Managing Vulnerabilities in Software Projects: the Case of NTT Data

Sabato Nocera
Department of Computer Science
University of Salerno
Fisciano, Italy
snocera@unisa.it

Simone Romano
Department of Computer Science
University of Salerno
Fisciano, Italy
siromano@unisa.it

Rita Francese
Department of Computer Science
University of Salerno
Fisciano, Italy
francese@unisa.it

Riccardo Burlon
NTT Data
Salerno, Italy
riccardo.burlon@nttdata.com

Giuseppe Scanniello
Department of Computer Science
University of Salerno
Fisciano, Italy
gscanniello@unisa.it

***Abstract*—Background: *Software vulnerabilities* are flaws in application source code that can be exploited to cause harm, hence companies must devise strategies to manage them.**
Aim: We want to understand how software vulnerabilities are managed in a big IT (Information Technology) service and consulting company like *NTT Data*.
Method: We conducted a focus group involving six software professionals working at NTT Data and analyzed the gathered data through a thematic analysis approach.
Results: We found that application security standards are defined based on the needs of the clients (*i.e.,* companies that commissioned NTT Data the software to be developed) and the projects' nature (*i.e.,* the development of greenfield projects vs. maintenance of existing ones). Also, to detect software vulnerabilities, *SAST* (*Static Application Security Testing*) tools are mainly used; among these, *SonarLint* and *SonarQube* appear to be the de-facto standards for NTT Data. Finally, not all software vulnerabilities are fixed; for example, the presence of some software vulnerabilities is tolerated by the clients, who take on the responsibility of not removing these vulnerabilities.
Conclusions: It seems that developers and NTT Data clients are not averse to securing their code. NTT Data follows the application security standards established with their clients. To detect software vulnerabilities, SonarLint and SonarQube appear to be the de-facto standards, so explaining to some extent the increasing attention on these tools by the software engineering research community.

***Index Terms*—Software Vulnerability, Focus Group, Qualitative Study**

I. Introduction

Software vulnerabilities (also known as *security issues* or *security concerns*) are flaws in the computational logic that would negatively affect confidentiality, integrity, and/or availability of a software system [1]. An attacker might exploit the vulnerabilities of a software system to damage its stakeholders. A few examples of stakeholders are the developers and the end-users of the software system, but also the possible client who commissioned NTT Data the software to be developed. The economic impact of software vulnerabilities can be dramatic, as in the case of *Heartbleed*, a software vulnerability in the popular *OpenSSL* cryptography library, which has cost at least USD 500 million to companies [2].

The common belief is that the responsibility for the introduction and presence of software vulnerabilities is ultimately of the developers. However, Assal and Chiasson [3] observed in their survey—conducted to explore the existing relationship between developers and secure software development—that developers are not averse to using security practices, but they simply follow company policies. On the other hand, security is often ignored because it may be considered a secondary requirement [4]. In this respect, it seems important to improve our body of knowledge on the role of both developers and companies in the management (*i.e.,* development and maintenance) of secure software.

As a first step in such direction, we asked one of the most important companies in our contact network to take part in a qualitative study. Unlike quantitative investigations, qualitative ones allow gaining a deeper understanding of the reasons and motivations behind a given phenomenon [5]. In particular, we conducted a focus group involving six software professionals working at *NTT Data*, a multinational company operating in the IT (Information Technology) service and consulting field with a revenue of over USD 20,000 billion in 2021. The company has a global presence and operates in more than 50 countries, with a workforce of over 150,000 employees worldwide, serving clients in various industries such as finance, healthcare, and the public sector.

We used a thematic analysis approach to analyze the transcript of the audio recording of the focus group session, as well as the notes taken by the researchers during that session.

The most important results of our qualitative investigation can be summarized as follows:

(i) Application security standards are defined based on clients' needs and projects' nature;
(ii) To detect software vulnerabilities, *SAST* (*Static Application Security Testing*) tools are mainly used;

(ii) *SonarLint* and *SonarQube* appear to be the de-facto standards for NTT Data;

(iv) Not all software vulnerabilities are fixed.

The rest of the paper is organized as follows. In Section II, we summarize research related to ours. We present the planning and execution of our focus group in Section III. The study results are presented in Section IV. We then discuss the obtained results and possible limitations of our study in Section V. Final remarks conclude the paper in Section VI.

II. Related Work

Few investigations have been conducted to study how software vulnerabilities are managed by companies [3], [6]–[8].

Weir *et al.* [6] analyzed a dataset of software security development activities led by developers. The authors point out that the adoption of security practices increased slowly until 2015, and then accelerated. This coincided with a greater focus on supporting cloud-based deployment and prioritizing component security.

Braz and Bacchelli [7] interviewed ten professional developers and surveyed 182 practitioners to investigate the assessment of software security in code reviews. Their results show that, despite developers recognizing their responsibility to uphold security standards during code reviews, security does not hold a prominent position among their priorities. On the other hand, companies offer limited assistance and incentives to developers (*e.g.,* lack of security training), yet they expect developers to still ensure security during reviews.

Assal and Chiasson [3] surveyed 123 software developers employed in North America, in order to investigate the existing relationship between developers and secure software development. The focus of their study was on how developers influence and are influenced by these security practices. The authors discovered that the primary challenges in effectively managing software vulnerabilities often arise due to insufficient organizational or process support throughout the various stages of development. On the other hand, developers acknowledge the importance of software security and are not reluctant to adopt secure programming practices.

Palombo *et al.* [8] conducted an ethnographic study of secure software development processes in a software company. They collected data for a year and a half by using the anthropological research method of participant observation. They found that software vulnerabilities arise from deliberate actions or oversights, which can be attributed to the difficulties associated with managing the various stakeholders' responsibilities in an economic ecosystem. Consequently, it is inaccurate to attribute the introduction of these software vulnerabilities solely to the insufficient knowledge or skills of developers.

Our study differs from the aforementioned ones because we conducted a focus group aimed to understand how vulnerabilities are managed in a big IT service and consulting company like NTT Data. One of the peculiarities of focus groups is their capacity to foster group interaction, which often results in the exploration of themes originally not planned, that are regarded as significant by the participants of the focus group [9].

III. The Focus Group

The use of empirical methods in software engineering is increasing, indeed a wide range of them has been defined to address different research problems [10]. Among these different kinds of empirical methods, the focus group is quick and cost-effective to obtain qualitative insights and feedback [11]. Rather than providing quantifiable responses to a specific research question, a focus group study provides a flow of input and interaction among participants related to a given topic of interest [12]. More in detail, a focus group is a planned discussion that researchers design to obtain personal perceptions of individuals (also participants from here onward), arranged in a group, on a defined area of interest [13]. Focus groups have the benefit of producing upfront and sometimes insightful information by fostering group interaction—*e.g.,* this can result in the exploration of topics not included among those initially defined, but that is deemed important by the participants [9]. The typical size of a focus group is between three and 12 participants [10].

We planned and conducted our focus group by following the guidelines by Kontio *et al.* [11], suggested for research in the software engineering field.

A. Defining the research problem

The aim of our focus group was to gain insights into how companies—specifically, NTT Data—manage software vulnerabilities in the code of the software systems they develop and/or maintain, delving into their policies and instruments for managing software vulnerabilities, as well as the process for resolving them.

B. Planning and execution of the focus group

We involved six software professionals working in the Italian division of NTT Data, who had expressed their willingness to take part in the study.

The services offered by NTT Data are divided into several key areas, including application development and management, IT consulting, digital transformation, cloud services, infrastructure services, and cybersecurity. The Italian division of NTT Data is known for its expertise in delivering large-scale software projects, such as the implementation of digital payment systems for major Italian banks, the development of IoT (Internet of Things) solutions for the manufacturing sector, and the realization of digital platforms for the public administration.

The participants in our study had: different job positions, direct work experience with software security practices, and long careers as software professionals (on average, more than 16 years, see Table I). Moreover, the participants belonged to different teams and worked on different software projects.

The focus group was held at a local headquarters of NTT Data in Salerno, Italy, during a one-hour session conducted in Italian[1] and audio-recorded. One of the authors of this paper (*i.e.,* the second) played the role of the *facilitator* of

[1]All the participants were Italian.

TABLE I: Demographic information on the six participants.

Position in the company	**Job tenure (years)**	**Experience as professional (years)**
1) Project Manager	3	17
2) Project Manager	4	17
3) Technical Project Manager	3	14
4) Team Lead & Senior Test Automation Engineer	5	14
5) Test Manager	3	15
6) Solution Architect	5	20
Average	*3.83*	*16.17*

the focus group—*i.e.,* he was responsible for ensuring that all relevant topics were addressed, limiting deviations in the discussion, and encouraging the participants to express their opinions, all while avoiding conditioning the participants. Two other researchers (*i.e.,* the first and the third) were instead responsible for collecting notes during the focus group session.

The focus group session began with an introduction in which the aim and ground rules of the session were explained to the participants (the fifth author was responsible for such introduction). In particular, the facilitator emphasized that the participants' contributions should represent the company's situation from their perspective and that we would ensure the confidentiality and anonymity of all data collected, including the identity of the participants. The facilitator also made it clear that the research would have been published with the company's name only if the company had wanted to.

According to a pre-defined script, the discussion followed a semi-structured approach in which the facilitator guided it by addressing the topics of interest one after another; in this way, it was possible to stimulate a prolific dialogue and engage all the participants while remaining consistent with the objective of the study.

The topics of interest initially identified, and thus covered by the interview script, are reported in Table II. The participants were prompted with open-ended questions about each of the aforementioned topics, therefore all of them had the opportunity to contribute to the discussion and we could further investigate key insights arising from their contributions. The greater part of the questions was initially defined in the script, albeit slightly different from how they were asked in the focus group; other questions emerged from the discussion with the participants. A non-exhaustive list of questions, translated into English, is provided below.

1) What is the company's policy regarding the management of software vulnerabilities?
2) How do you assess software security and when do you do so during the software development life cycle?
3) Which are the instruments you usually use for managing software vulnerabilities?
4) Who is usually responsible for resolving software vulnerabilities?
5) Are there software vulnerabilities you do not fix and what are the reasons for leaving them unfixed?

C. Analysis

We used a thematic analysis approach, namely template analysis [14], to analyze the transcript of the audio recording and the notes taken by the researchers during the focus group.

Thematic analysis is a qualitative method used to identify patterns, themes, and interpretations in data [15]. The main difference between template analysis and a conventional thematic analysis approach concerns the usage of a template—*i.e.,* a predefined and hierarchical set of themes. In template analyses, researchers develop templates based on their experience and knowledge of the phenomenon under investigation. However, templates are not fixed as their themes and hierarchical structure can evolve as the analysis progresses.

We used template analysis because of its flexibility and swiftness [14]. In our case, the template was created based on our teachers' and researchers' experience with software vulnerabilities and on the results of our study of the state of the art. To deal with researcher bias [16], the template analysis was executed by two authors of this paper (the first and second authors). In particular, this was to limit possible subjective evaluation during the template analysis.

IV. Results

In Table II, we show the initial template, whose themes were covered during the focus group. The themes emerging from the gathered data were consolidated in the final template reported in Table III.

Below, we present the themes of the final template, each including two/three sub-themes (see Table III).

A. Theme 1) Policy

This theme includes two sub-themes related to clients' needs and projects' nature.

a) Client's Needs. The contract with the client establishes the policy for managing software vulnerabilities, including the application security standards to be guaranteed and the instruments used for software vulnerability management—*e.g.,* SAST or *Dynamic Application Security Testing* (*DAST*) tools.

The application security standards are based on the severity classification of software vulnerabilities detected by SAST and/or DAST tools. For example, the company's base application security standards include no blocker software vulnerability (detected by using a SAST tool, namely *SonarQube*)

TABLE II: Initial template.

Theme	Focus
1) Policy	On the policy for managing software vulnerabilities
2) Instrumentation	On the instruments used for managing software vulnerabilities
3) Resolution	On the process followed to resolve software vulnerabilities

TABLE III: Final template.

Theme	Sub-theme
1) Policy	a) Client's needs
	b) Project nature
2) Instrumentation	c) SAST/DAST Tools
	d) Unit testing
	e) Third-party software
3) Resolution	f) Resolution in codebase
	g) Resolution in third-party software
	h) Resolution without fixing

and at most 30 critical software vulnerabilities in the delivered software [2].

The policy for managing software vulnerabilities, and thus the application security standards, depends on the client's needs—*e.g.,* those operating in the public administration or banking sector demand higher application security standards.

b) Project Nature. The company business is focused on the *(i)* development of *greenfield projects* and *(ii)* maintenance of *existing projects*.

The policy for managing software vulnerabilities depends on the kind of project. In particular, when developing a greenfield project, the company is responsible for the management of any software vulnerability. On the other hand, when taking charge of the maintenance of an existing project, it is necessary to assess how secure the inherited software is since software vulnerabilities in the inherited software are not usually under the responsibility of the company. If software vulnerabilities emerge from such assessment, the company informs the client so that it can decide which software vulnerabilities need to be managed by the company.

B. Theme 2) Instrumentation

We defined three sub-themes, within this main theme, which are related SAST/DAST tools, unit testing, and third-party software.

c) SAST/DAST Tools. The company mainly uses SAST tools for detecting software vulnerabilities within both IDEs (Integrated Development Environments) and CI (Continuous Integration) pipelines.

In particular, SonarLint is plugged into IDEs to detect software vulnerabilities when developers code, while SonarQube is used for scheduled (*e.g.,* weekly) or event-triggered (*e.g.,* merge request) security assessments. Both tools are open-source, developed by *SonarSource*, and share the same analysis engine (*i.e.,* they detect the same software vulnerabilities). The company opted for these tools because they can be considered de-facto standards. Also, they are cost-effective, easy to use, and faster than DAST tools.

Both SonarLint and SonarQube assess software security based on two kinds of security issues: *hotspots* (*i.e.,* security-sensitive code portions requiring reviews to determine whether, or not, to make such portions secure) and *vulnerabilities* (*i.e.,* security issues requiring fixings). The company considers these two kinds of security issues equally important—*i.e.,* what matters for the company is the severity of security issues, not their kind.

The participants asserted that SonarQube is not only used in greenfield projects but also largely used in the assessment of legacy software (*i.e.,* in maintenance projects).

Some clients require the company to use their CI pipelines or different SAST/DAST tools (*e.g., Fortify*, *Security Reviewer*, *CheckMate*, or *OWASP ZAP*). When this happens, the company comply such a request.

d) Unit Testing. The presence of software vulnerabilities is not usually detected by performing unit testing, exceptions are represented by very specific and easy-to-test software vulnerabilities (*e.g., SQL Injection*). However, the company values unit testing since its developers are used to running unit tests after resolving vulnerabilities in order to detect regressions. The company considers critical unit test coverage lower than 80%.

e) Third-party Software. If the client does not impose any constraint, NTT Data uses open-source third-party software (*e.g.,* frameworks or libraries) supported by an active community and widely used by other developers. This is because, in case of software vulnerabilities in such third-party software, the company's developers can find support from other developers. Also, the developers of such third-party software are usually fast in publishing software releases containing patches for newly-discovered software vulnerabilities.

C. Theme 3: Resolution

This theme includes three sub-theme related to the resolution of software vulnerabilities in the codebase and in third-

[2]SonarQube assigns a severity level to each kind of security issue based on *impact* (*i.e.,* to what extent a kind of security issue causes harm to the stakeholders of the software system if exploited by an attacker) and *likelihood* (*i.e.,* what is the probability that an attacker exploits that kind of security concerns). Blocker software vulnerabilities are characterized by a high impact and a high likelihood, while critical software vulnerabilities have a high impact but a low likelihood.

party software, and the resolution of software vulnerabilities without fixing them.

f) Resolution in the Codebase. Software vulnerabilities are usually resolved by the developer who originally wrote the source code containing such software vulnerabilities—exceptions are represented by junior developers, who can ask the help of senior ones to resolve them.

g) Resolution in Third-party Software. If software vulnerabilities are disclosed in third-party software, the resolution process usually consists of updating the third-party software to the releases containing the corresponding patches. The patches are published by the vendors (including open-source organizations) and are often an immediate way to fix these vulnerabilities because a one-click dependency update is most of the time all that is needed. Moreover, the company values unit testing (as mentioned before) so that regressions are likely to be detected after a dependency update.

h) Resolution without Fixing. Some software vulnerabilities are resolved without fixing them. In particular, some are false positives, others have a low severity level and their resolution is not part of the contract, while still others cannot be fixed due to cost and/or technological constraints (*e.g.,* the client needs to use an outdated communication protocol that should be updated to fix a given vulnerability). In any case, the client is informed about these security concerns and may ask NTT Data not to fix them.

V. Discussion

In this section, we first discuss the obtained findings and then the threats that might affect the validity of the obtained results.

A. Overall Discussion, Implications, and Future Research Directions

Below, we use frames to show the main findings of our study. Then, whenever possible, we outline research opportunities for academics and implications for practitioners.

Finding #1: The policy for managing software vulnerabilities, including the application security standards to be guaranteed and the instruments used for software vulnerability management, are established in the contract with the client. Such a policy depends on clients' needs, but also on projects' nature.

The establishment of the application security standards heavily relies on the client's demands, so it is necessary that clients are aware of the security level needed by their software. As software is increasingly pervasive, demanded, and adopted by different categories of people with different backgrounds, it should not be taken for granted that they have such awareness. Future research should seek to understand the degree to which clients are aware of the security level needed and achievable by the software they demand. On the other hand, practitioners should determine if their clients comprehend the security risks of the demanded software.

The above-mentioned finding confirms those from past studies [3], [7], [8]: developers are not averse to using security practices, they simply follow company policies established with clients.

Finally, since much of the software in circulation is legacy nowadays, it would be interesting for both researchers and practitioners to understand the degree of security and the ease with which legacy software can be secured.

Finding #2: Software vulnerabilities are mostly detected by means of SAST tools—*SonarLint* and *SonarQube* appear to be the de-facto standard in NTT Data. DAST tools are less used than SAST tools due to their higher execution costs and inferior ease to use.

SAST and DAST tools may detect different types of software vulnerabilities. Therefore, if software companies do not use both SAST and DAST tools, some software vulnerabilities may be overlooked. Clients need to be aware of the primary role of DAST tools, and then take a more informed decision about asking software companies to use DAST tools in conjunction with SAST tools.

To ease the adoption of DAST tools, researchers should devise strategies to reduce the execution costs of DAST tools.

Finally, the above-mentioned finding explains the increasing researchers' attention on SonarLint and SonarQube by the software engineering research community (*e.g.,* [17]–[21]).

Finding #3: Software vulnerabilities are usually resolved by the developer who originally wrote the source code containing such software vulnerabilities, if they lie in the codebase; or by updating software dependencies, if such software vulnerabilities lie in third-party software. Also, some software vulnerabilities are consciously left without a fix because their resolution is not part of the contract or due to cost and/or technological constraints.

The use of vulnerable software components threatens software security. Practitioners should assess the security of the individual components that make up their software. To that end, they should adopt Software Composition Analysis (SCA) tools to check their software supply chain security, as well as provide their clients with Software Bill of Materials (SBOMs) listing all the components of their software systems along with known software vulnerabilities [22]. Researchers could be interested in investigating the adoption of these practices in both the industrial and open-source contexts.

Finally, the decision of leaving some software vulnerabilities unfixed is driven by the contract or other constraints not attributable to the developers, but to the clients. Once again, we confirm findings from past work (*e.g.,* [3]): developers are not averse to using security practices, they simply follow company policies (agreed with clients).

B. Threats to Validity

Focus groups can be prone to threats associated with qualitative data, like *researcher bias* [23]. Other possible threats to the validity are related to focus group weakness [11]: *group dynamics*, *social acceptability*, *hidden agendas*, and *limited comprehension*. Further threats to validity concern *secrecy*, *false information*, the *number of participants*, and *generalization*. Below, we discuss these threats.

Researcher bias should not be present in our study since we are not part of NTT Data, thus we are neither responsible for managing software vulnerabilities in NTT Data nor we are interested in obtaining a specific outcome from the study. Moreover, two researchers performed the template analysis to avoid their expectancies influencing the results.

To mitigate a threat due to group dynamics, we used a semi-structured discussion approach; also, the facilitator balanced discussions and activated less active participants.

Social acceptability weakness was mitigated by laying out appropriate ground rules in the beginning. The facilitator took his role in driving the discussion to avoid as many as possible social acceptability issues.

Hidden agendas did not affect our study results because it was communicated to participants that the gathered data would be threaded confidentially (including the identity of the participants) and the results would be presented in an anonymous form. Moreover, having initially clarified the exploratory nature of the study and the absence of particular expectations from researchers, the participants were able to express themselves and intervene without pressure and with sincerity. We also emphasized that study results could be significant for both academy and industry.

As for limited comprehension, we approached the topics of interest in a top-down manner, starting with the general and moving to the specific, so that all participants had opportunities to talk.

Regarding secrecy, it does not affect the validity of results because relevant information concerned with proprietary or business reasons was not discussed in our focus group session.

The risk that the participants might profess false information about the company was mitigated by deciding that the paper would be published with the company's name only after they read it and agreed with what transpired.

As for the number of participants in the focus group, we are within the typical range—according to Kontio *et al.* [11], focus groups are typically conducted by involving three to twelve participants.

Finally, we recognize that our results may not hold outside the case of NTT Data (*e.g.,* in the case of other companies or in the open-source context).

VI. Conclusions

We conducted a focus group with six software professionals of NTT Data, a big IT service and consulting company, to understand how software vulnerabilities are managed in the software systems this company develops and/or maintains.

The company's policy for managing software vulnerabilities depends on clients' needs (*i.e.,* according to the context in which the clients operate, they can require higher or lower application security standards to be met) and the projects' nature (*i.e.,* development of greenfield projects vs. maintenance of existing ones). This finding confirms past ones (*e.g.,* [3]): developers are not averse to using security practices, they simply follow company policies (agreed with clients).

The company mainly uses SAST tools, like SonarLint and SonarQube, for detecting software vulnerabilities. These two tools appear to be de-facto standards for NTT Data. This explains the increasing researchers' attention on SonarLint and SonarQube (*e.g.,* [17]–[21]).

DAST tools are less used than SAST tools due to their higher execution costs and inferior ease to use. Clients need to be aware of the primary role of DAST tools, and then take a more informed decision about asking software companies to use DAST tools in conjunction with SAST tools.

Finally, not all software vulnerabilities are fixed. For example, the presence of some software vulnerabilities is tolerated by the clients—once again, developers (and the companies for which they work) are not averse to securing their code. We recognize the management of software vulnerabilities in open-source projects may be different, thus we advise future research on this matter.

References

[1] R. Shirey, "Internet security glossary, version 2," Tech. Rep., 2007.

[2] eWeek. (2020) Heartbleed ssl flaw's true cost will take time to tally. [Online]. Available: http://archive.md/IMt3t

[3] H. Assal and S. Chiasson, "Think secure from the beginning: A survey with software developers," in *Proceedings of CHI Conference on Human Factors in Computing Systems*. ACM, 2019, p. 1–13.

[4] M. Tahaei and K. Vaniea, "A survey on developer-centred security," in *Proceedings of IEEE European Symposium on Security and Privacy Workshops*. IEEE, 2019, pp. 129–138.

[5] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer, 2012.

[6] C. Weir, S. Migues, M. Ware, and L. Williams, "Infiltrating security into development: Exploring the world's largest software security study," in *Proceedings ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 2021, pp. 1326–1336.

[7] L. Braz and A. Bacchelli, "Software security during modern code review: the developer's perspective," in *Proceedings of ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 2022, pp. 810–821.

[8] H. Palombo, A. Z. Tabari, D. Lende, J. Ligatti, and X. Ou, "An ethnographic understanding of software (in) security and a co-creation model to improve secure software development," in *Proceedings of USENIX Conference on Usable Privacy and Security*. USENIX Association, 2020, pp. 12:1–12:16.

[9] R. Widdows, T. A. Hensler, and M. H. Wyncott, "The focus group interview: A method for assessing users' evaluation of library service," *College & Research Libraries*, vol. 52, no. 4, pp. 352–359, 1991.

[10] J. Kontio, J. Bragge, and L. Lehtola, *Guide to Advanced Empirical Software Engineering*. Springer, 2008, ch. The Focus Group Method as an Empirical Tool in Software Engineering, pp. 93–116.

[11] J. Kontio, L. Lehtola, and J. Bragge, "Using the focus group method in software engineering: obtaining practitioner and user experiences," in *Proceedings of International Symposium on Empirical Software Engineering*. IEEE, 2004, pp. 271–280.

[12] L. Lehtola, M. Kauppinen, and S. Kujala, "Requirements prioritization challenges in practice," in *Proceeding of Product Focused Software Process Improvement*. Springer, 2004, pp. 497–508.

[13] G. Scanniello, S. Romano, D. Fucci, B. Turhan, and N. Juristo, “Students’ and professionals’ perceptions of test-driven development: a focus group study,” in *Proceedings of Annual ACM Symposium on Applied Computing*. ACM, 2016, pp. 1422–1427.

[14] N. King, “Using templates in the thematic analysis of text,” *Essential Guide to Qualitative Methods in Organizational Research*, pp. 257–270, 2004.

[15] V. Braun and V. Clarke, “Using thematic analysis in psychology,” *Qualitative Research in Psychology*, vol. 3, pp. 77–101, 01 2006.

[16] S. Romano, D. Fucci, G. Scanniello, M. T. Baldassarre, B. Turhan, and N. Juristo, “On researcher bias in software engineering experiments,” *Journal of Systems and Software*, vol. 182, p. 111068, 2021.

[17] N. Saarimaki, M. T. Baldassarre, V. Lenarduzzi, and S. Romano, “On the accuracy of sonarqube technical debt remediation time,” in *In proceedings of Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2019, pp. 317–324.

[18] M. T. Baldassarre, V. Lenarduzzi, S. Romano, and N. Saarimäki, “On the diffuseness of technical debt items and accuracy of remediation time when using sonarqube,” *Inf. Softw. Technol.*, vol. 128, p. 106377, 2020.

[19] D. Pina, A. Goldman, and C. Seaman, “Sonarlizer xplorer: a tool to mine github projects and identify technical debt items using sonarqube,” in *International Conference on Technical Debt*. IEEE, 2022, pp. 71–75.

[20] G. Digkas, M. Lungu, P. Avgeriou, A. Chatzigeorgiou, and A. Ampatzoglou, “How do developers fix issues and pay back technical debt in the apache ecosystem?” in *Proceedings of IEEE International Conference on Software Analysis, Evolution and Reengineering*. IEEE, 2018, pp. 153–163.

[21] S. Romano, F. Zampetti, M. T. Baldassarre, M. Di Penta, and G. Scanniello, “Do static analysis tools affect software quality when using test-driven development?” in *Proceedings of ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM, 2022, p. 80–91.

[22] US Department of Commerce, “The minimum elements for a software bill of materials (SBOM),” http://archive.md/2023.04.18-062538/https://www.ntia.doc.gov/files/ntia/publications/sbom_minimum_elements_report.pdf, 2021.

[23] J. Langford, *Focus groups: Supporting effective product development*. CRC press, 2002.