding process, attempting to code data to answer the RQs, information was identified that does not directly support answering the research questions, but it could provide essential findings on security and performance verification, complementing the study results. Thus, this information was organized into a code category initially classified as *conjectures*. After a set of literature reviews was performed to lend credence to these conjectures, they were reclassified as security and performance verification moderator factors.

The coding process was performed to identify the conjectures following the principles of the coding phase of grounded theory [Corbin and Strauss 1998]. This methodology includes three coding phases. The open coding phase is an analytical process to identify concepts, their properties, and dimensions. In this phase, the data is fragmented and conceptually labeled in codes. When similarities between codes are found, they are grouped into categories. In axial coding, the categories can be rearranged in subcategories, and new categories can be created. Subsequently, the created categories are unified around a central core category, and relationships are established among them. This last phase was not performed in this study. Finally, step 4 of thematic synthesis [Cruzes and Dyba 2011] was used to translate the codes into themes.

Instantiating the coding process to this work, first, the first author of this thesis read every artifact creating codes related to each relevant part of the collected information. In this step, the researchers were not concerned about grouping codes into categories, but when the excerpts were similar, they were linked to the same code. For instance, when a participant said: "*The problem on using these tools is a large number of false positives*," and another said: "*There are organizations that perform security testing only with automated tools, but this generates a large number of false positives*." These two excerpts were linked to the same code "*Automated tools generate a large number of false positives*." After, other researchers reviewed the first phase of coding, suggesting corrections, mainly in the names assigned to each coding.

In the axial coding phase, Author 1 compared the created codes interactively, for example, by comparing code 1 with code 2, code 3, up to code N. Subsequently, by comparing code 2 with code 3, up to code N (code 1 is ignored because code 1 and code 2 have already been compared) and so on. When similarity was found between two codes, they were grouped into a category. This step required approximately four interactions, and some codes could not be grouped in a category. Then, the categories (as well as the codes in them) were analyzed to concatenate similar categories into one. It also required more than one interaction. Finally, Author 1 iterated through the categories to group them in a more general category, creating the structure: category > subcategory > codes. This phase was not performed linearly; the process was interrupted several times so that other researchers could review the coding already done. Thus, they suggested changes such as the inclusion of categories, change of category names, and rearrangement of codings between categories.

Finally, Author 1 iterated through the consolidated structure of categories, subcategories, and codes, translating them into a set of eight high-level themes representing conjectures about security and performance verification. Additionally, it is highly relevant to note that other researchers always iteratively check this phase in several sessions throughout the process.

Rapid reviews and snowballing methodology

Because the conjectures arose from practice and the participants' opinions, a set of rapid reviews (RR) were performed, consulting the technical literature and searching for confirmations for these conjectures.

RRs are a type of secondary study aiming to deliver evidence to practice promptly with lower effort than a traditional systematic review. To be faster, RR simplifies some steps of systematic reviews. For instance, the database search is limited, the quality appraisal is eliminated, or only one researcher is used to analyze the collected data [Tricco et al. 2015].

Eight RRs were conducted following the same protocol template, but core parts were replaced to guide each RR to target a specific conjecture. The templates used by each RR are presented in Appendices O-V. Table 14 shows the terms representing each conjecture and the keywords related to them.

Table 46 - Rapid reviews research questions structure

Conjecture	Term	Keywords
C01	suitable environment	Awareness OR recognition OR understanding OR comprehension OR importance OR relevance
C02	cross-functional team	Team* OR Staff* OR “Working Group”
C03	greater precision on the requirements definition	Requirement*
C04	suitable support tools	“support tool”
C05	suitable environment	environment*
C06	suitable methodology	methodolog*
C07	suitable planning	planning OR plan
C08	reuse of artifacts and knowledge	reuse OR reusability OR reusing

The research questions followed the structure presented in Table 11 by replacing *<<CONJECTURE>>* by the term representing the conjecture. For instance, the RR related to the verification environment had *<<CONJECTURE>>* replaced by a “*suitable environment*.”

Table 47 - Rapid reviews research questions structure

RR-RQ 1	What are the benefits of a *<<CONJECTURE>>* for the verification of security and performance?
RR-RQ 2	What problems do cause a *<<CONJECTURE>>* for the verification of security and performance?
RR-RQ 3	What are the challenges of creating a *<<CONJECTURE>>* for the verification of security and performance?
RR-RQ 4	What are the strategies to create a *<<CONJECTURE>>* for the verification of security and performance?

The search string followed the same principle as in the RQs. A search string template was defined and was adapted to target each conjecture. Table 12 presents the search string template used in each RR. For instance, in the search for conjecture

C03, <<*KEYWORD*>> was replaced by *Requirement**, for C05 it was replaced for *environment** and so on.

Template	("security verification" OR "performance verification" OR "security testing" OR "performance testing") AND (<<*KEYWORD*>>) AND ("software") AND ("benefit*" OR "problem*" OR "challenge*" OR "strateg*" OR "empirical stud*" OR "experimental stud*" OR "experiment*" OR "case stud*" OR "survey*")

After a search on the Scopus search engine, the following criteria guided the process of paper selection:

- **Inclusion criteria**
  - The paper must be in the context of software engineering; and
  - The paper must be in the context of performance and/or security verification; and
  - The paper must report a study related to <<*CONJECTURE*>> of security or performance verification activities; and
  - The paper must report an evidence-based study grounded in empirical methods (*e.g.*, interviews, surveys, case studies, formal experiment, among others) or a proof of concept; and
  - The paper must provide data to answer at least one of the RR research question; and
  - The paper must be written in the English language.

After data extraction, it was performed a snowballing to increase the literature coverage. The snowballing was backward with only one interaction, and the starter set was the included papers of the RRs. Table 16 presents the total number of papers founded, included in RR, and included in snowballing.

Table 48 - Number of papers of RR and snowballing

Conjecture	# Founded	# RR	# Snowballing	# Total
C01	63	2	0	2

C02	42	3	0	3
C03	129	6	8	14
C04	185	12	5	17
C05	117	3	1	4
C06	77	4	9	13
C07	41	3	0	3
C08	11	2	0	2

After, we imported the extraction forms in the MAXQDA tool and performed a coding and a thematic analysis process similar to the process described in Section 4.1.1.2. Thus, the output of the RRs is a set of mind maps with high-level themes representing the findings from the technical literature.

Finally, the themes of the RRs were matched to the themes of the case study. Thereby, the findings that emerged from the state of the practice are supported by the findings extracted from state of the art. Accordingly, credence was lent to the conjectures, turning them into security and performance verification moderator factors.

Survey methodology

The survey aims to confirm our interpretation of the information supporting the moderator factors and identify the relevance of each of them. The survey was performed in July 2019.

Four participants who had already participated in the case study were called the control group. Besides, the survey had 137 access but only 37 valid answers from external participants. Further information about the survey is available in Appendix W.

Appendix B LR1 Protocol: Searching relevant non-functional requirements

LITERATURE REVIEW PROTOCOL:

SEARCHING RELEVANT NON-FUNCTIONAL REQUIREMENTS

INICIAL LITERATURE REVIEW

Victor Vidigal Ribeiro

Guilherme Horta Travassos

Experimental Software Engineering Group at COPPE/UFRJ

March 2015

Review History

Date	Version	Description	Author(s)
27/03/2015	0.1	Define the initial goal and the first search string (30 papers)	Victor
-	0.2	Define Inclusion Criteria	Victor e Guilherme
-	0.3	Search String improvement (266 papers)	Guilherme e Victor

1. Main Research Scenario

Software testing aims to reveal inconsistencies between the requirements and implemented software system. Thus, inconsistencies revealed can be fixed, improving system quality.

However, although several works emphasize non-functional requirements (NFR) importance, there is an insufficient amount of addressing this type of requirement.

Lack of NFR evaluation may be the cause of low-quality systems that do not meet users' needs. In this way, the overall goal of this work investigates how NFR can be evaluated by software testing.

The first step to arriving at the overall is to know or discover what NFRs are the most important or relevant. So, to meet this specific goal, a literature review is performed with aims to find other literature reviews that present important or relevant NFR.

It is worth mentioning that this is not a systematic literature review, but it follows search string creation methodology by observing another to improve synonyms and criteria to include and exclude papers.

2. Research Protocol

Search String is build to return systematic literature reviews that identify non-functional requirements using PICO (Pai et al., 2004) principle in the way that synonyms are separated by logical connector “OR” and terms that compose the string are separated by “AND.” Thus, the first part of the search string is composed of Systematic Literature Review, and it synonyms separated by operator “OR” and second part by Non-functional Requirements and synonyms separated by operator “OR.”

In the next execution of the protocol, it is planned to evolve the String to a PICO approach (Pai et al., 2004).

2.1 Question Focus

This study's research objective is to identify relevant non-functional requirements.

2.2 Question Quality and Amplitude

- **Problem:** Non-functional requirements were neglected by software testing works. So it is essential to identify relevant requirements to define if it can be tested.
- **Question:**

  Main Question:

  What are the relevant non-functional requirements?
- **First Part:** Systematic Literature Review or Survey

  Keywords: "systematic review" OR "systematic literature review" OR "systematic mapping" OR "systematic investigation" OR "systematic analysis" OR "mapping study"

OR "structured literature review" OR "evidence-based literature review" OR "survey" OR "review of studies" OR "structured review" OR "systematic review" OR "literature review" OR "systematic literature review" OR "literature analysis" OR "meta-analysis" OR "analysis of research" OR "empirical body of knowledge" OR "overview of existing research" OR "body of published knowledge"

- **Second Part:** Non-functional Requirement or Quality Attributes

  Keywords: "non-functional requirements" OR "non-functional software requirement" OR "non-behavioral requirement" OR "non-functional property" OR "quality attribute" OR "quality requirement" OR "software characteristic"
- **Excluded Keywords:** None
- **Full String:** ("systematic review" OR "systematic literature review" OR "systematic mapping" OR "systematic investigation" OR "systematic analysis" OR "mapping study" OR "structured literature review" OR "evidence-based literature review" OR "survey" OR "review of studies" OR "structured review" OR "systematic review" OR "literature review" OR "systematic literature review" OR "literature analysis" OR "meta-analysis" OR "analysis of research" OR "empirical body of knowledge" OR "overview of existing research" OR "body of published knowledge") AND ("non-functional requirements" OR "non-functional software requirement" OR "non-behavioral requirement" OR "non-functional property" OR "quality attribute" OR "quality requirement" OR "software characteristic")

2.3 Source Selection

- **Sources Selection Criteria Definition:** Works presented as articles available on the web.
- **Studies Language:** English.
- **Source Identification**
  - **Source Search Method:** Search through Scopus web search engines.

2.4 Studies Selection

2.4.1 Studies Definition

- **Studies Inclusion and Exclusion Criteria Definition:**

  Inclusion Criteria:
  - The paper must present a systematic literature review, a survey or a similar study; AND
  - The paper must Identify relevant non-functional requirements; AND

- The paper must express practitioners' opinions, OR practitioners must.

Exclusion Criteria:

- The paper is not available; AND
- The paper presents an already included study (duplicity).

2.5 Information Extraction Strategy

For each selected paper, the following information shall be extracted and managed using the JabRef[9] reference tool and Microsoft Word:

Table 2: Information extraction fields

Field	Description
Reference information	
Abstract	
Study type	
System Domain/Type	
Non-Functional Requirements	

- **Reference information:** It aims to identify the paper by title, author, and publisher.
- **Abstract:** Its aims contextualize the researcher when to query the form.
- **Study Type:** It identifies the type of study, *e.g.*, systematic literature review, Survey.
- **System domain/Type:** It identifies the system type or domain in which research was done.
- **Non-Functional Requirements:** It identifies NFRs presented by the paper as relevant and their description.

2.5 Included papers

Ameller, D., Galster, M., Avgeriou, P., and Franch, X. A survey on quality attributes in service-based systems Software Quality Journal, Kluwer Academic Publishers, 2015
Bajpai, V. and Gorthi, R.On non-functional requirements: A survey 2012 IEEE Students' Conference on Electrical, Electronics and Computer Science: Innovation

[9] http://jabref.sourceforge.net/

for Humanity, SCEECS 2012, 2012

Becha, H. and Amyot, D. Non-functional properties in service-oriented architecture - A consumer's perspective Journal of Software, 2012, Vol. 7(3), pp. 575-587

Caracciolo, A., Lungu, M. and Nierstrasz, O. How do software architects specify and validate quality requirements? Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer Verlag, 2014, Vol. 8627 LNCS, pp. 374-389

Caro, A., Calero, C., Caballero, I. and Piattini, M. A proposal for a set of attributes relevant for Web portal data quality Software Quality Journal, 2008, Vol. 16(4), pp. 513-542

De La Vara, J., Wnuk, K., Svensson, R., SÃ¡nchez, J. and Regnell, B. An empirical study on the importance of quality requirements in industry SEKE 2011 - Proceedings of the 23rd International Conference on Software Engineering and Knowledge Engineering, 2011, pp. 438-443

Mairiza, D., Zowghi, D. and Nurmuliani, N. An investigation into the notion of non-functional requirements Proceedings of the ACM Symposium on Applied Computing, 2010, pp. 311-317

Montagud, S., AbrahÃ£o, S. and Insfran, E. A systematic review of quality attributes and measures for software product lines Software Quality Journal, 2012, Vol. 20(3-4), pp. 425-486

Nik Daud, N. and Kadir, W. Systematic mapping study of quality attributes measurement in service-oriented architecture Proceedings - ICIDT 2012, 8th International Conference on Information Science and Digital Content Technology, 2012, Vol. 3, pp. 626-631

Poort, E., Martens, N., Van De Weerd, I., and Van Vliet, H. How architects see non-functional requirements: Beware of modifiability Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2012, Vol. 7195 LNCS, pp. 37-51

Soares, L., Potena, P., Machado, I., Crnkovic, I., and De Almeida, E. d. Analysis of non-functional properties in software product lines: A systematic review Rabiser, R., Torkar, R. & Torkar, R. (ed.) Proceedings - 40th Euromicro Conference Series on Software Engineering and Advanced Applications, SEAA 2014, Institute of Electrical and Electronics Engineers Inc., 2014, pp. 328-335

Yang, Z., Li, Z. c., Jin, Z., and Chen, Y. A systematic literature review of requirements modeling and analysis for self-adaptive systems, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer Verlag, 2014, Vol. 8396 LNCS, pp. 55-71

Kaur, H., Ahamad, S., Verna, G. N., A case study upon non-functional requirements of the online banking system, International Journal of Computer Applications Technology and Research Volume 4– Issue 4, 220 - 225, 2015, ISSN:-2319–8656

Emilov, T., Khalili, A., Auer, S., Ubiquitous semantic applications: a systematic literature review

Appendix C LR2 Protocol: Software testing technics to non-functional requirements

LITERATURE REVIEW PROTOCOL:
SEARCHING SOFTWARE TESTING TECHNICS TO NON-FUNCTIONAL REQUIREMENTS

INICIAL LITERATURE REVIEW

Victor Vidigal Ribeiro
Guilherme Horta Travassos
Experimental Software Engineering Group at COPPE/UFRJ

March 2016

Review History

Date	Version	Description	Author(s)
-	0.1	Define initial string (114 papers and 37 included)	Victor
11/06/2015	0.2	Search String Improvement	Guilherme e Victor
18/06/2015	0.3	Search String improvement (81 returned, 58 included, 10 not found, and 13 excluded)	Guilherme e Victor
01/03/2016	1.0	Updating the literature review	Guilherme e Victor
18/04/2016	1.1	Inserting information about include/exclude process, *e.g.*, the amount of included/excluded paper.	Victor

1. Main Research Scenario

Software testing aims to reveal inconsistencies between the requirements and implemented software system. Thus, inconsistencies revealed can be fixed, improving system quality.

However, despite the fact that several works emphasize non-functional requirements (NFR) importance, there is an insufficient amount of addressing this type of requirement.

Lack of NFR evaluation may be the cause of low-quality systems that do not meet the user's needs. In this way, the overall goal of this work investigates how NFR can be evaluated by software testing.

This review is a second step to achieve the overall goal and aims to identify what non-functional requirements are covered by software testing and at what level the NFR is covered by software testing.

With the results of this review and review performed to identify important NFR at hand, we can matching results and identify what NFRs are the relevant uncovered by software testing.

It is worth mentioning that this is not a systematic literature review, but it follows search string creation methodology by observing another to improve synonyms and criteria to include and exclude papers.

2. Research Protocol

Search String is build to return papers about non-functional requirements software testing using PICO (Pai et al., 2004) principle in the way that synonyms are separated by logical connector “OR” and terms that compose the string are separated by “AND.” Thus, the first part of the search string is composed by the term “Non-Functional Requirements” and its synonyms separated by operator “OR” and second part by “Software Testing” and synonyms separated by operator “OR.”

In the next execution of the protocol, it is planned to evolve the String to a PICO approach (Pai et al., 2004).

2.1 Question Focus

This study's research objective is to identify relevant non-functional requirements.

2.2 Question Quality and Amplitude

- **Problem:** Software testing works neglected Non-functional requirements. So it is crucial to identify what are non-functional requirements that do not have proposed software testing procedure or technique.
- **Question:**

  Main Question:

  What are the approaches used to testing non-functional requirements?
- **First Part:** Software Testing

Keywords: "software test design" OR "software test suite" OR "software test" OR "software testing" OR "system test design" OR "system test suite" OR "system test" OR "system testing" OR "middleware test" OR "middleware testing" OR "property based software test" OR "property based software testing" OR "fault detection" OR "failure detection" OR "GUI test" OR "Graphical User Interfaces test" OR "test set" OR "non-functional testing" OR "model based testing" OR "test case" OR "testing infrastructure" OR "testing approach" OR "testing environment"

- **Second Part:** Non-Functional Requirement

  Keywords: "non-functional requirements" OR "non-functional software requirement" OR "non-behavioral requirement" OR "non-functional property" OR "quality attribute" OR "quality requirement" OR "software characteristic"
- **Excluded Keywords:** None
- **Full String:** ("software test design" OR "software test suite" OR "software test" OR "software testing" OR "system test design" OR "system test suite" OR "system test" OR "system testing" OR "middleware test" OR "middleware testing" OR "property based software test" OR "property based software testing" OR "fault detection" OR "failure detection" OR "GUI test" OR "Graphical User Interfaces test" OR "test set" OR "non-functional testing" OR "model based testing" OR "test case" OR "testing infrastructure" OR "testing approach" OR "testing environment") AND ("non-functional requirements" OR "non-functional software requirement" OR "non-behavioral requirement" OR "non-functional property" OR "quality attribute" OR "quality requirement" OR "software characteristic")

2.3 Source Selection

- **Sources Selection Criteria Definition:** Works presented as articles available on the web.
- **Studies Language:** English.
- **Source Identification**
  - **Source Search Method:** Search through Scopus web search engines.

2.4 Studies Selection

2.4.1 Studies Definition

- **Studies Inclusion and Exclusion Criteria Definition:**

  Inclusion Criteria:

- The paper should present a software testing procedure, technique, or any other type of proposal about non-functional requirements software testing.

Exclusion Criteria:

- The paper is not available; AND
- The paper presents an already included study (duplicity).

2.5 Included papers

Walter, T.a and Grabowski, J.b, “A framework for the specification of test cases for real-time distributed systems,” Information and Software Technology, 1999
Denaro, G.a and Polini, A.b and Emmerich, W.c, “Performance testing of distributed component architectures,” Springer Berlin Heidelberg, 2005
Cangussu, J.W. and Cooper, K. and Wong, E., “Reducing the number of test cases for performance evaluation of components,” 2007
Feng, Y. and Liu, X. and Kerridge, J., “A product line based aspect-oriented generative unit testing approach to building quality components,” 2007
MetsÃƒÂ¤, J.a, and Katara, M.b and Mikkonen, T.b, “Testing non-functional requirements with aspects: An industrial case study,” 2007
Dyrkon, K. and Wathne, F., “Automated testing of non-functional requirements,” 2008
Hanna, S. and Munro, M., “Fault-based web services testing,” 2008
Juszczyk, L. and Truong, H.-L. and Dustdar, S., “GENESIS - A framework for automatic generation and steering of testbeds of complex Web services,” 2008
Zou, J. and Pavlovski, C.J., “Control cases during the software development life-cycle,” 2008
Afzal, W. and Torkar, R. and Feldt, R., “A systematic review of search-based testing for non-functional system properties,” Information and Software Technology, 2009
Cangussu, J.W. and Cooper, K. and Wong, W.E., “A segment-based approach for the reduction of the number of test cases for performance evaluation of components,” International Journal of Software Engineering and Knowledge Engineering, 2009
Grossmann, J. and Serbanescu, D. and Schieferdecker, I., “Testing embedded real-time systems with TTCN-3”, 2009
Hill, J.H. and Turner, H.A. and Edmondson, J.R. and Schmidt, D.C., “Unit testing non-functional concerns of component-based distributed systems,” 2009
Kruse, P.M. and Wegener, J. and Wappler, S., “A highly configurable test system for evolutionary black-box testing of embedded systems,” 2009
Qu, B. and Ying, H. and Xie, X. and Lu, Y., “A developed dynamic environment fault injection tool for component security testing,” 2009
Romano, B.L.a and Braga E Silva, G.a and De Campos, H.F.a and Vieira, R.G.a and Da Cunha,

A.M.a and Silveira, F.F.b and Ramos, A.C.B.c, “Software testing for web-applications non-functional requirements,” 2009
Taranti, P.-G.a, and De Lucena, C.J.P.a and Choren, R.b, “An Industry Use Case: Testing SOA systems with MAS simulators,” 2009
Arnold, D.a and Corriveau, J.-P.a and Shi, W.b, “Modeling and validating requirements using executable contracts and scenarios,” 2010
Arnold, D.a and Corriveau, J.-P.a and Shi, W.b, “Modeling and validating requirements using executable contracts and scenarios,” 2010
Assad, R.E.a and Katter, T.b and Ferraz, F.S.a and Ferreira, L.P.a and Meira, S.R.L.a, “Security quality assurance on web-based application through security requirements tests: Elaboration, execution, and automation,” 2010
Bertolini, C. and Mota, A., “A framework for GUI testing based on use case design,” 2010
Farhat, S. and Simco, G. and Mitropoulos, F.J., “Using aspects for testing nonfunctional requirements in object-oriented systems,” 2010
Saifan, A. and Dingel, J., “A survey of using model-based testing to improve quality attributes in distributed systems,” 2010
De Sousa Santos, I.a and Santos, A.R.b and Neto, P.D.A.D.S.c, “Reusing functional testing in order to decrease performance and stress testing costs,” 2011
Grigorjevs, J., “Model-driven testing approach for embedded systems specifics verification based on UML model transformation,” 2011
Kormann, B. and Vogel-Heuser, B., “Automated test case generation approach for PLC control software exception handling using fault injection,” 2011
Eldh, S.a b and Sundmark, D.c d, “Robustness testing of mobile telecommunication systems: A case study on industrial practice and challenges,” 2012
Watanabe, W.M. and Fortes, R.P.M. and Dias, A.L., “Using acceptance tests to validate accessibility requirements in RIA,” 2012
Algroth, E., “Random visual GUI testing: Proof of concept,” 2013
Anisetti, M.a, and Ardagna, C.A.a and Damiani, E.a and Saonara, F.b, “A test-based security certification scheme for Web services,” ACM Transactions on the Web, 2013
Banerjee, A. and Chattopadhyay, S. and Roychoudhury, A., “Static analysis driven cache performance testing,” 2013
Gambi, A.a, and Filieri, A.b and Dustdar, S.c, “Iterative test suites refinement for elastic computing systems,” 2013
GarcÃƒÂ-a-DomÃƒÂ-nguez, A.a, and Medina-Bulo, I.a and Marcos-BÃƒÂ¡rcena, M.b, “An Approach for Model-Driven Design and Generation of Performance Test Cases with UML and MARTE,” Communications in Computer and Information Science, 2013

Manetti, V. and Petrella, L.M., “FITNESS: A framework for automatic testing of ASTERIX based software systems,” 2013
Sarwar, T. and Habib, W. and Arif, F., “Requirements based testing of software,” 2013
Toledo RodrÃƒÂ-guez, F.a and Reina, M.a and Baptista, F.a and Polo Usaola, M.b and PÃƒÂ©rez Lamancha, B.b, “Automated Generation of Performance Test Cases from Functional Tests for Web Applications,” Communications in Computer and Information Science, 2013
Zhuang, L.a and Gao, Z.a and Wu, H.b and Yang, C.X.b and Zheng, M.a, “Research on DB2 performance testing automation”, Advanced Materials Research, 2013
Batool, S. and Asghar, S., “Secure state UML: Modeling and testing security concerns of software systems using UML state machines,” Research Journal of Applied Sciences, Engineering and Technology, 2014
Marrone, S.a and Flammini, F.b and Mazzocca, N.c and Nardone, R.c and Vittorini, V.c, “Towards Model-Driven V\&V assessment of railway control systems,” International Journal on Software Tools for Technology Transfer, 2014
MetsÃƒÂ¤, J.a, and Maoz, S.b and Katara, M.c and Mikkonen, T.c, “Using aspects for testing of embedded software: Experiences from two industrial case studies,” Software Quality Journal, 2014
RodrÃƒÂ-guez, F.T.a c, and Lonetti, F.b and Bertolino, A.b and Usaola, M.P.c and Lamancha, B.P.c, “Extending UML testing profile towards non-functional test modeling,” 2014
Espada, A.R. and del Mar Gallardo, M. and SalmerÃƒÂ³n, A. and Merino, P., “Runtime verification of expected energy consumption in smartphones,” Lecture Notes in Computer Science, 2015
GarcÃƒÂ-a, B. and DueÃƒÂ±as, J.C., “Web browsing automation for applications quality control,” Journal of Web Engineering, 2015
Svensson, R.B.a and Regnell, B.b, “Aligning Quality Requirements and Test Results with QUPER's Roadmap View for Improved High-Level Decision-Making,” 2015
Ricca, F. and Tonella, P., “Analysis and testing of web applications,” 2001
Dustdar, S.a and Haslinger, S.b, “Testing of service-oriented architectures: A practical approach,” Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2004
Caliebe, P. and Lauer, C. and German, R., “Flexible integration testing of automotive ECUs by combining AUTOSAR and XCP,” 2011

Appendix D Identified security and performance verification support tools

Tool	Link
Brakeman	https://github.com/presidentbeef/brakeman
HP Fortify SCA	https://software.microfocus.com/en-us/products/static-code-analysis-sast/overview
Jenkins	https://jenkins.io/
Sonar	https://www.sonarqube.org/
Threadfix	https://threadfix.it/
Arachni	http://www.arachni-scanner.com/
OWASP Zed Attack Proxy (ZAP)	https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project
XSS ME	https://addons.mozilla.org/pt-BR/firefox/addon/xss-me/
SQL Injection ME	https://addons.mozilla.org/pt-BR/firefox/addon/sql-inject-me/
Burp Suite	https://portswigger.net/burp
Meta Exploit	https://www.metasploit.com/
NMAP	https://nmap.org/
Whatweb	https://github.com/urbanadventurer/WhatWeb
JMeter	https://jmeter.apache.org/
Postman	https://www.getpostman.com/
BlazeMeter	https://www.blazemeter.com/
Goldeneye	https://www.goldeneyeuk.com/
HTTPerf	https://github.com/httperf/httperf

SoapUI	https://www.soapui.org/
CA APM	https://www.ca.com/br/products/ca-application-performance-management.html

Appendix E Case study protocol

Objectives, issues, and topics being investigated

Objectives

Understand how security and performance verification has been performed in the software development industry.

Scope

Software development organizations that perform Security OR Performance Verification activities.

Research questions

- RQ1. What **practices** have been used in the organization to support the verification of security and performance?
  - RQ1.1 What are the **techniques** related to the practices?
  - RQ1.2 What are the **tools** used to support the practices?
  - RQ1.3 What are the **artifacts** covered by the practices?
- RQ2. How does the organization define its security and performance verification strategies?
  - RQ2.1 What are the **factors** influencing the decision-making on security and performance verification strategies?
  - RQ2.2 When does the decision on the verification strategy happen?
  - RQ2.3 How often the decisions on the verification strategy occur?
  - RQ2.4 Who is responsible for the decision making on the verification strategy?

Research method

Research method

- Multiple case studies. One of them including an observational phase and others with only semi-structured interviews and questionnaire phases.
  - One organization with one project for the main study: including observational, semi-structured, and questionnaires;
  - Two organizations with one project per each for the complementary study I: including semi-structured interviews and questionnaires
  - One organization and one project for the complementary study II: including only questionnaires.
- Flexible Design
  - try to improve the protocols during the study
- Predominantly qualitative
- Criteria for case selection:
  - Projects including security or performance software verification activities; AND
  - Projects with at least four people; AND

- Projects in progress for at least two months.

Methodology

- Grounded Theory (Glaser and Strauss, 1967), without pre-conceived theories – inductive approach – concepts will be suggested by the data rather than imposed from outside (Agar, 1980)
- Following the principles of Klein and Myers (1999)
- Collecting data from real setting for [3-4] months

Data collection

Data sources

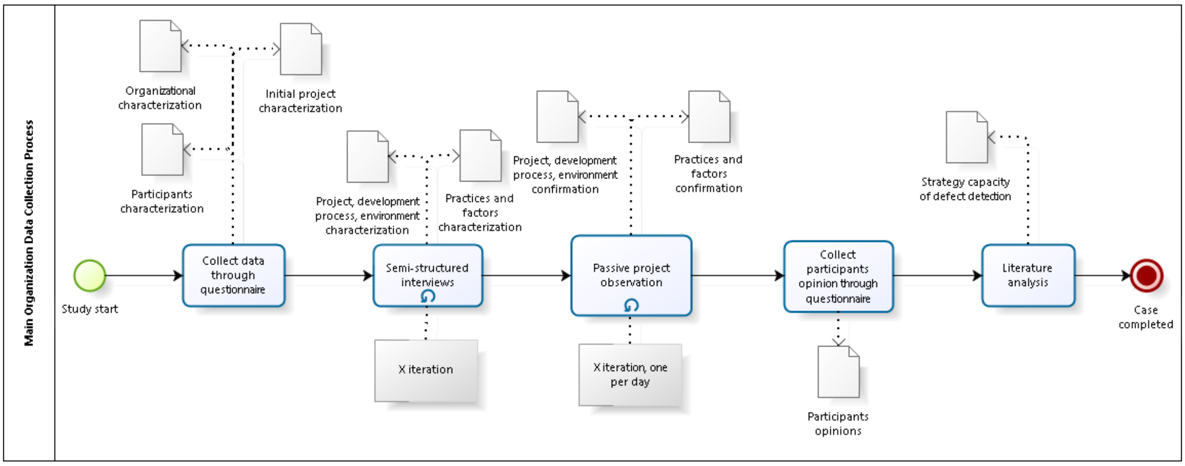
- A questionnaire to characterize the organization;
- Passive observation guided by protocols aiming to characterize the project regarding team, development process, environment, factors used to decision making, and practices being used;
- Questionnaires to characterize the participants' background;
- Semi-structured interviews aiming to characterize development process, environment, factors used to decision making, and practices being used; AND
  - obtain participants' opinions about the development process, environment, factors used to decision making, and practices being used.
- Literature analysis of the practices used regarding defect type capacity.

Data collection process

The data collection process is made to be flexible, allowing protocol improvements during the data collection process. It is divided into three different processes.

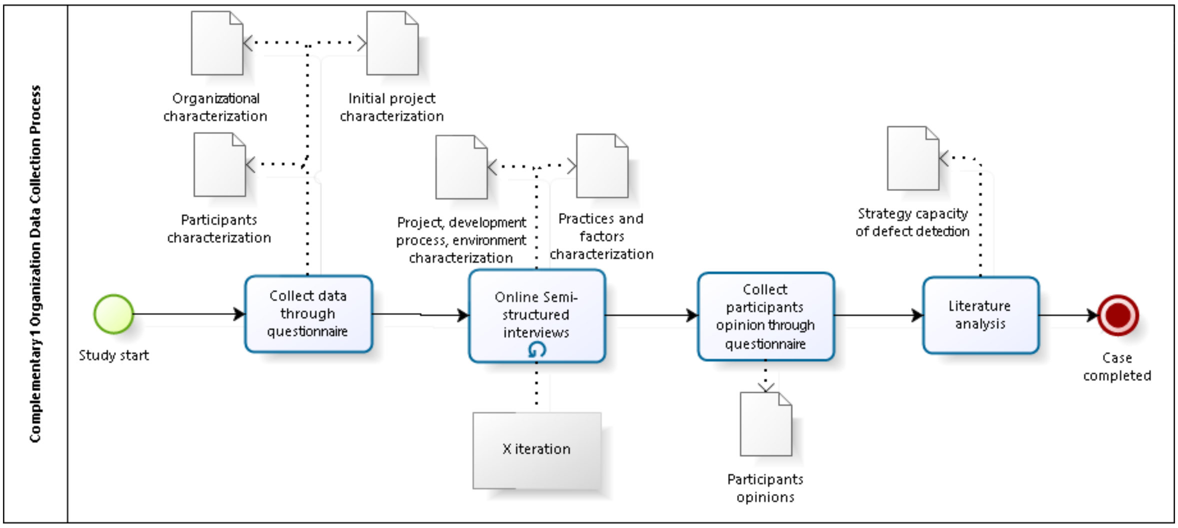
1.1.1.1 Main study data collection process

This process includes three different methods for data collection: observational, semi-structured interviews, and questionnaires.


Main Organization Data Collection Process
Organizational characterization
Initial project characterization
Participants characterization
Project, development process, environment characterization
Practices and factors characterization
Project, development process, environment confirmation
Practices and factors confirmation
Strategy capacity of defect detection
Study start
Collect data through questionnaire
Semi-structured interviews
Passive project observation
Collect participants opinion through questionnaire
Literature analysis
Case completed
X iteration
X iteration, one per day
Participants opinions

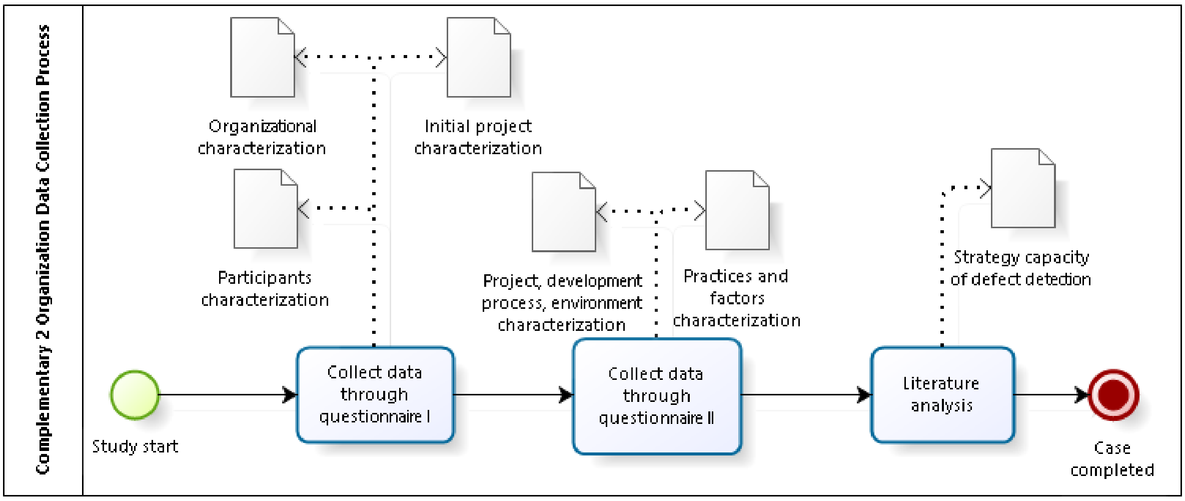
Complementary study I data collection process

This process includes two different methods for data collection: semi-structured interviews and questionnaires.


Complementary 1 Organization Data Collection Process
Organizational characterization
Initial project characterization
Participants characterization
Project, development process, environment characterization
Practices and factors characterization
Strategy capacity of defect detection
Study start
Collect data through questionnaire
Online Semi-structured interviews
Collect participants opinion through questionnaire
Literature analysis
Case completed
X iteration
Participants opinions

Complementary study II data collection process

The data collection is made through questionnaires.


Complementary 2 Organization Data Collection Process
Organizational characterization
Initial project characterization
Participants characterization
Project, development process, environment characterization
Practices and factors characterization
Strategy capacity of defect detection
Study start
Collect data through questionnaire I
Collect data through questionnaire II
Literature analysis
Case completed

Units of analysis

Software development **projects,** including security or performance verification activities.

Methods used to reduce bias

These methods are based on Stake (1995):

- **Methodological triangulation:** comparisons between data collected with qualitative methods to data collected with quantitative methods.
- **Data triangulation:** use of more than one data source or collecting the same data on different occasions:
  - comparing observation data with interview data;
  - comparing the perspectives of people from different points of view: testers view, inspectors view, managers view;

- checking for the consistency of what teams' members say about factors influencing decision-making regarding verification strategies over time;

- **Analyst triangulation:** several different researchers to analyze or to review the findings.

Data analysis

Approach to qualitative data analysis: Interpretative

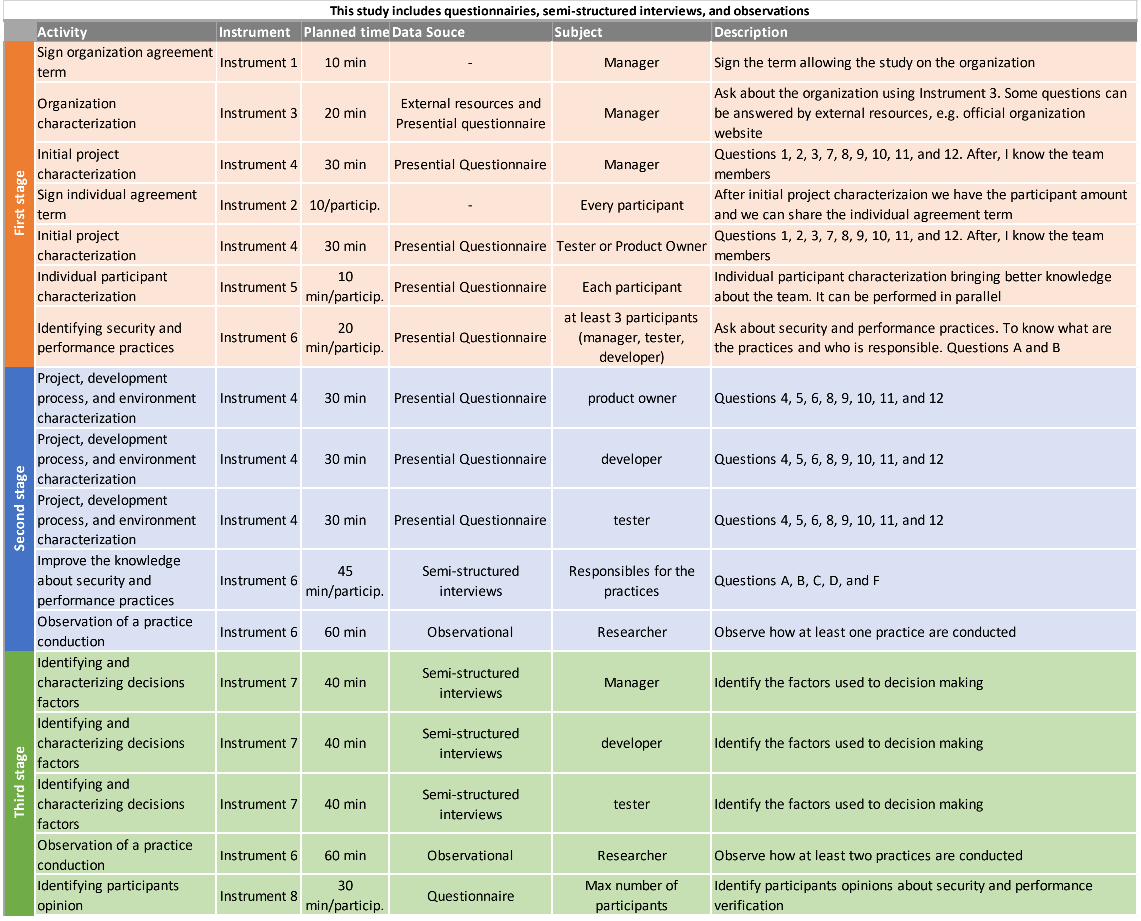
Qualitative method – dataset reduction

- Grounded-Theory
  - Interactive coding analysis with multiple researchers;
  - Next stages being directed by the discovered concepts;

Data collection protocol

Schedule

The schedules are different from primary and complementary studies.



This study includes questionnairies, semi-structured interviews, and observations

	Activity	Instrument	Planned time	Data Souce	Subject	Description
First stage	Sign organization agreement term	Instrument 1	10 min	-	Manager	Sign the term allowing the study on the organization
	Organization characterization	Instrument 3	20 min	External resources and Presential questionnaire	Manager	Ask about the organization using Instrument 3. Some questions can be answered by external resources, e.g. official organization website
	Initial project characterization	Instrument 4	30 min	Presential Questionnaire	Manager	Questions 1, 2, 3, 7, 8, 9, 10, 11, and 12. After, I know the team members
	Sign individual agreement term	Instrument 2	10/particip.	-	Every participant	After initial project characterizaion we have the participant amount and we can share the individual agreement term
	Initial project characterization	Instrument 4	30 min	Presential Questionnaire	Tester or Product Owner	Questions 1, 2, 3, 7, 8, 9, 10, 11, and 12. After, I know the team members
	Individual participant characterization	Instrument 5	10 min/particip.	Presential Questionnaire	Each participant	Individual participant characterization bringing better knowledge about the team. It can be performed in parallel
	Identifying security and performance practices	Instrument 6	20 min/particip.	Presential Questionnaire	at least 3 participants (manager, tester, developer)	Ask about security and performance practices. To know what are the practices and who is responsible. Questions A and B
Second stage	Project, development process, and environment characterization	Instrument 4	30 min	Presential Questionnaire	product owner	Questions 4, 5, 6, 8, 9, 10, 11, and 12
	Project, development process, and environment characterization	Instrument 4	30 min	Presential Questionnaire	developer	Questions 4, 5, 6, 8, 9, 10, 11, and 12
	Project, development process, and environment characterization	Instrument 4	30 min	Presential Questionnaire	tester	Questions 4, 5, 6, 8, 9, 10, 11, and 12
	Improve the knowledge about security and performance practices	Instrument 6	45 min/particip.	Semi-structured interviews	Responsibles for the practices	Questions A, B, C, D, and F
	Observation of a practice conduction	Instrument 6	60 min	Observational	Researcher	Observe how at least one practice are conducted
Third stage	Identifying and characterizing decisions factors	Instrument 7	40 min	Semi-structured interviews	Manager	Identify the factors used to decision making
	Identifying and characterizing decisions factors	Instrument 7	40 min	Semi-structured interviews	developer	Identify the factors used to decision making
	Identifying and characterizing decisions factors	Instrument 7	40 min	Semi-structured interviews	tester	Identify the factors used to decision making
	Observation of a practice conduction	Instrument 6	60 min	Observational	Researcher	Observe how at least two practices are conducted
	Identifying participants opinion	Instrument 8	30 min/particip.	Questionnaire	Max number of participants	Identify participants opinions about security and performance verification

Publication focus

Title: Characterizing the security and performance verification approaches on the Brazilian software industry

- Conferences
  - ICSE International Conference on Software Engineering, August or June, Gothenburg/Sweden, A1
    - CESI: Workshop on Conducting empirical studies in industry
    - SER&IP: Workshop on Software Engineering Research and Industrial Practice
  - FSE Foundations of Software Engineering, A1
  - ESEM Empirical Software Engineering and Measurement, A2
  - ISSTA International Symposium on Software Testing and Analysis, February, A2
  - ICST International Conference on Software Testing, Verification and Validation, Västerås/Sweden, September, B2
- Journals
  - **IJSSE International Journal of Secure Software Engineering**
  - Software Quality Journal, B2/0.787
  - Software Testing, Verification, and Reliability, B1/1.082

Possible organizations

1. INMETRO – Instituto Nacional de Metrologia, Qualidade e Tecnologia: atua de duas formas diferentes. Certificando empresas para avaliar empresas desenvolvedoras e também avaliando diretamente empresas desenvolvedoras. Segue Normas.
2. TCE – Tribunal de Contas do Estado do Rio de Janeiro: contato com o diretor geral de informática. Não tenho informações sobre o tipo de software desenvolvido, se seguem normas e se existe equipe de verificação.
3. IBM: Fiz contato com um funcionário que abriu a possibilidade de execução do case study na equipe dele. A IBM tem equipes de verificação separadas, mas muitas vezes a verificação ocorre dentro da própria equipe do projeto dependendo do orçamento do projeto. Pode ser uma ligação com a equipe de Verificação.
4. Petrobras: pessoa de contato indicou que existem normas de segurança referentes à manipulação dos dados em tempo de desenvolvimento. Não é bem o que interessa. Mas disse também que existe uma equipe que é responsável pela avaliação do software, esse pode ser o ponto.
5. <<Organization>>: ainda sem resposta
6. Tetra tech - empresa de desenvolvimento de Macaé que presta serviço para Petrobras (área de oceanografia): Contato disse que possui alguns dados confidenciais, mas que que pode tentar conseguir que um estudo seja executado.
7. Meliuz – sistema para ganhar pontos com compras e, posteriormente, recuperar parte do dinheiro. Arilo (ex aluno do GHT) trabalha lá: ainda não entrei em contato, mas tenho convicção de que é uma porta aberta. Entretanto, possivelmente não seguem nenhuma norma de segurança ou desempenho, apesar de terem preocupações com essas questões.
8. Guiando (guiando.com.br) – Empresa desenvolve um software para gestão de custos:
9. Projetos COPPETE – Prof. Zimbrão: Ainda não entrei em contato diretamente com Zimbrão por não ter os objetivos totalmente definidos, mas com pessoas

que trabalham nos projetos. Disseram que não possuem normas de segurança ou desempenho para seguir e não se preocupam com segurança, pois os sistemas são intranet.

10. CAEd – Empresa ligada a UFJF que desenvolve software para escolas: contato disse que estão fazendo software para gerenciar avaliações online e pode ser que tenham que seguir normas. Exige maior investigação, depois que tiver os objetivos bem definidos.
11. NC Brasil – Empresa de telecomunicações que possui equipe de desenvolvimento em JF: contato diz que não seguem qualquer tipo de Norma
12. Bol.com – e-commerce (Holanda): possível contato com ex-aluno da UFRJ. Difícil fazer case study pela distância...
13. Tiqs - trabalho com customização do ERP JD Edwards: contato disse que não seguem nenhum tipo de norma. Só olham para os 'requisitos' do cliente, implementam e entregam.

References

Boehm. B., M.H. Penedo, E.D. Stuckle, R.D. Williams, A.B. Pyster, "A Software Development Environment for Improving Productivity," *Computer*, Vol.17, Issue 6, June 1984. Page 30-44.

T. DeMarco and T. Lister. Peopleware. Productive Projects and Teams. Dorset House Publishing, 1987.

R. E. Stake. The Art of Case Study Research, Sage, 1995

K. H. Klein and M. D. Myers. A Set of Principles for Conducting and Evaluating Interpretative Field Studies in Information Systems, MIS Quarterly, vol. 23 no. 1, pp. 67–93, 1999.

B. W., Boehm, C. Abts, A. W. Brown, S. Chulani, B. K. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece. Software Cost Estimation with COCOMO II. Prentice-Hall, 2000.

G. R. Finnie, G. E. Wittig, and D. I. Petkov. Prioritizing software development productivity factors using the analytic hierarchy process. Journal of Systems and Software, 22(2):129–139, 1993.

C.Wohlin and M. Ahlgren. Soft factors and their impact on time to market. Software Quality Journal, 4(3):189–205, 1995.

R. D. Banker, S. M. Datar, and C. F. Kemerer. A model to evaluate variables impacting the productivity of software maintenance projects. Management Science, 37(1):1–18, 1991.

Appendix F C1: Case study presentation letter

Rio de Janeiro, 22 de Novembro de 2017

Ao Sr. ----------

Assunto: Apoio para Realização de Pesquisa Cientifica na área de Teste de Requisitos de Desempenho e Segurança de Software (solicita)

Prezado Sr. -----------,

somos um time de pesquisadores da COPPE/UFRJ e SINTEF/Noruega interessados no desenvolvimento de tecnologias de apoio ao teste de segurança e desempenho de software. Nossos estudos têm ocorrido em diferentes contextos e organizações no Brasil e na Noruega.

Considerando o espírito de colaboração existente entre nossas instituições (COPPE e <<organization>> do Brasil) ao longo do tempo, vimos muito respeitosamente solicitar seu apoio no sentido de autorizar que possamos acompanhar os processos de testes de segurança e/ou desempenho de projetos de software.

O apoio solicitado não envolve qualquer tipo de investimento, gasto ou obrigação com o estudo, mas apenas autorizar que o pesquisador Victor Vidigal Ribeiro, aluno de doutorado da COPPE/UFRJ, entreviste desenvolvedores que atuem em projetos de desenvolvimento software que contemplem atividades de testes para avaliar aspectos de segurança e desempenho. Informações mais detalhadas do estudo e atividades a serem realizadas se encontram no documento anexo.

Colocamo-nos a disposição para esclarecimentos e explicações adicionais relacionados ao estudo, cujos resultados entendemos serão de utilidade para a evolução da engenharia de software, os quais teremos prazer em compartilhar.

Atenciosamente

Guilherme Horta Travassos – Professor Titular
Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ
ght@cos.ufrj.br – (21) 3938-8712

ANEXO

Este documento tem por finalidade apresentar a proposta de estudo a ser realizada, fornecer informações sobre as instituições de pesquisa envolvidas, os pesquisadores e os objetivos do estudo a ser realizado.

Organizações envolvidas

Duas instituições de pesquisa estão envolvidas no estudo: Grupo de Engenharia de Software Experimental da COPPE/UFRJ (Brasil) representado pelo Prof. Guilherme Horta Travassos e SINTEF (Noruega), representado pela pesquisadora Daniela Soares Cruzes.

O Grupo de Engenharia de Software Experimental[10] (ESE) faz parte do Programa de Engenharia e de Sistemas e Computação[11] (PESC) que, por sua vez, pertence ao Instituto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia[12] (COPPE) da Universidade Federal do Rio de Janeiro (UFRJ). A finalidade do grupo é realizar pesquisas relacionadas as tecnologias que envolvem o desenvolvimento de sistemas de software com o apoio de métodos científicos.

O grupo de engenharia experimental foi criado no ano de 2001 e, desde então, foram concluídas 16 teses de doutorado e 39 dissertações de mestrado. Atualmente, o grupo é formado por 9 doutorandos, 4 mestrandos e 2 pesquisadores cursando estágio de pós-doutorado. O coordenador do grupo, professor titular Guilherme Horta Travassos (http://lattes.cnpq.br/7541486051032916, ght@cos.ufrj.br), possui excelência reconhecida de pesquisa em engenharia de software com vários trabalhos publicados em periódicos, anais de eventos e capítulos de livros voltados, dentre outros, ao teste de software. Em particular, Victor Vidigal Ribeiro (http://lattes.cnpq.br/5208608664907557, vidigal@cos.ufrj.br), um dos doutorandos do Grupo ESE, é o principal interessado neste estudo.

O SINTEF[13] é uma organização Norueguesa de pesquisas aplicadas, tecnologia e inovação que conta com um quadro de cerca de 2000 colaboradores pesquisando em diversas áreas como energia renovável, óleo e gás, saúde e bem-estar e também desenvolvimento de sistemas de software. Nesse estudo, o SINTEF é representado pela pesquisadora doutora Daniela Soares Cruzes (http://lattes.cnpq.br/8004792005050914, daniela@sintef.no) que possui larga

[10] http://lens-ese.cos.ufrj.br/

[11] http://www.cos.ufrj.br/

[12] http://www.coppe.ufrj.br

[13] https://www.sintef.no/en/

experiência relacionada a pesquisa na área de engenharia de software na indústria com artigos publicados e orientações de mestrado concluídas.

Objetivo do estudo

O objetivo do estudo é compreender como as organizações que desenvolvem software conduzem as atividades de verificação (testes e inspeções) de segurança e desempenho. Dessa forma, é possível identificar problemas reais enfrentados pelas organizações e, assim, contribuir com resultados de pesquisas que ajudem a solucionar esses problemas.

É importante esclarecer que não é objetivo do estudo realizar algum tipo de julgamento quanto as práticas utilizadas pela organização ou sobre seus colaboradores. Além disso, não há necessidade de que o resultado das atividades de verificação, por exemplo as falhas reveladas, sejam explicitadas aos pesquisadores, mantendo assim a confidencialidade dos resultados dessas atividades.

Objeto do estudo

O objeto do estudo é representado por projetos de desenvolvimento de software que incluam no processo de desenvolvimento atividades de verificação (teste e inspeção) de segurança ou desempenho, possuam no mínimo quatro membros na equipe de desenvolvimento e estejam em andamento a pelo menos dois meses.

Fases e duração do estudo

O estudo contempla uma fase de questionários, na qual alguns membros da equipe de desenvolvimento serão solicitados a responder perguntas sobre o processo de desenvolvimento do projeto com a finalidade de caracterização.

Com o projeto caracterizado, a próxima fase contempla entrevistas semiestruturadas com os membros da equipe de desenvolvimento. Essa fase tem por finalidade aprofundar o entendimento das práticas de verificação de segurança e desempenho aplicadas ao projeto. Essas entrevistas têm duração de até 45 minutos.

Contribuições do estudo

A principal contribuição do estudo é auxiliar o entendimento e caracterização das atividades relacionadas à verificação de segurança e desempenho e, assim, evidenciar e sugerir oportunidades de melhorias no processo de desenvolvimento. Essas melhorias aumentarão a capacidade de produção de sistemas com mais Segurança e melhor Desempenho.

Adicionalmente, há uma contribuição para a sociedade, pois os resultados podem aumentar a capacidade das organizações de desenvolver sistemas de software mais seguros e que tenham melhor desempenho.

Sobre Confidencialidade

Os pesquisadores se comprometem a conduzir o estudo com ética e manter sigilo sobre quaisquer dados que possam ser utilizados para identificar a instituição ou seus colaboradores.

Considerações finais

Consideramos que pesquisa aplicada, com colaboração explícita entre academia e indústria, é um dos pilares para a evolução tecnológica de um país. Dessa forma, entendemos que os resultados deste estudo contribuem efetivamente para tornar os sistemas desenvolvidos no Brasil mais seguros e com melhor desempenho. Assim, esperamos que a proposta de estudo seja aceita para que seus resultados possam ser utilizados em tempo.

Appendix G I1: Case study Organization agreement term

Termo de consentimento organizacional

Eu, _________________________, Brasileiro, portador do RG ___________________, atuando como ___________________________________, na <<organization>>, autorizo a realização do estudo que tem o objetivo de caracterizar as atividades de verificação de segurança e desempenho no contexto de desenvolvimento de software Brasileiro, fomentando o desenvolvimento de software mais seguros e com melhor desempenho, sob condições de respeito às regras declaradas a seguir.

- Os pesquisadores comprometem a conduzir o estudo com ética e manter sigilo sobre quaisquer dados que possam ser utilizados para identificar a instituição e/ou seus colaboradores;
- O estudo não tem o objetivo de julgar ou avaliar a forma como as atividades da organização são conduzidas nem a forma como seus colaboradores executam suas tarefas;
- O estudo deve ser executado de forma a interferir minimamente nas atividades desempenhadas pelos colaboradores e sem prejudicar suas metas de produtividade;
- Nenhum dos colaboradores é obrigado a participar do estudo; e
- Qualquer colaborador pode desistir de participar do estudo a qualquer momento;

Agradecemos a compreensão e colaboração para o progresso da ciência.

____

XXXXXXXXXX

Gerente do Projeto YYYYYYYY

Rio de Janeiro, ____ de ____________ de 2017

Appendix H I2: Case study – Participant agreement term

Termo de consentimento para participantes

Eu declaro ter mais de 18 anos de idade e que concordo em participar do estudo não invasivo e impessoal conduzido pelos pesquisadores Guilherme Horta Travassos, Daniela Soares Cruzes e o doutorando Victor Vidigal Ribeiro. O estudo faz parte de pesquisas realizadas no contexto de uma tese de doutorado na Universidade Federal do Rio de Janeiro (COPPE/UFRJ).

Objetivo

O objetivo do estudo é melhorar a compreensão sobre como as atividades de verificação de segurança e desempenho são desempenhadas nas organizações de desenvolvimento de software no Brasil.

Procedimento

Os participantes serão questionados sobre as atividades de desenvolvimento de software e principalmente atividades referentes à verificação (testes e inspeções) de segurança e desempenho. As perguntas serão realizadas na forma de questionários e entrevistas. O estudo contempla também uma fase de observação, na qual as atividades executadas pelos participantes do estudo serão observadas pelo pesquisador.

Confidencialidade

Toda informação coletada neste estudo é confidencial. Dessa forma, o nome do participante não será divulgado em momento algum. Da mesma forma, o participante se compromete a manter sigilo das técnicas e documentos apresentados durante o estudo.

Benefícios e liberdade de desistência

Os benefícios gerados pelo estudo estão relacionados com a melhoria das atividades de verificação de segurança e desempenho. Dessa forma, podendo melhorar a qualidade dos sistemas de software produzidos no Brasil. Além disso, o estudo permite revisar o processo de desenvolvimento utilizado pela organização podendo fornecer subsídios para seu aprimoramento e melhor compreensão.

Eu entendo que sou livre para realizar perguntas a qualquer momento ou solicitar que qualquer informação relacionada a minha pessoa não seja incluída no estudo.

Eu entendo que o estudo não tem a finalidade de avaliação pessoal do participante. Assim, entendo que minha participação não afetará, de forma alguma, minha posição dentro da organização e que não serei julgado por minhas respostas.

Dessa forma, declaro que estou de acordo com os termos anteriores e que participo de livre e espontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento de tecnologias para o desenvolvimento de software.

Nome Completo em letra de forma: ____

Assinatura ____

Data _____/_____/__________

Appendix I   I3: Case study – Organization characterization

Caracterização da organização

ID Organização _____

1. Tipo de indústria/domínio em que os sistemas desenvolvidos são utilizados
   - A. Bancário
   - B. Educação
   - C. Telecomunicações
   - D. Bens de consumo
   - E. Viagens
   - F. Outro ______________________________
2. Natureza da organização
   - A. Pública
   - B. Privada
   - C. Outro ______________________________
3. Número de colaboradores ________
4. Número de colaboradores de TI ________
5. A organização possui algum tipo de certificação (CMMI, ISO, MPS.Br)?
   - A. Sim. Qual(is)? _________________________________
   - B. Não
6. Informação de contexto relevante

   Espaço libre para descrição de informações de contexto que possam ser importantes

____

____

____

____

____

____

____

Appendix J I4: Case study – Project characterization

Caracterização do projeto, equipe, processo de desenvolvimento e ambiente

ID Projeto _____

Informações gerais do projeto

1. Descrição resumida do projeto

____

____

____

2. Domínio do projeto
   A. Bancário
   B. Educação
   C. Telecomunicação
   D. Bens de consumo
   E. Outro ____________________________

Informações gerais do ambiente

3. Tamanho da equipe _________

4. Tipo de aplicação
   A. Web
   B. Mobile
   C. Desktop
   D. Embedded
   E. Outro ____________________________

5. Linguagens de programação utilizadas
   Sugestões: Java, Javascript, Python, Html, Ruby

____

____

____

6. Ferramentas e frameworks utilizados
   Sugestões: IDE, bug tracking, ferramentas de testes, frameworks de persistência

____

____

____

7. Qual a principal prática ágil utilizada?
   Sugestões: Scrum, XP, Hubrid, Scrumban

Outras práticas utilizadas

A. Scrum
B. XP
C. Scrumban
D. Code and tests
E. Continuous integration
F. Daily deployment
G. Daily meeting
H. Pair programming
I. TDD
J. Root cause analysis
K. Stories
L. Outras: ______________________________

Informações sobre processo de desenvolvimento

8. Descrição da fase de requisitos

Sugestões:

- Descrição de como os requisitos são identificados: entrevistas com clientes, brainstorm, normas.

- Descrição de como os requisitos são representados: especificação formal, histórias

- Ferramentas utilizadas para apoiar a fase de requisitos: editores de texto, ferramentas específicas para requitios, DotProject, OpenReq.

____
____
____
____
____
____
____
____

9. Descrição da fase de projeto

Sugestões:

- Descrição de como o projeto do sistema é representado: diagramas da UML, DTR.

- Ferramentas utilizadas para apoiar a fase de projetos: StarUML, AstahUML, Enterprise architect.

____
____
____
____
____

10. Descrição da fase de implementação

Sugestões:

- Descrição do ciclo de vida de uma tarefa, iniciando de como ela é distribuída aos programadores até o momento em que é considerada como concluída.

11. Descrição da fase de V&V

Sugestões:

- Quem é o principal responsável pelas atividades de V&V?

- Quando as atividades de V&V são iniciadas?

- Quem executa as atividades de verificação? Apenas testadores? Desenvolvedores também executam?

- Equipe faz uso de TDD?

12. Descrição da fase de entrega

Sugestões:

Descrever o processo que vai do empacotamento do sistema até a disponibilização para os usuários finais

Appendix K I5: Case study – Participant characterization

Caracterização dos participantes da equipe de desenvolvimento

ID Participante _____

ID Projeto _____

1. Nível de formação
   A. Tecnólogo
   B. Graduação
   C. Especialização
   D. Mestrado
   E. Doutorado
   F. Outro ____________________

2. Experiência com desenvolvimento de software __________________________

3. Experiência com Atividades de Testes ________________________

4. Experiência com Atividades de Testes de Segurança ou Desempenho ________________

5. Número de projetos de desenvolvimento de software que participou ____________

6. Função dentro da equipe de desenvolvimento atual ______________________________________

   Sugestões: programador, analista, tester, PO

Appendix L I6: Case study – Verification practices identification

Identificação das práticas de verificação de segurança e desempenho

ID Projeto _____

1. Quais as práticas são utilizadas para realizar verificação de segurança e desempenho?
   A. Prática _________________________
   B. Responsável _________________________
   C. Técnica _________________________
   D. Artefato alvo _________________________
   E. Ferramentas _________________________
   F. Quando _________________________
   G. Descrição

____
____
____

Sugestões:

Quais as práticas são utilizadas para realizar verificação de segurança e desempenho?

A. Prática: inspeção de pares para segurança
B. Responsável: committer
C. Técnica: Experiência do inspetor
D. Artefato alvo: Código fonte
E. Ferramentas: nenhuma
F. Quando: a cada sprint
G. Descrição: Nós revisamos o código fonte procurando por falhas de segurança que conhecemos previamente a partir de projetos anteriores

Quais as práticas são utilizadas para realizar verificação de segurança e desempenho?

A. Prática: Teste de unidade
B. Responsável: Product owner
C. Técnica: Técnica OWASP
D. Artefato alvo: Código fonte
E. Ferramentas: JUnit
F. Quando: antes de entregas que consideramos importantes ou críticas
G. Descrição: construímos casos de testes com a técnica OWASP e executamos esses casos de testes antes de releases que consideramos importantes ou críticas. O sistema só é disponibilizado ao usuário depois que os casos de testes passam sem nenhuma falha.

Appendix M I7: Case study – Identification of decision-making factors

Identificação dos fatores de decisão

ID Projeto _____

Informações gerais

1. Quando é decidido que determinada prática de verificação deve ser utilizada?
   Por exemplo, quando foi decidido que testes de unidade deveriam ser utilizados?

   Sugestões: imposto pela organização, no início de cada projeto, gerente de projetos decide, decidimos a cada sprint

____

____

2. Com que frequência as práticas de verificação são reconsideradas?
   Por exemplo, decidir que uma prática não deve mais ser utilizada ou incluir uma nova prática

   Sugestões: a cada sprint, cada entrega, diante do número de defeitos encontrados, feedback de usuários

____

____

3. Que fatores influenciam na decisão de utilizar determinada técnica?
   Sugestões: criticidade do sistema, tempo de entrega, orçamento, experiência da equipe, conhecimento da técnica pelo gerente de projetos

____

____

4. Quem é o responsável pela tomada de decisões em relação à escolha das práticas de verificação?
   Sugestões: *product owner*, gerente, gerente de teste

____

____

Informações sobre prática {prática_específica}

5. Quais os fatores influenciaram especificamente na escolha dessa prática?
   Sugestões: porque essa prática foi escolhida ou porque foi escolhido essa ferramenta para execução da prática.

____

____

Informações complementares

6. Observações sobre os fatores de decisão?

Appendix N I8: Case study – Participant opinion

Identificação da opinião dos participantes sobre as atividades de verificação de segurança e desempenho

ID Participante _____

ID Projeto _____

1. Se existissem duas ou três práticas que você pudesse mudar para melhorar a segurança ou desempenho dos sistemas produzidos, qual você mudaria? Por quê?

   Sugestões:

   - Incluiria/removeria alguma prática

   - Mudar o momento em que alguma prática é executada

   - Alterar a ferramenta utilizada

____

____

____

2. Qual a sua opinião sobre a segurança e desempenho dos projetos que você participou?

   Sugestões:

   - Você acha que a segurança do sistema desenvolvido é apropriada?

   - Você acha que o desempenho (ex.: tempo de resposta, consumo de memória) do sistema desenvolvido é adequado?

____

____

____

3. Na sua opinião, quais fatores deveriam ser considerados para tomar decisões relacionadas às práticas de verificação utilizadas no projeto? Por quê?

   Sugestões: por exemplo, alguma coisa que você entende que o gerente deve considerar para adotar ou remover alguma prática de verificação do processo de desenvolvimento.

   - Orçamento do projeto

   - Prazo para entrega

   - Criticidade do sistema

   - Falta de treinamento da equipe

____

____

____

Appendix O RR01 – Suitable environment protocol

Rapid Reviews Protocol:

Software Security and Performance **Awareness**

Introduction

Authors

Victor Vidigal Ribeiro – vidigal@cos.ufrj.br
Guilherme Horta Travassos – ght@cos.ufrj.br
Daniela Soares Cruzes – daniela.s.cruzes@sintef.no

Context

During the analysis of a case study research related to security and performance verification, we could make observations that led us to build some conjectures[14] about this subject. One of them regards the awareness of security and performance verification importance. Based on our observations, we could hypostatize that

"Awareness of the importance of software system Security and Performance contributes to verification."

Therefore, this Rapid Review (RR) aims to verify the existence of published studies supporting our conjecture or studies proposing solutions to support system software security and performance awareness.

Research Questions

- **RQ1:** What are the **benefits** of awareness of the importance of software security and performance?
- **RQ2:** What are the **problems** caused by the lack of awareness of the importance of software security and performance?
- **RQ3:** What are the **challenges** to improving awareness of the importance of software security and performance?
- **RQ4:** What are the **strategies** to support awareness improvement of the importance of security and performance?

Search Strategy

The Scopus[15] search engine and the following search string support this RR:

[14] https://www.merriam-webster.com/dictionary/conjectures

[15] https://www.scopus.com

```
TITLE-ABS-KEY ( ( "security verification" OR
"performance verification" OR "security testing" OR
"performance testing" ) AND ( awareness OR
recognition OR understanding OR comprehension OR
importance OR relevance ) AND ( "software" ) AND (
"benefit*" OR "problem*" OR "challenge*" OR
"strateg*" OR "empirical stud*" OR "experimental
stud*" OR "experiment*" OR "case stud*" OR
"survey*" ) )
```

Selection procedure

The following selection procedure is performed by one researcher:

Run the search string;
Apply the inclusion criteria based on the paper Title;
Apply the inclusion criteria based on the paper Abstract;
Apply the inclusion criteria based on the paper Full Text;

Inclusion criteria

The paper must be in the context of **software engineering**; and
The paper must be in the context of **software system performance or security**; and
The paper must provide data to **answer** at least one of the RR **research questions**.
The paper must be written in the **English language**.

Extraction procedure

The extraction procedure is performed by one researcher, using the form presented in section **0**

Extraction form

<paper_id> <paper_reference>	
Description	<A brief description of the study objectives>
Study type	
Benefits	
Problems	
Challenges	
Strategies	

Synthesis Procedure

In this RR, the extraction form provides a synthesized way to represent extracted data. Thus, we do not perform any synthesis procedure.

However, the synthesis is usually performed through a narrative summary or a Thematic Analysis when the number of selected papers is not high Erro! Fonte de referência não encontrada..

References

C. Tricco et al. A scoping review of rapid review methods. BMC Medicine, 2015.

B. Cartaxo et al. Evidence briefings: Towards a medium to transfer knowledge from systematic reviews to practitioners. ESEM, 2016.

Appendix P RR02 – Cross-functional team protocol

Rapid Reviews Protocol:

Software Security and Performance

Multidisciplinary Team

Introduction

Authors

Victor Vidigal Ribeiro – vidigal@cos.ufrj.br
Guilherme Horta Travassos – ght@cos.ufrj.br
Daniela Soares Cruzes – daniela.s.cruzes@sintef.no

Context

During the analysis of a case study research related to security and performance verification, we could make observations that led us to build some conjectures[16] about this subject. One of them regards the security and performance verification environment. Based on our observations, we could hypostatize that

"Software security and performance verification need a multidisciplinary team."

Therefore, this Rapid Review (RR) aims to verify the existence of published studies supporting our conjecture or studies proposing solutions to improve the disciplinary of the security and performance verification team.

Research Questions

- **RQ1:** What are the **benefits** of a multidisciplinary team acting in the verification of security and performance?
- **RQ2:** What **problems** do cause the lack of multidisciplinary team acting in the verification of security and performance?
- **RQ3:** What are the **challenges** to have a multidisciplinary team acting in the verification of security and performance?
- **RQ4:** What are the **strategies** to have a multidisciplinary team working on verification of security and performance?

Search Strategy

[16] https://www.merriam-webster.com/dictionary/conjectures

The Scopus[17] search engine and the following search string support this RR:

```
TITLE-ABS-KEY ( ( "security verification" OR
"performance verification" OR "security testing" OR
"performance testing" ) AND ( team* OR staff* OR
"Working Group" ) AND ( "software" ) AND ( "benefit"
OR "problem" OR "challenge" OR "strategy" OR
"empirical study" OR "experimental study" OR "formal
experiment" OR "experiment" OR "case study" OR
survey ) )
```

Selection procedure

The following selection procedure is performed by one researcher:

Run the search string;
Apply the inclusion criteria based on the paper Title;
Apply the inclusion criteria based on the paper Abstract;
Apply the inclusion criteria based on the paper Full Text;

Inclusion criteria

The paper must be in the context of **software engineering**; and
The paper must be in the context of **performance and/or security verification**; and
The paper must report a **study related to the verification team**; and
The paper must report a **primary study**; and
The paper must report an **evidence-based study** grounded in empirical methods (*e.g.*, interviews, surveys, case studies, formal experiment, among others); and
The paper must provide data to **answer** at least one of the RR **research questions**.
The paper must be written in the **English language**.

Extraction procedure

The extraction procedure is performed by one researcher, using the form presented in section **0**

Extraction form

<paper_id>:<paper_reference>	
Description	<A brief description of the study objectives>
Study type	<Identify the type of study reported by paper (*e.g.*, survey, formal experiment)>
Benefits	- <beneft_1> - <benefit_2> - ...
Problems	- <problem_1> - <problem_2> - ...
Challenges	- <challenge_1>

[17] https://www.scopus.com

	- <challenge_2> - …
Strategies	- <strategy_1> - <strategy_2> - …

Synthesis Procedure

In this RR, the extraction form provides a synthesized way to represent extracted data. Thus, we do not perform any synthesis procedure.

However, the synthesis is usually performed through a narrative summary or a Thematic Analysis when the number of selected papers is not high Erro! Fonte de referência não encontrada..

References

C. Tricco et al. A scoping review of rapid review methods. BMC Medicine, 2015.

B. Cartaxo et al. Evidence briefings: Towards a medium to transfer knowledge from systematic reviews to practitioners. ESEM, 2016.

Appendix Q RR03 – Suitable requirements protocol

Rapid Reviews Protocol:

Software Security and Performance Requirements

Introduction

Authors

Victor Vidigal Ribeiro – vidigal@cos.ufrj.br
Guilherme Horta Travassos – ght@cos.ufrj.br
Daniela Soares Cruzes – daniela.s.cruzes@sintef.no

Context

During the analysis of a case study research related to security and performance verification, we could make observations that led us to build some conjectures[18] about this subject. One of them regards the security and performance verification requirements. Based on our observations, we could hypostatize that *"Greater precision on the requirements definition contributes to verification."* Therefore, this Rapid Review (RR) aims to verify the existence of published studies supporting our conjecture or studies proposing solutions to support the definition of security and performance requirements.

Research Questions

- **RQ1:** What are the **benefits** of a requirements precise definition for the verification of security and performance?
- **RQ2:** What are the **problems** of an imprecise definition requirement for verification of security and performance?
- **RQ3:** What are the **challenges** on the precise definition of the requirements for the verification of security and performance?
- **RQ4:** What are the **strategies** to support a precise definition of the verification of security and performance requirements?

Search Strategy

The Scopus[19] search engine and the following search string support this RR:

```
TITLE-ABS-KEY ( ( "security verification" OR
"performance verification" OR "security testing" OR
"performance testing" ) AND ( requirement* ) AND (
```

[18] https://www.merriam-webster.com/dictionary/conjectures

[19] https://www.scopus.com

```
"software" )   AND   ( "benefit*"   OR   "problem*"   OR
"challenge*"  OR  "strateg*"  OR  "empirical stud*"  OR
"experimental  stud*"   OR   "experiment*"   OR   "case
stud*"  OR  "survey*" ) )
```

Selection procedure

The following selection procedure is performed by one researcher:

Run the search string;
Apply the inclusion criteria based on the paper Title;
Apply the inclusion criteria based on the paper Abstract;
Apply the inclusion criteria based on the paper Full Text;

Inclusion criteria

The paper must be in the context of **software engineering**; and
The paper must be in the context of **performance or security verification**; and
The paper must report a **security or performance verification requirement related study**; and
The paper must report a **primary study**; and
The paper must report an **evidence-based study** grounded in empirical methods (*e.g.*, interviews, surveys, case studies, formal experiment, among others); and
The paper must provide data to **answer** at least one of the RR **research questions**.
The paper must be written in the **English language**.

Extraction procedure

The extraction procedure is performed by one researcher, using the form presented in section **0**

Extraction form

<paper_id> <paper_reference>	
Description	<A brief description of the study objectives>
Study type	
Benefits	
Problems	
Challenges	
Strategies	

Synthesis Procedure

In this RR, the extraction form provides a synthesized way to represent extracted data. Thus, we do not perform any synthesis procedure. However, the synthesis is usually performed through a narrative summary or a Thematic Analysis when the number of selected papers is not high Erro! Fonte de referência não encontrada..

References

C. Tricco et al. A scoping review of rapid review methods. BMC Medicine, 2015.

B. Cartaxo et al. Evidence briefings: Towards a medium to transfer knowledge from systematic reviews to practitioners. ESEM, 2016.

Appendix R RR04 – Support tools protocol

Rapid Reviews Protocol:

Software Security and Performance **suitable support tools**

Introduction

Authors

Victor Vidigal Ribeiro – vidigal@cos.ufrj.br
Guilherme Horta Travassos – ght@cos.ufrj.br
Daniela Soares Cruzes – daniela.s.cruzes@sintef.no

Context

During the analysis of a case study research related to security and performance verification, we could make observations that led us to build some conjectures[20] about this subject. One of them regards the security and performance verification support tools. Based on our observations, we could hypostatize that

"A suitable support tool contributes to the verification of security and performance."

Therefore, this Rapid Review (RR) aims to verify the existence of published studies supporting our conjecture or studies proposing solutions to improve the security and performance verification support tools.

Research Questions

- **RQ1:** What are the **benefits** of a suitable support tool for the verification of security and performance?
- **RQ2:** What **problems** do cause an unsuitable support tool for the verification of security and performance?
- **RQ3:** What are the **challenges** to create a suitable support tool for the verification of security and performance?
- **RQ4:** What are the **strategies** to create a suitable support tool for the verification of security and performance?

Search Strategy

The Scopus[21] search engine and the following search string support this RR:

[20] https://www.merriam-webster.com/dictionary/conjectures

[21] https://www.scopus.com

```
TITLE-ABS-KEY ( ( "security verification" OR
"performance verification" OR "security testing" OR
"performance testing" ) AND ("support tool" ) AND (
"software" ) AND ( "benefit" OR "problem" OR
"challenge" OR "strategy" OR "empirical study" OR
"experimental study" OR "formal experiment" OR
"experiment" OR "case study" OR survey ) )
```

Selection procedure

The following selection procedure is performed by one researcher:

Run the search string;
Apply the inclusion criteria based on the paper Title;
Apply the inclusion criteria based on the paper Abstract;
Apply the inclusion criteria based on the paper Full Text;

Inclusion criteria

The paper must be in the context of **software engineering**; and
The paper must be in the context of **performance and/or security verification**; and
The paper must report **verification support tool-related study**; and
The paper must report a **primary study**; and
The paper must report an **evidence-based study** grounded in empirical methods (*e.g.*, interviews, surveys, case studies, formal experiment, among others); and
The paper must provide data to **answer** at least one of the RR **research questions**.
The paper must be written in the **English language**.

Extraction procedure

The extraction procedure is performed by one researcher, using the form presented in section **0**

Extraction form

<paper_id>:<paper_reference>	
Description	<A brief description of the study objectives>
Study type	<Identify the type of study reported by paper (*e.g.*, survey, formal experiment)>
Benefits	<Set of **benefits** brought by a suitable support tool of security and performance verification> - <beneft_1> - <benefit_2> - ...
Problems	<Set of **problems** brought by an unsuitable security and performance verification support tool> - <problem_1> - <problem_2> - …
Challenges	<Set of **challenges** in providing a suitable support tool to security and performance verification>

	- <challenge_1> - <challenge_2> - …
Strategies	<Set of **strategies** that can be used to provide a suitable support tool for security and performance verification> -

Synthesis Procedure

In this RR, the extraction form provides a synthesized way to represent extracted data. Thus, we do not perform any synthesis procedure.

However, the synthesis is usually performed through a narrative summary or a Thematic Analysis when the number of selected papers is not high Erro! Fonte de referência não encontrada..

References

C. Tricco et al. A scoping review of rapid review methods. BMC Medicine, 2015.

B. Cartaxo et al. Evidence briefings: Towards a medium to transfer knowledge from systematic reviews to practitioners. ESEM, 2016.

Appendix S RR05 – Suitable environment protocol

Rapid Reviews Protocol:

Software Security and Performance **Environment**

Introduction

Authors

Victor Vidigal Ribeiro – vidigal@cos.ufrj.br
Guilherme Horta Travassos – ght@cos.ufrj.br
Daniela Soares Cruzes – daniela.s.cruzes@sintef.no

Context

During the analysis of a case study research related to security and performance verification, we could make observations that led us to build some conjectures[22] about this subject. One of them regards the security and performance verification environment. Based on our observations, we could hypostatize that

"A suitable environment contributes to the verification of security and performance."

Therefore, this Rapid Review (RR) aims to verify the existence of published studies supporting our conjecture or studies proposing solutions to improve the security and performance verification environments.

Research Questions

- **RQ1:** What are the **benefits** of a suitable environment for the verification of security and performance?
- **RQ2:** What **problems** do cause an unsuitable environment for the verification of security and performance?
- **RQ3:** What are the **challenges** to create a suitable environment for the verification of security and performance?
- **RQ4:** What are the **strategies** to create a suitable environment for the verification of security and performance?

Search Strategy

The Scopus[23] search engine and the following search string support this RR:

[22] https://www.merriam-webster.com/dictionary/conjectures

[23] https://www.scopus.com

```
TITLE-ABS-KEY ( ( "security verification" OR
"performance verification" OR "security testing" OR
"performance testing" ) AND ( environment ) AND (
"software" ) AND ( "benefit" OR "problem" OR
"challenge" OR "strategy" OR "empirical study" OR
"experimental study" OR "formal experiment" OR
"experiment" OR "case study" OR survey ) )
```

Selection procedure

The following selection procedure is performed by one researcher:

Run the search string;
Apply the inclusion criteria based on the paper Title;
Apply the inclusion criteria based on the paper Abstract;
Apply the inclusion criteria based on the paper Full Text;

Inclusion criteria

The paper must be in the context of **software engineering**; and
The paper must be in the context of **performance and/or security verification**; and
The paper must report a **verification environment related study**; and
The paper must report a **primary study**; and
The paper must report an **evidence-based study** grounded in empirical methods (*e.g.*, interviews, surveys, case studies, formal experiment, among others); and
The paper must provide data to **answer** at least one of the RR **research questions**.
The paper must be written in the **English language**.

Extraction procedure

The extraction procedure is performed by one researcher, using the form presented in section **0**

Extraction form

<paper_id>:<paper_reference>	
Description	<A brief description of the study objectives>
Study type	<Identify the type of study reported by paper (*e.g.*, survey, formal experiment)>
Benefits	<Set of **benefits** brought by a suitable environment of security and performance verification> - <beneft_1> - <benefit_2> - ...
Problems	<Set of **problems** brought by an unsuitable security and performance verification environment> - <problem_1> - <problem_2> - …
Challenges	<Set of **challenges** in providing a suitable environment to security and performance verification>

	- <challenge_1> - <challenge_2> - …
Strategies	<Set of **strategies** that can be used to provide a suitable environment for security and performance verification> -

Synthesis Procedure

In this RR, the extraction form provides a synthesized way to represent extracted data. Thus, we do not perform any synthesis procedure.

However, the synthesis is usually performed through a narrative summary or a Thematic Analysis when the number of selected papers is not high Erro! Fonte de referência não encontrada..

References

C. Tricco et al. A scoping review of rapid review methods. BMC Medicine, 2015.

B. Cartaxo et al. Evidence briefings: Towards a medium to transfer knowledge from systematic reviews to practitioners. ESEM, 2016.

Appendix T RR06 – Suitable methodology protocol

Rapid Reviews Protocol:

Software Security and Performance Verification **Methodology**

Introduction

Authors

Victor Vidigal Ribeiro – vidigal@cos.ufrj.br
Guilherme Horta Travassos – ght@cos.ufrj.br
Daniela Soares Cruzes – daniela.s.cruzes@sintef.no

Context

During the analysis of a case study research related to security and performance verification, we could make observations that led us to build some conjectures[24] about this subject. One of them regards the security and performance verification methodology. Based on our observations, we could hypostatize that

"A suitable methodology contributes to the verification of security and performance."

Therefore, this Rapid Review (RR) aims to verify the existence of published studies supporting our conjecture or studies proposing solutions to improve the security and performance verification methodology.

Research Questions

- **RQ1:** What are the **benefits** of a suitable methodology for the verification of security and performance?
- **RQ2:** What **problems** do cause an unsuitable methodology for the verification of security and performance?
- **RQ3:** What are the **challenges** to create a suitable methodology for the verification of security and performance?
- **RQ4:** What are the **strategies** to create a suitable methodology for the verification of security and performance?

Search Strategy

[24] https://www.merriam-webster.com/dictionary/conjectures

The Scopus[25] search engine and the following search string support this RR:

```
TITLE-ABS-KEY ( ( "security verification" OR
"performance verification" OR "security testing" OR
"performance testing" ) AND ("methodolog*" ) AND (
"software" ) AND ( "benefit" OR "problem" OR
"challenge" OR "strategy" OR "empirical study" OR
"experimental study" OR "formal experiment" OR
"experiment" OR "case study" OR survey ) )
```

Selection procedure

The following selection procedure is performed by one researcher:

Run the search string;
Apply the inclusion criteria based on the paper Title;
Apply the inclusion criteria based on the paper Abstract;
Apply the inclusion criteria based on the paper Full Text;

Inclusion criteria

The paper must be in the context of **software engineering**; and
The paper must be in the context of **performance and/or security verification**; and
The paper must report findings regarding a **verification methodology**; and
The paper must report a **primary study**; and
The paper must report an **evidence-based study** grounded in empirical methods (*e.g.*, interviews, surveys, case studies, formal experiment, among others); and
The paper must provide data to **answer** at least one of the RR **research questions**.
The paper must be writing in the **English language**.

Extraction procedure

The extraction procedure is performed by one researcher, using the form presented in section **0**

Extraction form

<paper_id>:<paper_reference>	
Description	<A brief description of the study objectives>
Study type	<Identify the type of study reported by paper (*e.g.*, survey, formal experiment)>
Benefits	<Set of **benefits** brought by a suitable methodology of security and performance verification> - <beneft_1> - <benefit_2> - ...
Problems	<Set of **problems** brought by an unsuitable security and

[25] https://www.scopus.com

	performance verification methodology> - <problem_1> - <problem_2> - …
Challenges	<Set of **challenges** in providing a suitable methodology to security and performance verification> - <challenge_1> - <challenge_2> - …
Strategies	<Set of **strategies** that can be used to provide a suitable methodology for security and performance verification> -

Synthesis Procedure

In this RR, the extraction form provides a synthesized way to represent extracted data. Thus, we do not perform any synthesis procedure.

However, the synthesis is usually performed through a narrative summary or a Thematic Analysis when the number of selected papers is not high Erro! Fonte de referência não encontrada..

References

C. Tricco et al. A scoping review of rapid review methods. BMC Medicine, 2015.

B. Cartaxo et al. Evidence briefings: Towards a medium to transfer knowledge from systematic reviews to practitioners. ESEM, 2016.

Appendix U RR07 – Verification planning protocol

Rapid Reviews Protocol:

Software Security and Performance Verification **Planning**

Introduction

Authors

Victor Vidigal Ribeiro – vidigal@cos.ufrj.br
Guilherme Horta Travassos – ght@cos.ufrj.br
Daniela Soares Cruzes – daniela.s.cruzes@sintef.no

Context

During the analysis of a case study research related to security and performance verification, we could make observations that led us to build some conjectures[26] about this subject. One of them regards the security and performance verification Planning. Based on our observations, we could hypostatize that

"A suitable planning contributes to the verification of security and performance."

Therefore, this Rapid Review (RR) aims to verify the existence of published studies supporting our conjecture or studies proposing solutions to improve the security and performance verification planning.

Research Questions

- **RQ1:** What are the **benefits** of proper planning for the verification of security and performance?
- **RQ2:** What **problems** do cause unsuitable planning for the verification of security and performance?
- **RQ3:** What are the **challenges** to create proper planning for the verification of security and performance?
- **RQ4:** What are the **strategies** to create proper planning for the verification of security and performance?

Search Strategy

[26] https://www.merriam-webster.com/dictionary/conjectures

The Scopus[27] search engine and the following search string support this RR:

```
TITLE-ABS-KEY ( ( "security verification" OR
"performance verification" OR "security testing" OR
"performance testing" ) AND (planning OR plan) AND (
"software" ) AND ( "benefit" OR "problem" OR
"challenge" OR "strategy" OR "empirical study" OR
"experimental study" OR "formal experiment" OR
"experiment" OR "case study" OR survey ) )
```

Selection procedure

The following selection procedure is performed by one researcher:

Run the search string;
Apply the inclusion criteria based on the paper Title;
Apply the inclusion criteria based on the paper Abstract;
Apply the inclusion criteria based on the paper Full Text;

Inclusion criteria

The paper must be in the context of **software engineering**; and
The paper must be in the context of **performance and/or security verification**; and
The paper must report findings regarding **verification planning**; and
The paper must report a **primary study**; and
The paper must report an **evidence-based study** grounded in empirical methods (*e.g.*, interviews, surveys, case studies, formal experiment, among others); and
The paper must provide data to **answer** at least one of the RR **research questions**.
The paper must be writing in the **English language**.

Extraction procedure

The extraction procedure is performed by one researcher, using the form presented in section **0**

Extraction form

<paper_id>:<paper_reference>	
Description	<A brief description of the study objectives>
Study type	<Identify the type of study reported by paper (*e.g.*, survey, formal experiment)>
Benefits	<Set of **benefits** brought by a suitable planning of security and performance verification> - <beneft_1> - <benefit_2> - ...
Problems	<Set of **problems** brought by an unsuitable security and

[27] https://www.scopus.com

	performance verification planning> - <problem_1> - <problem_2> - …
Challenges	<Set of **challenges** in providing a suitable planning to security and performance verification> - <challenge_1> - <challenge_2> - …
Strategies	<Set of **strategies** that can be used to provide suitable planning for security and performance verification> -

Synthesis Procedure

In this RR, the extraction form provides a synthesized way to represent extracted data. Thus, we do not perform any synthesis procedure.

However, the synthesis is usually performed through a narrative summary or a Thematic Analysis when the number of selected papers is not high Erro! Fonte de referência não encontrada..

References

C. Tricco et al. A scoping review of rapid review methods. BMC Medicine, 2015.

B. Cartaxo et al. Evidence briefings: Towards a medium to transfer knowledge from systematic reviews to practitioners. ESEM, 2016.

Appendix V RR08 – Reuse protocol

Rapid Reviews Protocol:
Software Security and Performance Verification
Reuse

Introduction

Authors

Victor Vidigal Ribeiro – vidigal@cos.ufrj.br
Guilherme Horta Travassos – ght@cos.ufrj.br
Daniela Soares Cruzes – daniela.s.cruzes@sintef.no

Context

During the analysis of a case study research related to security and performance verification, we could make observations that led us to build some conjectures[28] about this subject. One of them regards the security and performance verification reuse. Based on our observations, we could hypostatize that

"The reuse of artifacts and knowledge contributes to the verification of security and performance."

Therefore, this Rapid Review (RR) aims to verify the existence of published studies supporting our conjecture or studies proposing solutions to improve the security and performance verification reuse.

Research Questions

- **RQ1:** What are the **benefits** of suitable reuse for the verification of security and performance?
- **RQ2:** What **problems** do cause unsuitable reuse for the verification of security and performance?
- **RQ3:** What are the **challenges** to create suitable reuse for the verification of security and performance?
- **RQ4:** What are the **strategies** to create suitable reuse for the verification of security and performance?

Search Strategy

The Scopus[29] search engine and the following search string support this RR:

```
TITLE-ABS-KEY  (  (  "security  verification"   OR
```

[28] https://www.merriam-webster.com/dictionary/conjectures

[29] https://www.scopus.com

```
"performance verification" OR "security testing" OR
"performance testing" ) AND (reuse OR reusability
OR reusing) AND ( "software" ) AND ( "benefit" OR
"problem" OR "challenge" OR "strategy" OR
"empirical study" OR "experimental study" OR "formal
experiment" OR "experiment" OR "case study" OR
survey ) )
```

Selection procedure

The following selection procedure is performed by one researcher:

Run the search string;
Apply the inclusion criteria based on the paper Title;
Apply the inclusion criteria based on the paper Abstract;
Apply the inclusion criteria based on the paper Full Text;

Inclusion criteria

The paper must be in the context of **software engineering**; and
The paper must be in the context of **performance and/or security verification**; and
The paper must report findings regarding **verification reuse practices**; and
The paper must report a **primary study**; and
The paper must report an **evidence-based study** grounded in empirical methods (*e.g.*, interviews, surveys, case studies, formal experiment, among others); and
The paper must provide data to **answer** at least one of the RR **research questions**.
The paper must be writing in the **English language**.

Extraction procedure

The extraction procedure is performed by one researcher, using the form presented in section **0**

Extraction form

<paper_id>:<paper_reference>	
Description	<A brief description of the study objectives>
Study type	<Identify the type of study reported by paper (*e.g.*, survey, formal experiment)>
Benefits	<Set of **benefits** brought by a suitable reuse of security and performance verification> - <beneft_1> - <benefit_2> - ...
Problems	<Set of **problems** brought by a reuse security and performance verification reuse> - <problem_1> - <problem_2> - ...
Challenges	<Set of **challenges** in providing a suitable reuse to security and performance verification>

	- <challenge_1> - <challenge_2> - …
Strategies	<Set of **strategies** that can be used to provide suitable reuse for security and performance verification> -

Synthesis Procedure

In this RR, the extraction form provides a synthesized way to represent extracted data. Thus, we do not perform any synthesis procedure.

However, the synthesis is usually performed through a narrative summary or a Thematic Analysis when the number of selected papers is not high Erro! Fonte de referência não encontrada..

References

C. Tricco et al. A scoping review of rapid review methods. BMC Medicine, 2015.

B. Cartaxo et al. Evidence briefings: Towards a medium to transfer knowledge from systematic reviews to practitioners. ESEM, 2016.

Appendix W Survey plan

Identification

- Title: Assessing moderator factors of software security and performance verification in the Brazilian software Industry
- Theme: security and performance verification
- Technical area: Software engineering - Software verification – security and performance verification
- Authors:
  - Victor Vidigal Ribeiro (COPPE/UFRJ)
  - Daniela Soares Cruzes (SINTEF)
  - Guilherme Horta Travassos (COPPE/UFRJ)
- Date plan: 2019 May

Introduction

After a case study research, it was possible to identify a set of eight moderator factors influencing the verification of software security and performance and actions to promote these moderators. In sequence, these moderator factors and actions were confirmed by a set of literature reviews.

This study intends to evaluate the pertinence[30] of pre-identified moderator factors and the actions used to promote these moderators according to software practitioners' perception. Additionally, it pretends to identify new moderator factors and actions that can be applied to promote moderator factors.

Characterization

- Type: Descriptive [Linåker, Johan; Sulaman, Sardar Muhammad; Maiani de Mello, Rafael; Höst 2015]
- Domain: Software developers professionals
- Language: Portuguese and English
- Execution expectancy: 2019 June and July

Definition of the experimental study

Global Objective

Assess moderator factors influencing the verification of security and performance as well as actions that can be employed to promote such moderator factors.

Specific Objective 1

Analyze	moderator factors of security and performance verification

[30] "*having a clear decisive relevance to the matter in hand*" https://www.merriam-webster.com/dictionary/pertinent

with the purpose of	characterize
with respect to	their pertinence regarding security and performance verification
from the point of view of	software practitioners
in the context of	software development organizations

Specific Objective 2

Analyze	actions to promote the moderator factors of security and performance verification
with the purpose of	Characterize
with respect to	their pertinence regarding the ability to promote the moderator factors
from the point of view of	software practitioners
in the context of	software development organizations

General research questions

The following research questions were formulated to reach the study objectives:

- RQ 1 What is the pertinence of the identified moderator factors?
  - What is the pertinence of the promoting actions identified to each moderator factor?
- RQ 2 What are the moderator factors influencing the verification of security and performance?
  - RQ 2.1 What are the actions that could be done to promote the new and identified moderator?

Subjects selection

- Target audience: software development practitioners
- Population: Brazilian software organizations
- Sampling technique: (see section) by convenience, using contacts from researchers involved in the study, and executing the survey in practitioners' conferences.
- Unit of observation: Software practitioner
- Unit of analysis: Software practitioner
- Search unit: (see next section).

Sampling strategy

Blog divulgation

- Publication a post
  - Method: publication of a blog post
  - http://www.tumblr.com
  - http://medium.com
  - www.memoriacache.com.br
  - http://dev.to/
  - https://www.codeproject.com
  - https://slashdot.org
  - https://quora.com/ and https://pt.quora.com/

Social networks

Using the search strings listed below:

English	**Portuguese**
Software testing	Teste de software
Software developer	Desenvolvedor de software

Software development	Desenvolvimento de software
Software engineering	Engenharia de software
Software engineer	Engenheiro de software
Software security	Segurança de software
Software performance	Desempenho de software
Security testing	Teste de segurança
Performance testing	Teste de desempenho

- Linkedin (https://www.linkedin.com)
    - Criterion: search for the groups of the first five pages
    - Method: make a comment on the group page
- Facebook (https://www.facebook.com)
    - Criterion: search for the groups with more than 100 users. Limit of 50 groups by the search string
    - Method: make a comment on the group page
- Twitter (https://twitter.com)
    - Tweet with the following hashtag:
        - **English**: #softwaretesting #softwaredeveloper #softwaredevelopment #softwareengineer #softwareengineering #softwaresecurity #softwareperformance #securitytesting #performancetesting
        - **Portuguese**: #testedesoftware #desenvolvedordesoftware #desenvolvimentodesoftware #engenheirodesoftware #engenhariadesoftware #testedeseguranca #segurançadesoftware #desempenhodesoftware #testededesempenho
- Reddit (https://www.reddit.com/)
    - Criterion: search for subreddits using defined search strings (sort by '*top*')
    - Method: send message to the first 50 subreddits

Testing communities

- https://agiletesters.com.br/topic/271/f%C3%B3runs-comunidades-de-teste-de-software
- **http://gtsw.blogspot.com/**
- **http://www.aprendendotestar.com.br/comunidades.html**
- **http://guts-rs.blogspot.com/**
- **https://qualidadebr.wordpress.com/tag/comunidade-testadores/**
- https://www.ministryoftesting.com/

Questions-answers services

- Ask a question on https://sqa.stackexchange.com/
- Ask a question on https://softwareengineering.stackexchange.com/
- Ask a question on https://security.stackexchange.com/

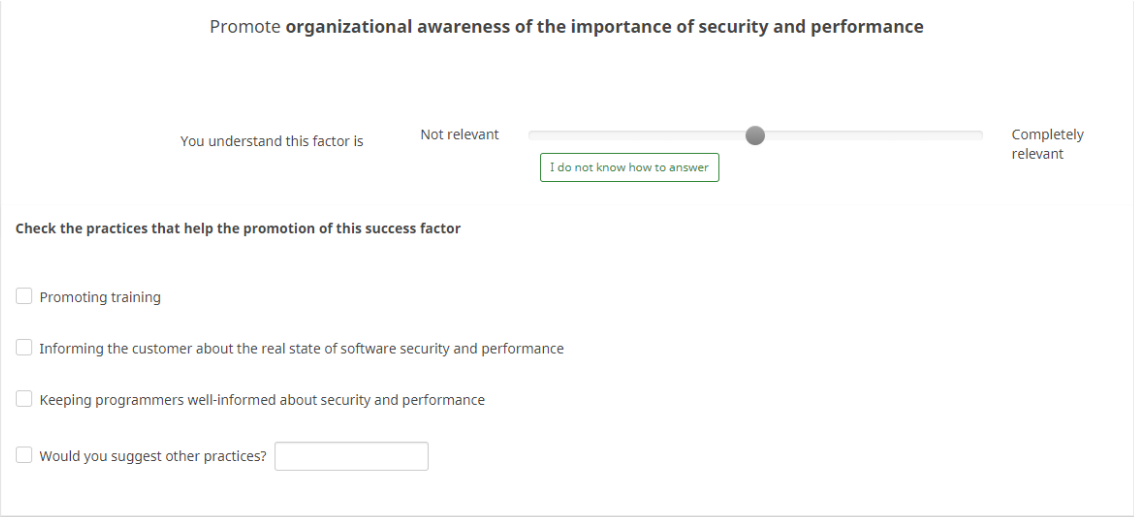
E-mail groups

- https://groups.google.com/forum/#!forum/guts-rs-sucesu
- https://groups.google.com/forum/#!forum/teste-de-software-pe

Resources

- Software: Internet browser, LimeSurvey, Statistical analysis software, Microsoft excel
- Questionnaire: an instrument composed of a list of questions. It can be answered online on the LimeSurvey platform or printed.

Survey questions



Promote **organizational awareness of the importance of security and performance**

You understand this factor is — Not relevant — Completely relevant

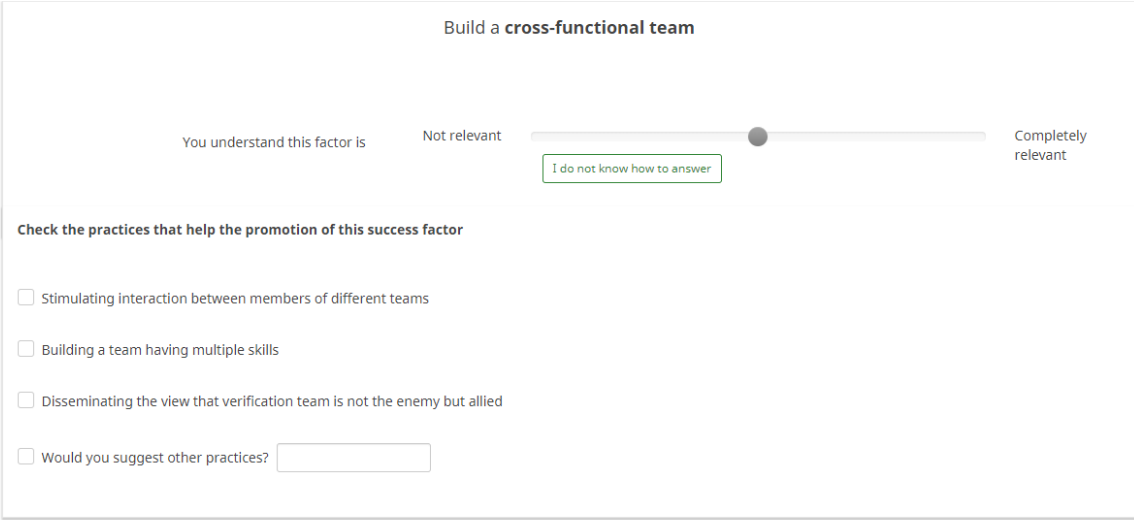
Check the practices that help the promotion of this success factor

□ Promoting training

□ Informing the customer about the real state of software security and performance

□ Keeping programmers well-informed about security and performance

□ Would you suggest other practices?



Build a **cross-functional team**

You understand this factor is — Not relevant — Completely relevant

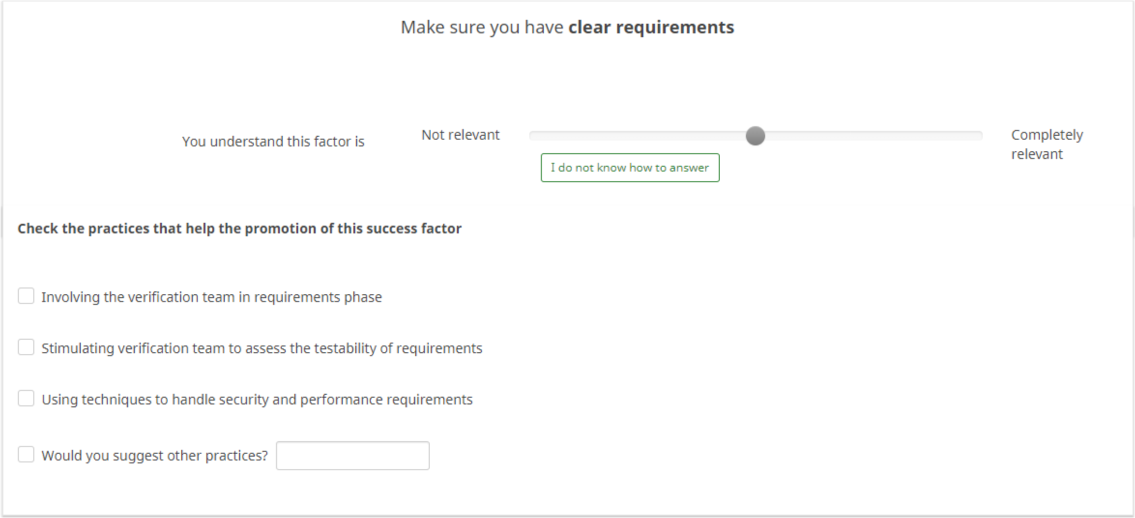
Check the practices that help the promotion of this success factor

□ Stimulating interaction between members of different teams

□ Building a team having multiple skills

□ Disseminating the view that verification team is not the enemy but allied

□ Would you suggest other practices?



Make sure you have **clear requirements**

You understand this factor is — Not relevant — Completely relevant

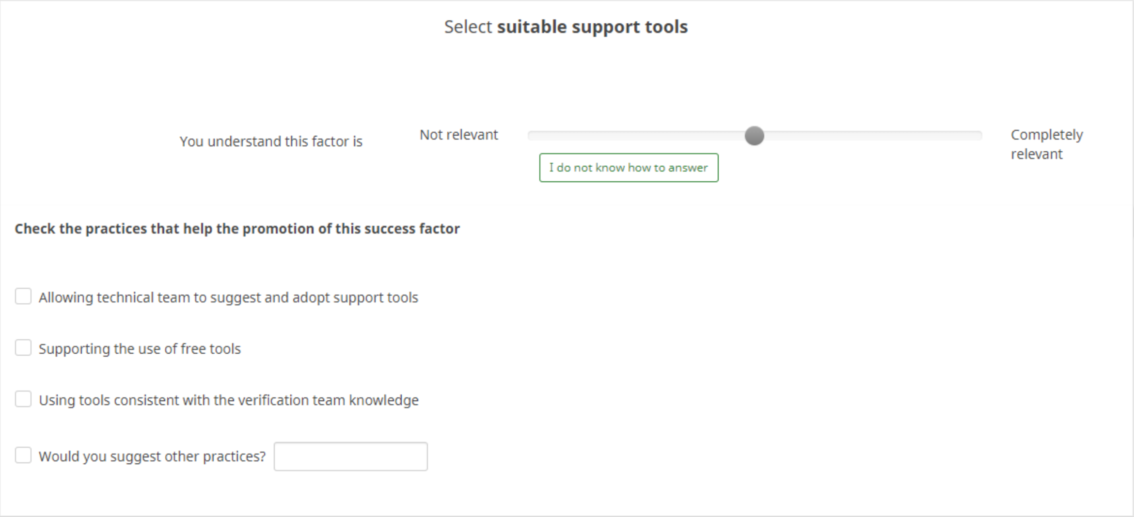
Check the practices that help the promotion of this success factor

□ Involving the verification team in requirements phase

□ Stimulating verification team to assess the testability of requirements

□ Using techniques to handle security and performance requirements

□ Would you suggest other practices?



Select **suitable support tools**

You understand this factor is Not relevant Completely relevant

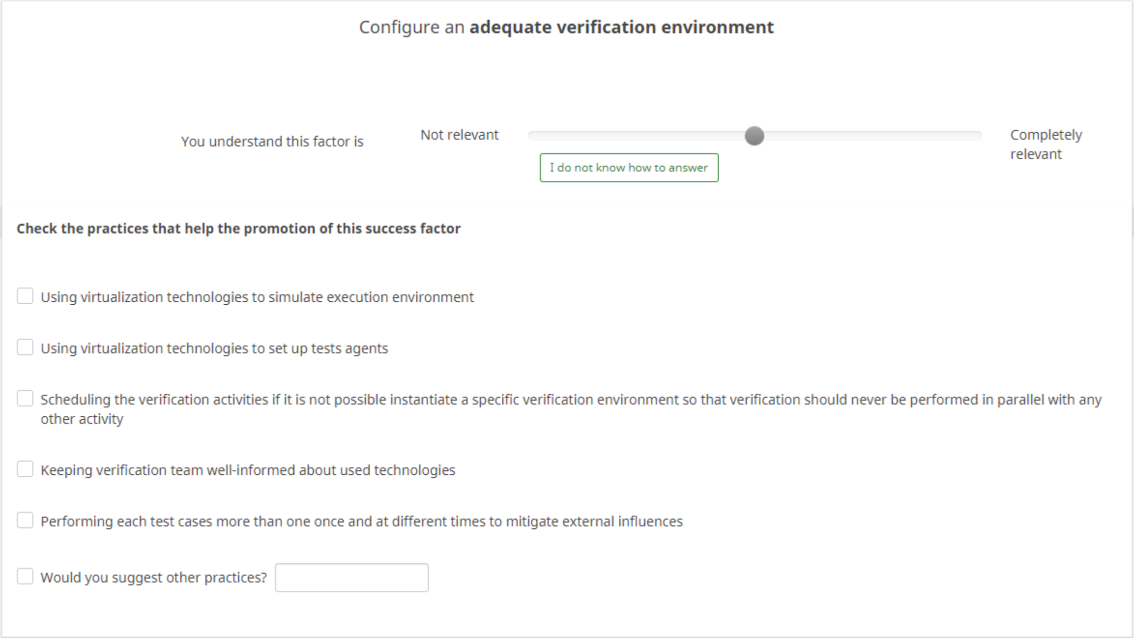
Check the practices that help the promotion of this success factor

□ Allowing technical team to suggest and adopt support tools

□ Supporting the use of free tools

□ Using tools consistent with the verification team knowledge

□ Would you suggest other practices?



Configure an **adequate verification environment**

You understand this factor is Not relevant Completely relevant

Check the practices that help the promotion of this success factor

□ Using virtualization technologies to simulate execution environment

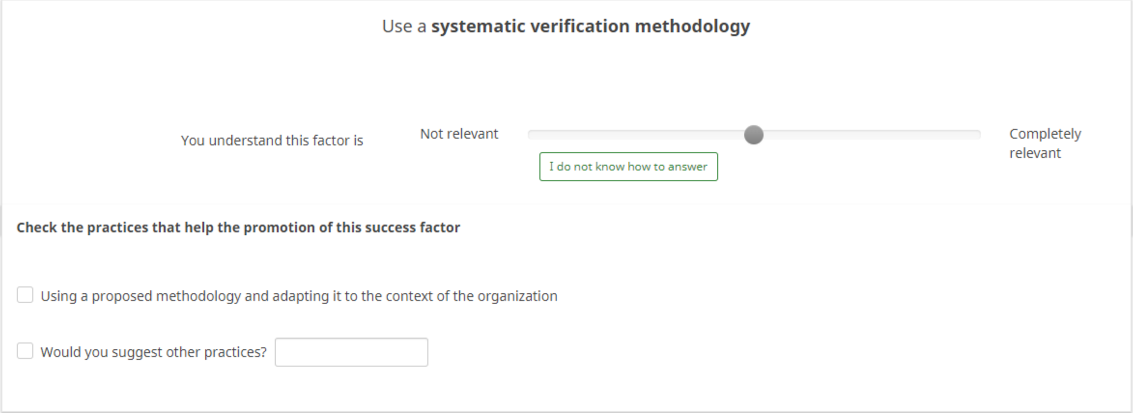
□ Using virtualization technologies to set up tests agents

□ Scheduling the verification activities if it is not possible instantiate a specific verification environment so that verification should never be performed in parallel with any other activity

□ Keeping verification team well-informed about used technologies

□ Performing each test cases more than one once and at different times to mitigate external influences

□ Would you suggest other practices?



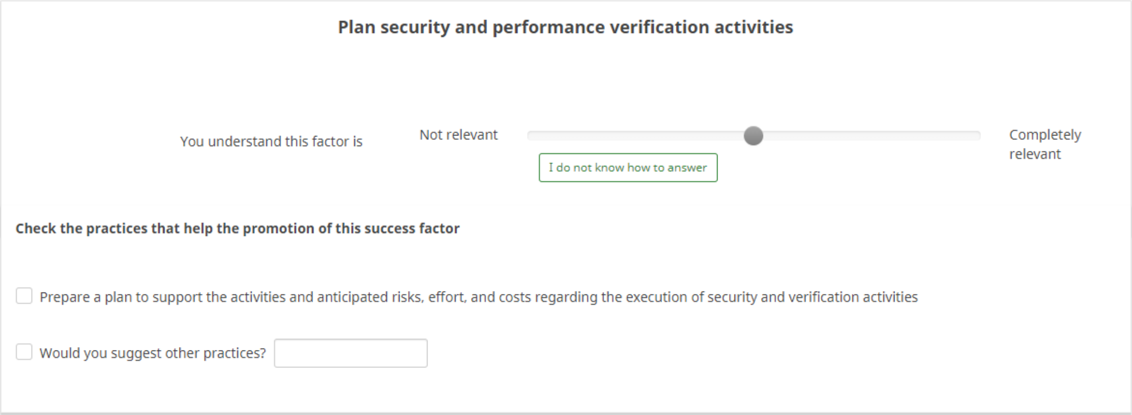
Use a **systematic verification methodology**

You understand this factor is Not relevant Completely relevant

Check the practices that help the promotion of this success factor

□ Using a proposed methodology and adapting it to the context of the organization

□ Would you suggest other practices?



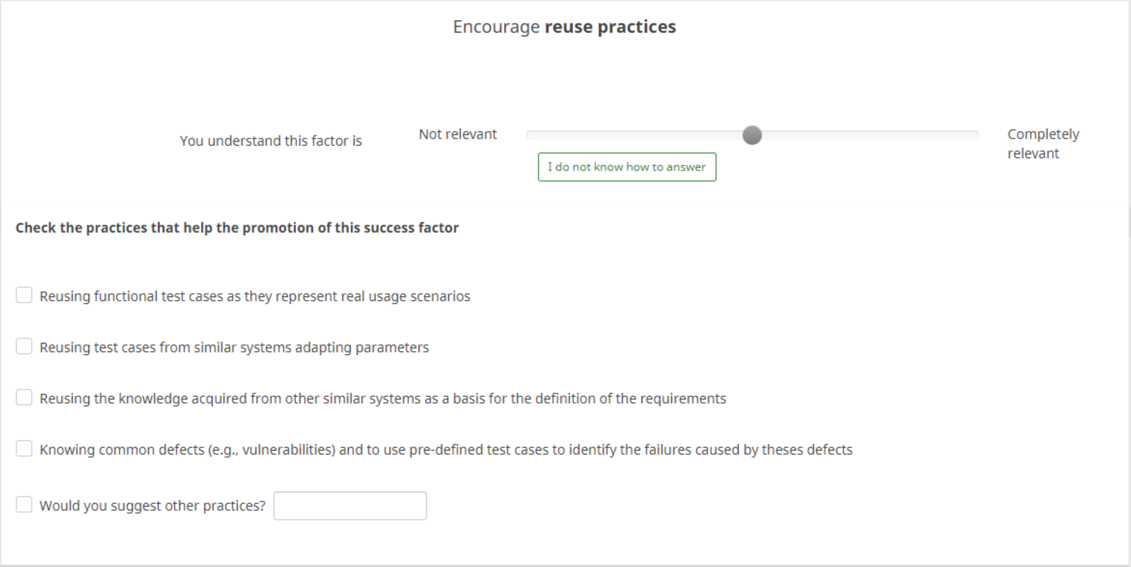
Plan security and performance verification activities

You understand this factor is Not relevant ● Completely relevant

Check the practices that help the promotion of this success factor

□ Prepare a plan to support the activities and anticipated risks, effort, and costs regarding the execution of security and verification activities

□ Would you suggest other practices?



Encourage **reuse practices**

You understand this factor is Not relevant ● Completely relevant

Check the practices that help the promotion of this success factor

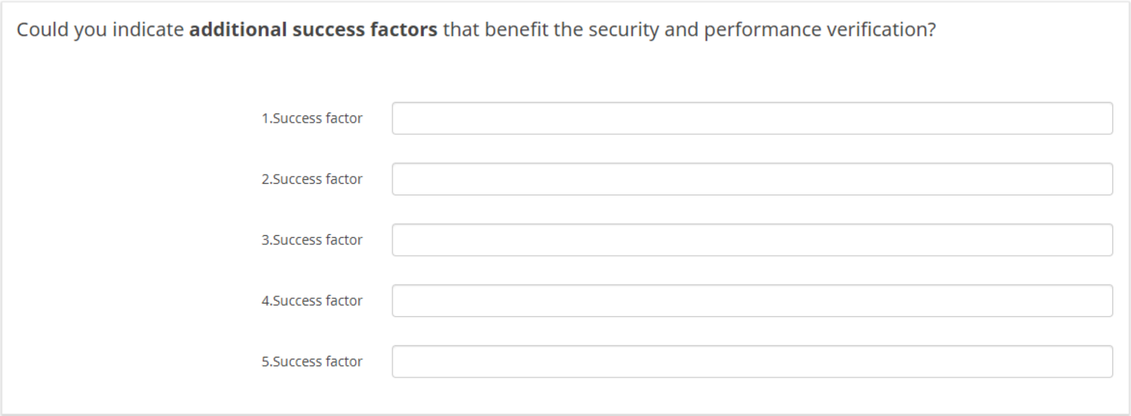
□ Reusing functional test cases as they represent real usage scenarios

□ Reusing test cases from similar systems adapting parameters

□ Reusing the knowledge acquired from other similar systems as a basis for the definition of the requirements

□ Knowing common defects (e.g., vulnerabilities) and to use pre-defined test cases to identify the failures caused by theses defects

□ Would you suggest other practices?



Could you indicate **additional success factors** that benefit the security and performance verification?

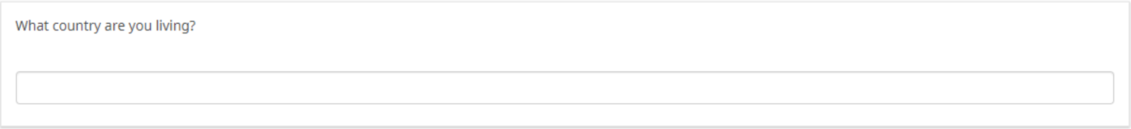
1.Success factor

2.Success factor

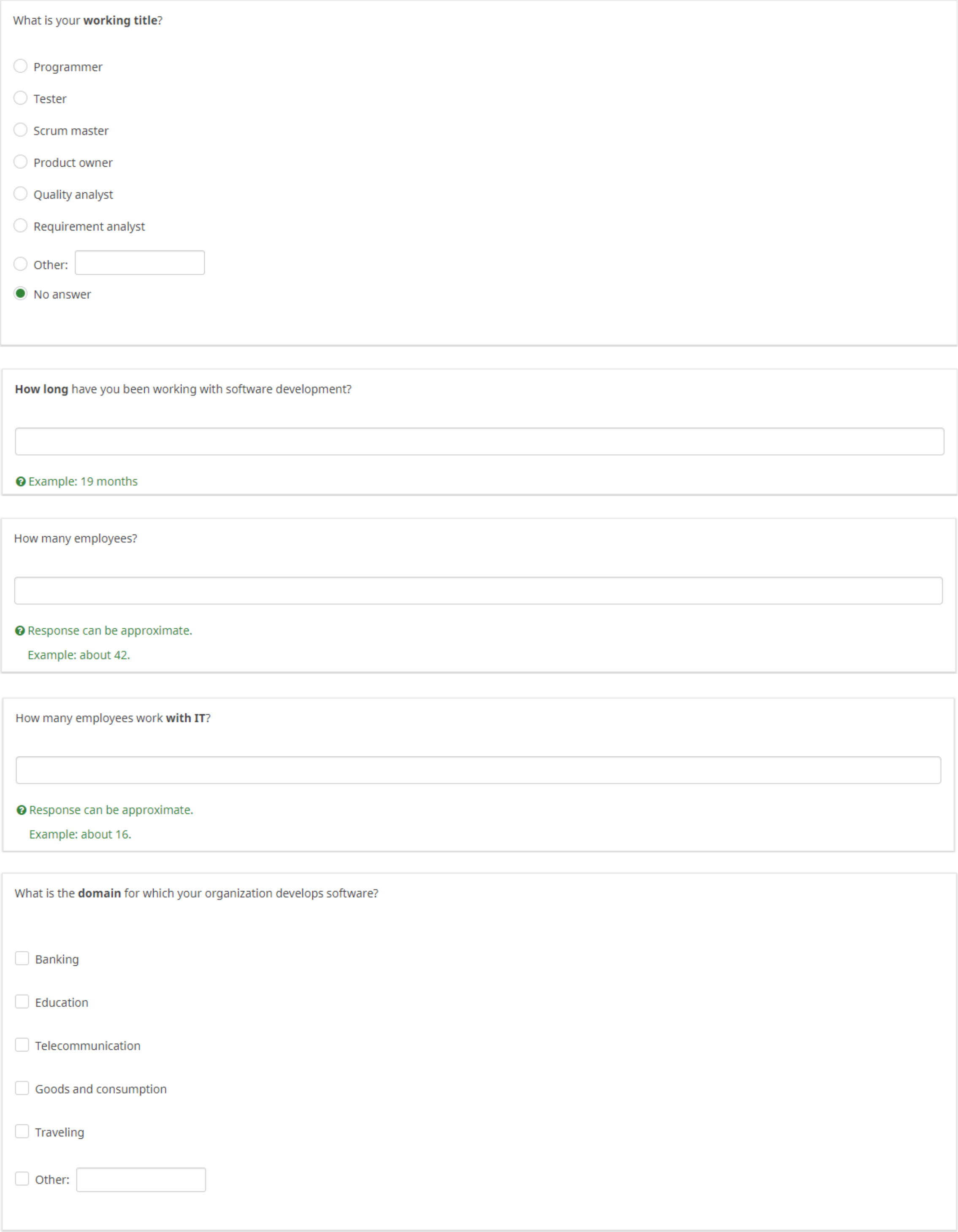
3.Success factor

4.Success factor

5.Success factor



What country are you living?



What is your **working title**?

○ Programmer

○ Tester

○ Scrum master

○ Product owner

○ Quality analyst

○ Requirement analyst

○ Other:

● No answer

How long have you been working with software development?

❓ Example: 19 months

How many employees?

❓ Response can be approximate.

Example: about 42.

How many employees work **with IT**?

❓ Response can be approximate.

Example: about 16.

What is the **domain** for which your organization develops software?

□ Banking

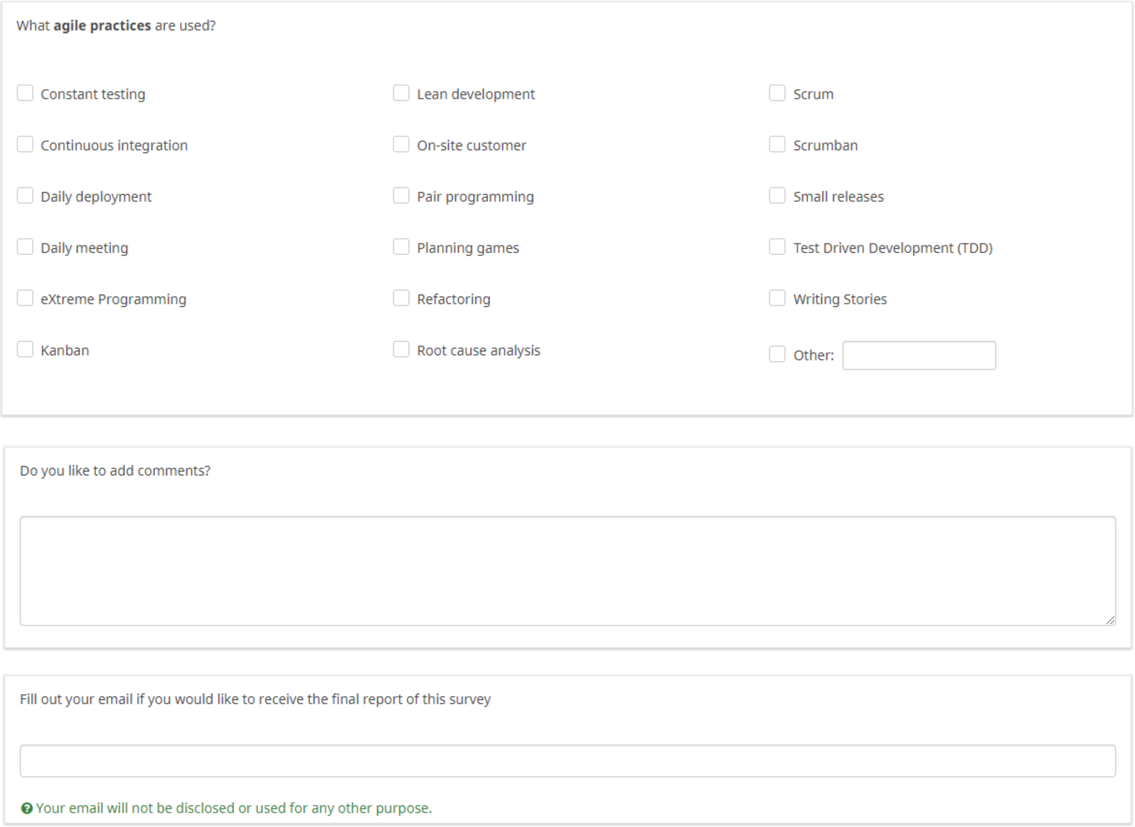
□ Education

□ Telecommunication

□ Goods and consumption

□ Traveling

□ Other:



What **agile practices** are used?

□ Constant testing

□ Continuous integration

□ Daily deployment

□ Daily meeting

□ eXtreme Programming

□ Kanban

□ Lean development

□ On-site customer

□ Pair programming

□ Planning games

□ Refactoring

□ Root cause analysis

□ Scrum

□ Scrumban

□ Small releases

□ Test Driven Development (TDD)

□ Writing Stories

□ Other: ______

Do you like to add comments?

Fill out your email if you would like to receive the final report of this survey

Your email will not be disclosed or used for any other purpose.

Threat to validity

- Generalization of the results: depending on the number of Brazilian organizations and managers answering the survey, the confidence level of the results, from a statistical point of view, might be low, therefore being difficult to generalize the results to the entire population. However, it is only possible to reach this conclusion after the answers are received.
- Conclusion validity: it might be impossible to extend some of the research questions to all professionals depending on the respondents' functions or roles in the organizations, as each role on the project might have a different perception.
- Construct validity: to improve the construct validity of the study, both the plan and the survey questionnaires will be reviewed by other researchers, discussed, and corrected if necessary. Also, a supervised pilot will be executed with at least one organization, to assure that the respondents have the same understanding of the questions as the researchers.

References

M. Linåker, Johan; Sulaman, Sardar Muhammad; Maiani de Mello, Rafael; Höst, "Guidelines for conducting surveys in software engineering v. 1.1," no. May 2015.