IoT Security Evaluation: A Set of Combined Strategies of Inspection, Testing, and Experimentation

Bruno Pedraça de Souza

PESC/COPPE – Universidade Federal do Rio de Janeiro (UFRJ)

bpsouza@cos.ufrj.br

Abstract. *IoT software systems research has significantly expanded recently, particularly with AI technologies' emergence and widespread adoption. Given their critical nature of continuous connectivity, such systems demand robust security measures. Therefore, it is fundamental to apply technologies that support their quality assurance (QA), specifically security. This thesis proposes strategies combining quality assurance technologies (software inspection, testing, and continuous experimentation) to enhance the final quality of these systems.*

1. Research Context and Problem

Over the past decade, IoT-based systems have grown in popularity, especially with the advancement of Artificial Intelligence (AI) technologies, cost reduction of devices and sensors, and sustainability. IoT generally integrates with other software systems, devices, sensors, middleware, and actuators to perceive, receive, and send environmental data (Motta et al., 2018). As with any paradigm shift, the emergence of IoT software systems brings challenges in their development, deployment, and ongoing evaluation, such as ensuring and addressing security concerns (Motta et al., 2019a).

Security[1] is one of the most important quality requirements. Issues related to security have often been a challenge for the software engineering community, for instance, cyber-attacks, and system code vulnerabilities. In IoT software systems, security challenges are even more critical and difficult as new attack surfaces emerge, such as device vulnerabilities, IoT Botnets, social engineering, etc (Pessoa and Travassos., 2024). To deal with these issues, quality assurance (QA) techniques, such as software inspection, testing, and continuous experimentation (CE), play an essential role and can be used to mitigate them.

Software inspection, testing techniques, and CE are typically applied sequentially throughout the development lifecycle. Inspections focus on identifying defects within software artifacts, like requirements and code, without needing an executable system. Conversely, testing involves executing the software to reveal failures in its behavior. Both software inspection and testing evaluate requirements. CE is a method for conducting experiments in many iterations to evaluate and evolve conjectural requirements of the software systems according to the final user's feedback (Auer and Felderer, 2018). A conjectural requirement is a ‘possible requirement’ with some level of uncertainty and must be experienced through an experimental study. Once

[1] We follow ISO definition regarding security: “*the degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization*.”

evaluated, the conjectural requirement is implemented as a real system (or feature) requirement or discarded.

To better illustrate some IoT security issues that QA techniques and CE can assess, I will use a real example in a research project case. The problem is secure communication channels[2] that must be implemented in an IoT software system. It was necessary to ensure the communication channel was secure when exchanging data between things (devices) and software systems. To achieve this, a viable solution would be to implement an encryption solution. It is worth highlighting that the software in the system had low-cost devices, so choosing the 'best' encryption algorithm would not be feasible. A brief example is described below.

> *The security requirement specified for the project context was: The software system must use DES[3] algorithms. Such a requirement demonstrates that it was not clear enough for the development team to implement, thus requiring a requirements inspection step. Thus, the security requirement specified was verified by software inspection and corrected. The corrected requirement is that the software system must use a cryptography component to process data received/sent from/to systems and devices (De Paiva et al., 2023). Once this requirement was corrected and implemented, it was possible to describe the test cases for the testing. However, our development team has encountered another issue, which DES algorithm to use? So, after identifying some algorithms, the development team chose SPECK. SPECK was chosen because it was proposed to be used in low-power computing devices, such as sensor networks, including IoT devices (De Paiva et al., 2023).*
>
> *Thus, our team did not know the impact of incorporating SPECK into the software system. This issue was difficult to evaluate since the development team must consider the software and hardware characteristics. One possible solution was CE since the development team could have different software versions using different encryption algorithm solutions. So, De Souza et al. (2023b) applied CE to evaluate the solution of the secure communication channel and choose the best way to implement the cryptography algorithm and how it would impact the software system's performance. As their results, the authors infer that CE can be useful to evaluate some security issues.*

It is well-known that software inspection, testing techniques, and CE are useful when used correctly, although they have limitations. In this Doctoral Symposium paper, I discuss how using software inspection, testing, and CE combined can be more effective in enhancing the quality of IoT software systems.

This thesis proposes a set of strategies to evaluate the security of software systems. Figure 1 shows an extract of two combinations. The current version of the strategies is composed of seven strategies. Each strategy must be applied individually. Furthermore, each of these processes is used in a specific development context. Section 4 introduces an excerpt of the proposal.

---

[2] A secure communication channel is defined as transmitting data between two or more parties that protect the confidentiality, integrity, and authenticity of the information exchanged.

[3] DES means Data encryption standard.

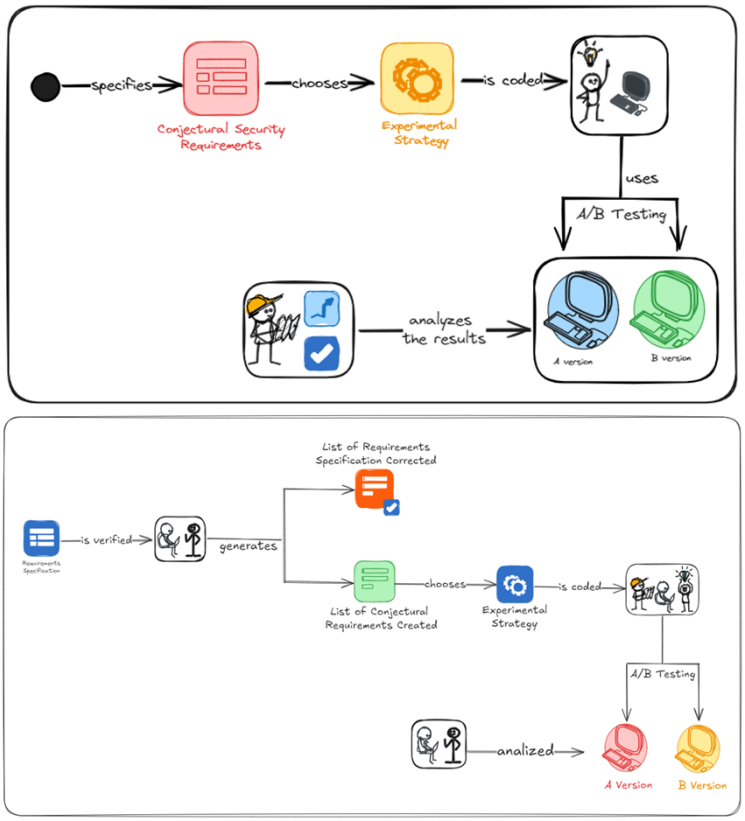

Figure 1. Set of the Combination of QA techniques and CE to evaluate security.

2. Research Goals and Questions

For this thesis proposal, I present the main research question (RQ) guiding this proposal: "*Does the combination of software inspection, testing techniques, and continuous experimentation increase confidence in the systems' quality regarding evaluating security requirements in IoT software systems?*" This research question was split into the following secondary questions:

RQ1: *What are the available security requirements issues in the technical literature concerning IoT software systems?* This research aims to identify and categorize the security problems prevalent in IoT software, as reported in the state of the art.

RQ2: *What can verification, validation, and continuous experimentation technologies be used separately to tackle security requirements issues in IoT software systems?* This research question aims to identify the main V&V for assessing the security requirements found in RQ1 in the technical literature. Furthermore, we strive to identify the strategies and processes used by CE.

RQ3: *How can verification, validation, and CE technologies be combined to address security requirements in IoT software projects?* The goal of this RQ3 is to figure out how verification, validation, and CE can be merged or used to ensure that security requirements are met according to the users' goals.

RQ4: *Would a computational infrastructure support the development team choose the most appropriate strategy to solve such security issues?* This research question aims to verify whether a computational tool helps the development team choose the proper set of verification, validation, and CE technologies to assess security requirements in IoT software systems.

RQ5: *Do the proposed strategies and their computational support tool present adequate results in the development team's respective uses?* This research question aims to evaluate their combination and computational support tools through experimental studies to acquire evidence of their usefulness, efficiency, and effectiveness[4].

Each RQ is associated with at least one sub-objective. Thus, this research aims to provide QA strategies that combine software inspection, software testing, and experimentation technologies to have (conjectural) security requirements implemented with high quality in IoT software systems. To do so, we split the general goal and better detail, it consists in conducting a series of experimental studies the following sub-objectives:

1. Identify and investigate the IoT security requirements available in technical literature through secondary studies.
2. Identify quality assurance technologies, such as software inspection, software testing, and CE, to evaluate IoT software systems' security.
3. Propose strategies to combine software inspection, testing, and experimentation to address the security requirements of IoT software systems.
4. Build and evaluate the feasibility of a computational infrastructure for supporting software practitioners to operationalize the combination of adequate software inspections, software testing, and experimentation technologies for IoT software systems.

Up to now, I have reached sub-objective 3, in which we proposed a set of strategies.

3. Research Methodology

A well-defined research method is necessary in any scientific work since its steps must be clearly defined and described. Thus, in this ongoing thesis proposal, we are conducting an evidence-based methodology (Mafra et al., 2006). It consists of conducting a series of experimental studies (primary and secondary) to mature the approach to be designed, engineered, and adopted by practitioners. Besides, this methodology has traditionally been used in research carried out by the Experimental Software Engineering (ESE) group at COPPE/UFRJ, and it is an extension of the method proposed by Shull et al. (2001), as presented in Figure 2.

[4] *Effectiveness*: number of security issues resolved by total number of issues identified.

Figure 2. Research methodology.

- *Literature Studies*: we have conducted two studies (Erthal et al., 2023; De Souza et al., 2023a) to observe the State of the Art of Continuous Experimentation in IoT Software Systems and Software Engineering.
- *Thesis Proposal*: this step aims to define initial strategies for evaluating security requirements and vulnerabilities using software inspection, testing, and CE in software development (Figure 1).
- *Set of Experimental Studies*: to validate the thesis proposal, it is intended to plan and execute primary studies regarding the proposed approach. The study follows Mafra's methodology (2006). So, four primary studies, from a feasibility study to a case study, are planned to evaluate and evolve the proposed strategies and software technology before transferring it to the industrial environment. Figure 3 shows the methodology applied in this step (to be carried out).
    - The feasibility study evaluates whether the proposed software technology achieves its goal of assessing security/vulnerability requirements using the selected approach.
    - The observational study aims to improve the proposed software technology to understand its application better. In general, it can allow its evolution to a second version (in progress).
    - Case studies 1 and 2 aim to verify whether the proposed strategy fits or can be used in industrial scenarios/development contexts.

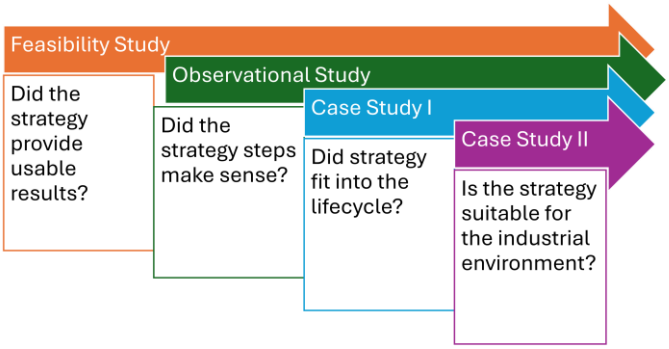

Figure 3. Steps to be executed in Experimental Studies.

4. IoT Security Evaluation: Towards Combined Strategies

This section presents an initial proposal to use and combine software inspection, software testing, and CE to support the quality of security requirements in the IoT context. These combinations can potentially strengthen IoT software security and guide the implementation and evaluation of security aspects throughout the development process.

Seven strategies were proposed for the evaluation of security aspects. Some of them are already known as inspection, testing, and CE. However, I proposed specific artifacts to support the security assessment of each technique. The strategies are:

The strategies can be used individually, as happens in traditional software development, i.e., they can only be used for inspection, tests, and experimentation. The choice of the most appropriate strategy will rely on the characteristics of the IoT software systems and their artifacts. Each strategy focuses on the security aspects of the IoT software system and comprises planning and execution. Next, I will present one strategy and its characteristics.

4.1 Evaluating Security Requirements and Conjectural Requirements Using Software Inspections and Experimentation

Experimentation and inspection together form one of the seven strategies. This strategy is initiated in the first phase of the development process, requirements specification, and is applied as the requirements documents are being defined. Initially, a checklist is used to verify the requirements artifacts, identifying defects and inconsistencies. Once these documents have been inspected, any detected defects must be corrected. These defects can be identified by applying a specific checklist (Figure 4a). CE is closely linked to inspection, requiring a thorough understanding of the system requirements. This knowledge is essential for distinguishing between system requirements and conjectural ones. As requirements inspection relies on the proposed checklist, experimentation utilizes a template to generate conjectural security requirements (Figure 4b).

	Questions
16	Are security requirements necessary?
17	Are highly complex passwords required, for example using a combination of numbers, characters and symbols (letters and special characters)?
18	Are passwords required to be updated every x days?
19	Are there ways described to protect delivered and received data from unauthorized people?
20	Are encrypted communication channels required?
21	Have authentication mechanisms been identified? Is it explained in the requirement document? For instance, two-factor authentication (2FA), or digital certificates.
22	Are requirements specified with sufficient information about security policies?
23	Is there a need to have backup policies?
24	Is communication among subsystems clearly defined?
25	Is it specified that only devices registered in the system send and receive data?
26	Are there requirements to finish the session if the maximum inactivity time is exceeded?
27	Have audit and continuous monitoring policies been described?
28	Are Virtual Private Networks (VPNs) necessary?

A

Conjectural Requirements (CRs)		
ID: CR [2]	**CR**: [Proposed]	**Priority:** [High]
Description of CRs	**The software system is expected to have** a secure communication channel. **So** encrypted communication. **However, we do not know**: 1. the impact of using encrypted communication channels on system performance. **The hypotheses are:** H0: There is no difference between SAFE IoT level of protection and performance when using or not using an encryption algorithm. H1: The SAFE IoT level of protection improves, and performance decreases when using an encryption algorithm. H2: The SAFE IoT level of protection improves, and performance does not decrease when using an encryption algorithm. H3: The SAFE IoT level of protection does not improve, and performance decreases when using an encryption algorithm.	
Metric Defined	The metrics used were the level of protection and time (in seconds) of sending messages to a broker.	
Experimental Method	We used A/B testing to evaluate this conjectural security requirement.	

B

Figure 4.A The checklist for evaluating security requirements. Figure 4.B The template for specifying and configuring CE.

The step of checking the requirement artifact is vital because it defines and divides the requirements into system and conjectural requirements. Then, the software engineer can differentiate what is a requirement and what is a conjectural requirement. A system will implement the conjectural requirements to determine whether they are required or not. Even if the system has not been implemented, planning an experimental study on the conjectural requirement is still possible. Figure 5 shows the flow of the strategy to be followed.

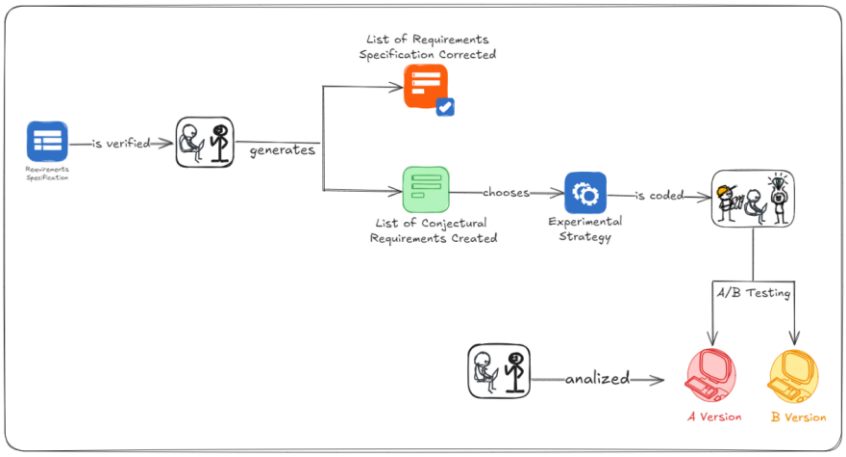

Figure 5. Strategy to assess security requirements using software inspection and conjectural security requirements using experimentation software.

The strategy's application scenario would be to apply the checklist to all requirements artifacts created. Then, we can list all requirements, specifically security, for which the development team is unsure about their implementation and verify their feasibility through CE. Finally, the development team corrects the system requirement artifacts and plans the conjectural requirements evaluations.

4. Preliminary Results and Ongoing Studies

Several results were obtained following the methodology steps (Section 3). Regarding the literature studies, organizing the body of knowledge on continuous experimentation in Software Engineering was possible. The first literature study aimed to characterize CE by examining its definitions, processes, and experimental strategies (Erthal et al., 2023). The second literature study explored works that applied CE to contemporary software systems (CSSs), highlighting the main challenges and opportunities of using CE in CSSs (Souza et al., 2023). These literature studies provided a foundation for this thesis. In alignment with these studies, IoT software projects were crucial in establishing combined strategies. I have successfully achieved sub-objective 3.

As for the next steps, I am evaluating the checklist proposed to verify if it is effective and efficient in detecting defects within the specified requirements for security. I am also planning experimental studies to evaluate each proposed strategy using cases. All artifacts, like the security checklist and the conjectural requirement template, must be evaluated. The strategy must be validated before implementing computational support. It is worth highlighting this research began in mid-2020 and is expected to be completed in the middle of 2026.

5. The Road Ahead

This research originated from an initial problem encountered in an IoT software project and was further shaped by two literature studies. The motivation behind this thesis proposal is the potential to combine software inspection, software testing, and continuous experimentation to assess the security aspects of IoT software systems while maintaining a high level of quality.

Security, a major concern for any software system, requires special attention and has been extensively studied over the last few years. However, final solutions are far from being provided. Therefore, creating means to facilitate the evaluation of this requirement is advantageous. Once the strategies are evaluated and validated, the aim is to transfer them to the industrial environment so that development teams can take a more evidence-based approach.

References

Auer, F., and Felderer, M. (2018) "Current state of research on continuous experimentation: a systematic mapping study." 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2018, p. 335-344.

De Paiva, B.D., De Souza, B.P., and Travassos, G. H. (2023) “CryptoComponent: um Componente de Criptografia para Sistemas de Software IoT de Baixo Custo.” In Anais Estendidos do XIV Congresso Brasileiro de Software: Teoria e Prática (pp. 90-99).

De Souza, B.P., de Paiva, B.D., and Travassos, G. H. (2023a) “Using Experimentation to Evaluate Security Requirements in IoT Software Systems.” In 2023 Symposium on Internet of Things (SIoT) (pp. 1-5).

De Souza B.P., Santos P.S., and Travassos G.H. (2023b) “On Challenges and Opportunities of Using Continuous Experimentation in the Engineering of Contemporary Software Systems.” In Proceedings of the XIX Brazilian Symposium on Information Systems 2023 May 29 (pp. 372-379).

Erthal, V.M., De Souza, B.P., dos Santos, P.S.M., and Travassos, G.H. (2023). “Characterization of continuous experimentation in software engineering: Expressions, models, and strategies.” Science of Computer Programming, 229, 102961.

ISO/IEC 27000, 2018. Information technology — Security techniques — Information security management systems — Overview and vocabulary. https://standards.iso.org/ittf/PubliclyAvailableStandards/index.html.

Mafra, S.N., Barcelos, R.F., and Travassos, G. H. (2006) “Aplicando uma metodologia baseada em evidência na definição de novas tecnologias de software.” In Anais do XX Simpósio Brasileiro de Engenharia de Software, 2006, p. 239-254. SBC.

Motta, R. C., DE Oliveira, K. M., Travassos, G. H. (2018) "On challenges in engineering IoT software systems," Proceedings of the XXXII Brazilian Symposium on Software Engineering, New York, NY, USA, ACM, 2018, p. 42–51. DOI: 10.1145/3266237.3266263.

Motta, R. C., Silva, V., Travassos, G. H. (2019) “Towards a more in-depth understanding of the IoT Paradigm and its challenges.” JOURNAL OF SOFTWARE ENGINEERING RESEARCH AND DEVELOPMENT, v. 7, p. 1, 2019.