Universiteit Leiden

Master ICTiBPS

"Exploring the role, effectiveness and evaluation of non-technical practices in secure software development, currently used in organisations."

Name:	Bruno Sebastian Smith
Student ID:	S3731545
Date:	01/04/2025
1st supervisor:	Olga Gadyatskaya
2nd supervisor:	Nusa Zidaric
3rd supervisor:	Arina Kudriavtseva

Master's Thesis ICT in Business and the Public Sector

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

This research investigated the non-technical practices employed by organisations in secure software development and evaluated the metrics used to assess their effectiveness. The primary aim of the study was to identify the non-technical practices use by organisations, consult applicants on their opinions of these practices and their effectiveness, explore the challenges organisations faced in their implementation and usage, establish how different organisations interact with them and assess how their effectiveness was measured.

A series of semi-structured interviews and exercises were conducted with experienced professionals across a diverse range of organisations, followed by thematic analysis. The findings revealed that while organisations widely recognised the value of non-technical practices in bolstering security, quality, and consistency, their adoption remained inconsistent.

We found that organisations generally favoured customised, agile/Dev Ops development approaches that, though flexible, were prone to inconsistencies and outside pressure.

Practitioners mentioned a wide array of non-technical practices they used or considered indispensable for their SSDL, while highlighting issues with some and disregarding others. Key challenges identified in their adoption included resource constraints, a lack of formal evaluation methods, and difficulties in balancing robust security with productivity.

Organisations at different stages in their lifecycle were found to adopt distinct approaches to implementing non-technical practices, each tailored to the specific challenges and objectives they encountered.

Proper implementation and application of non-technical practices was seen as vital for their success, as even non-technical practices perceived as effective were doubted through experiences with bad implementation.

We also found that non-technical measures were typically mainly evaluated informally and through qualitative methods, making the effectiveness of non-technical practices problematic to determine or justify.

In conclusion, the research underscored that, despite the recognised long-term benefits of non-technical practices in fortifying security and other factors in development, the absence of formal frameworks and structured evaluation methods highlighted the need for more systematic approaches to assess their effectiveness and guide their implementation in order to realise their full potential.

Contents

1 Introduction

Global spending on information security and the development of secure software has reached unprecedented levels, exceeding 175,000 million USD in 2023 [54]. This remarkable figure is expected to continue rising in the coming years as the introduction of IoT (Internet of Things) tools and cloud computing fuel the rapid expansion in the volume of software and data that organisations produce [54]. Consequently, the need for robust protection of these systems has become ever more pressing [54]. As a response, secure software development frameworks have been consistently developed by different organisations, to ensure that security is integrated throughout the entire development process. These frameworks have emerged due to the increasing frequency of cyberattacks and the harmful impact of vulnerabilities resulting from non-secure development practices [7].

Although technical solutions receive significant attention, research highlights that non-technical factors are equally crucial for maintaining effective security management in software development. For example, Alavi, Islam and Mouratidis [2] found that human factors significantly influence security outcomes within organisations, even when security measures are in place. This is underscored by the 2024 Data Breach Investigation Report [60], which revealed that 68% of data breaches in 2024 were linked to non-malicious human factors, such as a person falling victim to a social engineering attack or making an error. Further studies indicate that the majority of software project failures and approximately 80% of workplace failures can be attributed to deficiencies in non-technical skills rather than technical shortcomings [45].

In light of these findings, most Secure Software Development Lifecycle (SSDLC) frameworks have begun to incorporate practices that extend beyond conventional technical measures to combat these issues[28]. Kudriavtseva & Gadyatskaya [28] suggests that these practices, can be categorised into the organisational, behavioural, legal, policy and governance aspects in the development process of software systems and labelled them as auxiliary/non-technical practices.

For instance, SSDLC like the Deloitte 2023 Secure Software Development Lifecycle [8], Microsoft SDL-Agile 2012 [36] and the NIST 800-218 2022 [39] all include non-technical practices to different extents. However, Kudriavtseva & Gadyatskaya [28] found that popular published methodologies still primarily focus on technical practices when looking to add security to their development process. This disregard has been consistent over some time, as early as 2010 Pirzadeh [43] observed that developers frequently overlook or devalue these practices during the later stages of software development. Moreover, Kudriavtseva & Gadyatskaya [28] described the overall impact and efficacy of non-technical security practices as unclear, revealing a significant gap in methodologies justifications of implementation and academic research which warrants further investigation.

Similarly, evidence from Kaminsky [25] indicates that non-technical leadership practices in IT project management are vital to project success, yet these aspects remain under explored. Despite their recognised importance, non-technical/auxiliary practices remain under researched, and their current utilisation and efficiency within the global industry remain largely unknown [28].

1.1 Definition of Effectiveness in this Research:

In this research, *"effectiveness"* is defined as the extent to which the benefits delivered, particularly improvements in security, justify the investment into non-technical practices made by

organisations. Investment may include various forms of resources such as time, money, personnel, and organisational effort. For example, this encompasses the costs associated with staff training, the development and implementation of new security policies and any additional expenses related to process reorganisation. The evaluation examines whether these investments lead to process and security improvements in the perception of the participants, such as a reduction in vulnerabilities, quicker incident response times, and better compliance with security standards, thereby enhancing the overall security posture of the organisations secure software development approach.

1.2 Research Questions (RQs)

Due to the presented issues, opportunities, and gaps in the existing research, we established our main research question.

RQ: "Exploring the role and effectiveness of non-technical/auxiliary practices in secure software development, currently used in organisations."

1.3 Research Sub Questions

Furthermore, the literature review conducted for this thesis and presented in the next chapter led to the formulation of the following sub-research questions:

RQ1: "What is the perception on non-technical practices and their impact on Software development?"
RQ2: "Which non-technical practices are commonly involved in the secure software methodologies and are used in industry?"
RQ3: "Which non-technical practices are perceived as effective and ineffective for practitioners?"
RQ4: "What are the most common challenges faced by organisations when adopting non-technical software practices?"
RQ5: "Why do organisations perceived as having a higher degree of maturity by developers integrate more non-technical practices in their secure software development process?"
RQ6: "What methods are used by organisations to estimate the effectiveness of non-technical practices in the context of developing secure software?"

Non-technical practices, also known as auxiliary practices, are currently not a fixed or established term or category used in academia or industry. Though they may occasionally be referenced by different papers under this title, non-technical practices currently have different definitions as to what they encompass, most often though agree on that they focus on factors such as human and organisational in some sense.
In this thesis, we will follow this thinking and they will be defined as practices that are used to manage the human and organisational factors within an organisation's secure software development process, based on the definition by Kudriavtseva & Gadyatskaya [28].
The research questions (RQs) discussed in this thesis focus on understanding the role, perceived effectiveness and approaches used to measure the effectiveness of non-technical/auxiliary practices in secure software development within organisations. Specifically, the study aims to explore

how these practices are generally defined and perceived in terms of their impact on secure software development (RQ1). It also seeks to identify which non-technical practices are commonly used (RQ2) and whether practitioners view them as effective or ineffective (RQ3). Additionally, the research examines the challenges organisations face when implementing and applying these practices (RQ4) and whether organisations with a higher perceived maturity level of development tend to integrate more non-technical practices (RQ5). Finally, it investigates how organisations measure and justify the effectiveness of these practices (RQ6).

1.4 Contributions

This thesis explores non-technical practices within secure software development as implemented by current organisations. It provides a comprehensive understanding of how these practices are applied in practice, how their effectiveness is perceived, and how their impact is measured. The findings offer diverse applications, including actionable recommendations and potential policy guidelines for development teams and organisations.

The study contributes to the ICT in business and cybersecurity knowledge base by identifying shortcomings in the selection and application of these practices, updating best practices based on new insights, and inspiring future research. It offers valuable insights into how organisations, categorised by maturity level and size, perceive and adopt non-technical practices, while highlighting the common challenges and opportunities faced.

By adopting non-technical practices proven effective in real-world settings, developers can build securer software, managers can better prioritise security efforts, academics can understand and strengthen theoretical frameworks and users ultimately benefit from safer systems. The thesis also presents a detailed overview of current adoption patterns and perceived effectiveness, contrasting real-world practice with theoretical perspectives and methodologies. Furthermore, it introduces benchmarks on non-technical practices, approaches to support process refinement and address gaps between research and practice.

In summary, this research fills a significant gap in secure software engineering literature and has practical implications for both organisations and regulators, contributing to the advancement of stronger security in software development.

1.5 Outline of the thesis

This thesis is organised into several sections to provide a clear and coherent account of the research conducted. Chapter one, found above, introduces the study, outlining its background, research problem, aims, significance and contributions. Chapter two presents a review of the relevant literature, situating the study within existing research and identifying gaps that this thesis seeks to address.

Chapter three explains the research methodology, justifying the qualitative approach and detailing the data collection methods, particularly the use of interviews and the sample demographic, as well as the analytical framework employed. Chapter four presents the findings/results, offering the correlated thematic insights derived from the interview data. Chapter five discusses these findings in relation to the literature, highlighting key differences, similarities and implications.

Finally, chapter six concludes the study, summarising the main findings, acknowledging limitations, and suggesting directions for future research, as well as providing recommendations to organisations. Following this, are the appendix and reference list where related material can be

found. This structure aims to ensure a logical flow of information, guiding the reader through the research process and its outcomes.

2 Literature Review

In conducting this literature review, we employed Google Scholar to identify relevant academic papers, while Google Search was utilised to locate relevant grey literature, such as blogs, white papers, reports, and government documents. Due to the limited number of sources found, a "snowballing approach" was used as one of the main methods to find papers relevant on the topic.

2.1 Keywords:

(SSDLC) Secure software development lifecycle, (SSDM) Secure software development methodology, Methodology, non-technical practices, Secure software, Effectiveness

2.2 Search terms:

To search for secure software development methodologies, we utilised the following search strings:

- *(SSDLC OR SSDM) AND (Methodology OR Framework OR Model OR Standard OR Lifecycle) AND ("Non-Technical Practices" OR "Non-Technical Aspects")*
- *(SSDLC OR SSDM) AND (Methodology) AND (Effectiveness OR Efficiency) AND ("Best Practices" OR "Optimisation")*
- *(SSDLC OR SSDM) AND ("Secure Software" OR "Security Software") AND (Methodology OR Framework OR Model OR Standard OR Lifecycle)*
- *("Methodology") AND (SSDLC OR SSDM) AND ("Non-Technical Practices" OR "Non-Technical Aspects") AND (Effectiveness OR Efficiency)*
- *(SSDLC OR SSDM) AND (Effectiveness OR Efficiency) AND ("Non-Technical Practices" OR "Non-Technical Aspects")*

2.3 Literature Review Structure

In this section, we will discuss the following relevant themes in order to create a foundation of understanding for the findings later discussed in the thesis. The literature review will include these themes and be structured in the following way:
To provide a strong foundation for this literature review, we begin with an overview of *Software Development Methodologies*. This establishes the necessary context for understanding how non-technical practices fit within the broader development ecosystem of organisations.
Following this, we examine Human Factors in Secure Software Development Life Cycles. Previous research, such as that by Kudriavtseva & Gadyatskaya [28], suggests that non-technical practices are primarily used to manage human and organisational factors in secure software development. To understand the role of these practices, it is first essential to explore "Human Factors in SDLC's" they aim to address.
Continuing this trend, we then review "Organisational Factors in Secure Software Development Life Cycles. Having established the human aspects that non-technical practices seek to manage,

it is equally important to examine the organisational factors within the SSDLC's that these practices influence.
With a clearer understanding of the factors being managed, we move on to the "Definition of Non-Technical Practices in Secure Software Development". A precise definition is crucial to ensuring clarity in the research scope and establishing what, according to existing literature, constitutes non-technical practices in this context.
Given that security is a key priority in secure software development, we next explore "The Role of Non-Technical Practices in Enhancing Security in the Software Development Lifecycle". By reviewing previous research, this section aims to understand how and if non-technical practices have been shown to contribute to secure development.
As previously mentioned, despite their advantages non-technical practices are not always widely adopted. Therefore, we examined the "Barriers to the Adoption of Non-Technical Practices by Organisations". Understanding these challenges will help identify the common barriers and whether the same barriers persist in different organisational contexts.
We then turn to the "Influence of Organisational Maturity on Non-Technical Practice Usage and Adoption". Organisations at different maturity levels approach security in varying ways, and this section reviews methodologies previously used to examine how maturity impacts the adoption of non-technical practices in this context.
Finally, to assess the real impact of these practices, we look at approaches to "Measuring the Effectiveness of Non-Technical Practices". Without clear evaluation metrics, their benefits remain uncertain, so this section explores existing approaches for assessing and justifying their effectiveness in improving software security.

2.4 Literature Analysis

2.4.1 Overview of Secure Software Development Methodologies

The rapid advancement of the digital age and the growing reliance on technology have underscored the importance of security as a critical component of software quality [27]. In response, there has been a continuous rise in Secure Software Development Methodologies (SSDMs) since at least 2004. These methodologies aim to embed security into the development processes that guide modern software engineering [28]. Secure Software Development (SSD) is defined as the practice of integrating security at all stages of the Software Development Life Cycle (SDLC), ensuring a holistic approach to improving the overall security of software products [27].
The push toward SSDMs stems from the significant risks associated with vulnerabilities in software systems, particularly as these systems grow more interconnected and complex [28]. Security weaknesses in critical environments are substantial risks and can lead to significant damages, making robust security practices an essential focus of development [26]. These threats can arise at any phase of the SDLC, from inadvertent errors to malicious acts by insiders or external agents [27]. To address these vulnerabilities, organisations have increasingly prioritised integrating security measures early and maintaining them continuously throughout the development process [28]. This approach has been shown to reduce both the number of vulnerabilities and the cost of fixing them as software becomes more complex over time [28].
SSDMs typically follow a set of established phases, including project inception or planning, analysis and requirements gathering, architectural and detailed design, implementation, verification and testing, release and maintenance, and, finally, disposal [28]. These methodologies

have been developed by academic, industrial, and governmental organisations worldwide, reflecting diverse approaches to incorporating security into software development. For instance, SSDMs may be based on sequential models, like waterfall, or iterative approaches, such as agile [28]. Despite existing models, many organisations prefer to adopt custom frameworks or select specific security activities that suit their unique needs, rather than adhering to formal SSDMs [63].
Developers also often express dissatisfaction with having to work with hybrid models that attempt to merge methodologies, with criticisms labelling such approaches as "waterfall with extra meetings," "fake cargo cult agile," or "Resume-Driven Development (RDD)" [46]. These sentiments highlight the gap between theoretical frameworks and practical application in real-world settings. In their study [27] also found that "software security often fails because their development is generally based on ad-hoc foundations or follow traditional development processes".
It should be noted that there is a fine line between projects that adopt a custom framework versus those who do not use a framework but select specific security activities [63].
And as KHAN et al. [27] observe, "Despite all of the efforts, software that offers high standards of security integrity is uncommon." This critique reflects the ongoing need for research to refine SSDMs and improve their implementation. Bridging the gap between methodology and practice remains a critical challenge for the field, emphasising the need for continuous innovation and evaluation.

2.4.2 Human Factors in the SDLC: Human Factors in Secure Software Development

The literature on secure software development highlights the critical role that human and social factors play in influencing software engineering, the Software Development Life Cycle (SDLC), and the practices embedded within it [24]. This perspective is widely shared among researchers, who emphasise that software product development is fundamentally an activity rooted in social and human dynamics [34]. As John et al. [24] state, "Software is developed for people and by people," underscoring the central role of human elements in software engineering. Similarly, Restrepo-Tamayo and Gasca-Hurtado [44] describe software development as "an intellectually challenging activity that demands collaborative work," emphasising its predominantly social and people-centred nature.
Human factors profoundly influence how critical practices are adopted, applied, and maintained throughout the SDLC [44]. Therefore, any framework aimed at improving an organisations software development process must consider these factors. Restrepo-Tamayo and Gasca-Hurtado [44] identify 13 key human factors that affect software development, linking aspects such as communication to project performance and success, motivation to job satisfaction, and interpersonal relationships to software quality, team knowledge acquisition, and value creation. Their findings illustrate the wide range of human factors at play in software development and their significant impact on software project success. Properly addressing these aspects can enhance security, developer productivity, improve product quality, and boost customer and user satisfaction [24].
However, the strong influence of human factors on software development also necessitates acknowledging the potential for human error. In a review of a secure coding competition, vulnerabilities stemming from misunderstandings of security concepts were the most common issue, appearing in 78% of the projects analysed [61]. As mentioned in the introduction,

the 2024 Data Breach Investigation Report by Verizon [60], it was found that 68% of data breaches involved human error, the report finding that people continue to play a very large role in incidents and breaches of systems alike. Additional, in 2022 the World Economic Forum in their Global Risk Report stated that 95% of cybersecurity issues could be traced back to human error, with many more examples such as this existing [64]. This shows that, while well-managed human factors can positively contribute to development and its security, not doing so can also present risks and challenges. For instance, a simple lack of motivation has been found to prevent developers from prioritising security, which is important as additionally not all developers are themselves intrinsically motivated to integrate security measures during development from the beginning [37]. Security may fail to capture developers interest, leading to insufficient attention being paid to it during the steps conducted in their organisations SDLC during development, which can lead to vulnerabilities [37].

2.4.3 Organisational factors in the SDLC: Organisational factors in Secure Software Development

As mentioned, human factors are not the only non-technical factors relevant to the successful development of secure, high-quality software. As put by Sommerville [53] "Secure software development is influenced by a complex interplay of technical, human, social, and organisational factors". This was supported by Tuape [57] in his research of small software companies, who stated that organisational, governance, and business environment factors majorly affect the software development process. Other researchers also share this sentiment, highlighting the interplay between cognitive and organisational factors, such as work organisation, expertise, and tool use, and their importance in software development [62]. For example, organisational factors like familiarity, physical proximity, meeting settings, and number of participants of said meetings are found to affect communication effort in software development projects [47]. Shehzada and Kausarb [48] found twelve organisational factors that affect successful agile projects, such as knowledge management, organisational culture or management commitment, once again highlighting the impact of such organisational factors.
Organisational factors are found to have strong correlation to the quality and success of secure software development and have a significantly influence on the success of information systems development projects [1]. As early as 1998, research shows that developers seem to also be aware of this [11]. As in a survey of IT professionals by Doherty [11], the majority believed organisational issues are more important than technical issues in determining the success or failure of information systems development projects. This can also be seen as organisational metrics, developed in a study by Nagappan et. al [38], outperformed traditional software quality metrics in predicting failure-prone software components. Their study provided empirical evidence supporting the claim that organisational structure affects software quality [38]. Similarly, in a case study at the Engineering Division of Software Development Organisation in Sri Lanka, Knowledge sharing, staff training, and coworker relationships were found to be key organisational factors that affected software quality during development[19].
Other organisational factors such as authoritarian leadership styles, functional structures, and a resistant culture were also found to negatively impact the success of software development projects [22]. A simplification of this is a statement from Bjarne Stroustrup, the creator of C++, stating that "An organisation that treats its programmers as morons will soon have programmers that are willing and able to act like morons only"[55]. Other research mirrored such findings, stating that organisational factors such as structure and culture can negatively

impact software quality by pressuring developers to make decisions that compromise code quality [32]. Similar studies found that organisational factors were critical for successful software process improvement, explaining over 50% of the variance towards the original aims [12].
As development teams become increasingly global and culturally diverse, new organisational challenges emerge, including misaligned cultural expectations, communication gaps, language barriers and other non-technical issues [24]. These challenges underline the importance of managing non-technical skills such as communication, emotional intelligence, and leadership in IT-related roles to combat such issues [9]. Studying these factors is particularly valuable for software organisations seeking to foster team productivity and improve secure development outcomes [34].

2.4.4 Definition of Non-Technical Practices in Secure Software Development

As the significance of the human and organisational factors in software development have been recognised by researchers and organisations, they have looked to implement practices that manage its flaws and promote its benefits in the name of better security, productivity, and quality.
Though not an officially coined phrase, the practices to manage such factors have been referred to as "non-technical" or "auxiliary" practices in parts of the secure software development community [28].
Non-technical practices are incorporated in most of the published and peer-reviewed SSDMs developed by academic, industrial, and governmental organisations. This is evident in the diversity of the 28 secure development methodologies analysed by Kudriavtseva & Gadyatskaya [28], all of which include non-technical practices to some extent, regardless of their development approach, whether Waterfall, Agile, or Dev Ops.
Non-technical skills and practices as a category can be interpreted in various ways, leading to some ambiguity about what the term encompasses due to its broad use across different fields. Commonly however, it refers to skills or practices related to the human dimension of a project [15]. For instance, in the field of medicine, the Royal College of Surgeons of Edinburgh defines non-technical skills in categories such as situational awareness, decision-making, communication and teamwork, and leadership [56]. In the Deepwater Horizon oil platform incident investigation, the Chemical Safety Board USA called for the development and use of non-technical skills like communication, teamwork, and decision-making to prevent similar incidents [15]. In software development, Restrepo-Tamayo and Gasca-Hurtado [44] classified non-technical skills as encompassing communication, emotional intelligence, or leadership, which are known to be essential in IT-related roles while Lacher et al. [30] defined non-technical skills as cognitive, personal, and social abilities that enhance technical skills and contribute to overall performance. This shows that the definition of non-technical skills varies across fields while retaining core commonalities, even across different industries.
Although not uniformly defined, non-technical skills and practices in different fields often overlap and can also be referred to by other terms, such as "auxiliary", "soft" or "people skills" and more [35]. These are prevalent in many fields, including being a major part of secure software development [35]. However, it is important to note that non-technical skills, while related, are not synonymous with only human factors in most fields [15].
For this thesis, the definition of non-technical practices provided by Kudriavtseva & Gadyatskaya [28] will be adopted. They define non-technical practices as an overarching designation of those practices used to manage human and organisational aspects of secure software

development.
Their study identified these practices as encompassing organisational, behavioural, legal, policy, and governance aspects. These practices are not limited to specific stages of development but are instead organisation-wide or span all commonly agreed on stages of the SDLC [28].
The idea and necessity of such organisation-wide security practices spanning the whole of a development process is supported by Khan et al. [27], who found that software security is now a primary need for secure software development (SSD) at every phase of the SDLC.
From their evaluation of prevalent methodologies, Kudriavtseva & Gadyatskaya [28] established nine overlapping categories of non-technical practices spreading over all stages of secure software development methodologies: Risk Management, Security Metrics/Measurement, Building a Culture of Security, Understanding Human Behaviour, Policies, Strategies, Standards, and Conventions, Auxiliary Practices of Incident or Vulnerability Response, Communication Process and Customer Responsibilities, Ethics and Privacy. These categorisations, though not all encompassing, provide a holistic account that can be used as a basis for and will be used in this thesis as a framework to outline the place and differentiation of non-technical practices in secure software development.
The literary findings presented above, emphasise the significant influence of non-technical practices on all individual stages of the SSDLC, showcasing the recognition of their integral role in addressing the human, organisational and other dimensions of secure software development, in spite of the differences of what they encompass.

2.4.5 The Role of Non-Technical Practices in Enhancing Security in the Software Development Lifecycle

As mentioned in the previous section, non-technical practices are essential for improving software security within the Secure Software Development Life Cycle (SSDLC). While technical tools and frameworks have advanced to address security vulnerabilities, these tools can only be effective if developers are motivated and guided to use them. Non-technical factors such as motivation play a crucial role in whether developers engage with security practices, and non-technical practices are key in fostering such motivation. Research shows that even the best-documented security tools are ineffective if developers fail to apply such security practices in the development process [61]. Non-technical practices, such as addressing human, organisational, and cultural factors, help create an environment where security practices are applied consistently, ultimately improving the software's security posture, showing the connection between modern technical and non-technical practices.
Developers may also be deterred from implementing security features and practices by other factors such as competing priorities, lack of resources, or insufficient security knowledge [5]. Additionally, developers' perceptions such as seeing security as irrelevant or obstructive can also reduce their engagement with security practices, reducing overall security [37]. Non-technical practices like awareness programs, clear task definitions, and leadership-driven security initiatives help reshape these perceptions, making security more relevant and aligned with business goals.
A strong organisational security culture is particularly influential in moving developers to apply security measures, Arizon-Peretz et al. [4] argue that a company that emphasises security and provides clear expectations can empower developers to take proactive steps in securing their code. Conversely, a weak security culture can lead to neglect, underscoring the importance of non-technical practices such as establishing a security-first mindset and offering consistent

training.
Clear security policies and standards are critical non-technical practices that guide developers in applying security measures. Mokhberi and Beznosov [37] found that policies requiring developers to use security tools can significantly improve code security. These policies institutionalise security, ensuring that it is consistently applied throughout the development process. Similarly, Xiao et al. [65] found that mandatory use of security tools ensures security is embedded in the development lifecycle, rather than left to individual discretion.
In addition to such policies, practices such as code reviews and adherence to secure coding standards also enhance security during development. Zahan [66] showed that manual code reviews were pivotal in identifying vulnerabilities, while structured review processes systematically reduced security risks. This is a strong example of how non-technical practices directly contribute to development security by establishing systematic cross developer reviews and promoting the sharing of best practices between developers.
Training programs and knowledge-sharing initiatives are also key non-technical practices that enhance developers' ability to identify and address security risks. Zahan [66] found that organisations prioritising training and knowledge sharing had lower vulnerability rates than their counterparts. These practices equip developers with the tools and knowledge to address emerging threats, creating a security-aware workforce armed to deal with such issues.
Jiang et al. [23] also noted that organisational support activities such as training, quality assurance, and process engineering are linked to better security outcomes of development projects. A supportive development environment was another non-technical factor that helps both security and overall project performance by ensuring developers are prepared and capable of handling security challenges through the provided support [23].
As mentioned previously, human code reviews are an effective non-technical practice for improving security. Apvrille and Pourzandi [3] showed that peer reviews by developers significantly reduces bug rates and identified vulnerabilities early into the development process. This collaborative process fosters shared responsibility for security and allows developers to learn from each other [3].
Another example of non-technical practices is pair programming, where two or more developers work together on the same code, improving security and quality through shared knowledge. It encourages communication and ensures coding standards and security practices are aligned [51]. Such collaborative non-technical efforts foster both personal accountability and team-wide commitment to security.
The culture within an organisation greatly influences how security practices are adopted [28]. SAFECode [50] emphasised that organisational culture determines whether security is prioritised or neglected. A strong security culture encourages developers to view security as integral to their role. Non-technical practices, such as leadership involvement, clear communication, and rewards for security contributions, reinforce this culture and motivate developers to adopt secure development practices. [22]
Mokhberi and Beznosov [37] also noted that a company's security culture is one of the most influential factors in encouraging the use of security tools. When developers see that security aligns with business goals, they are more likely to engage with security practices, improving the overall security of the software.

2.4.6 Barriers to the Adoption of Non-Technical Practices by Organisations

The adoption of non-technical practices in secure software development is increasingly recognised as critical for project outcomes, as demonstrated by Wee [63], who found this as many companies are now taking cybersecurity more seriously. However, despite this recognition, numerous barriers hinder the widespread adoption of these practices by said organisations.
Research by Mokhberi and Beznosov [37] identified multiple factors preventing developers from implementing security in secure software engineering (SSE) environments. Competing priorities, insufficient organisational support, and the absence of clear guidelines of government regulations and organisational policies are some of the factors which often hinder adoption. Similarly, resource constraints such as limited budgets, time, and personnel add to such problems [37].
Singer [51] observed that software engineering practices are not always adopted by developers, even when mandated by their organisations. A documented process does not guarantee compliance, as implementation depends heavily on non-technical factors such as organisational support and the prioritisation of security [29]. Gartner in 2023 found in a survey that 74% of employees would be willing to bypass cybersecurity guidance if it helped their team achieve a business objective [17]. Without strong institutional backing and a defined plan, developers are often discouraged from adopting and using secure practices into their development process, making security a secondary concern to other more immediate issues.
Adopting non-technical practices requires significant changes to organisational culture, social patterns, and developer behaviours. Nikitina & Mattsson [40] found that introducing software methods often demands adjustments not just in processes but also in cultural norms and stakeholder behaviours to be effective. Singer [51] further emphasised this by finding that developers must often change their current behaviours when new practices are introduced to their development process, ranging from minor adjustments to more drastic shifts.
The introduction of obligatory practices can also face resistance from developers if critical non-technical factors like social pressure, perceived usefulness, and compatibility with existing workflows are not addressed [51]. A lack of proper social integration of these practices can lead to non-compliance, undermining the benefits of secure software development methodologies (SSDMs) and practices. Mokhberi and Beznosov [37] echoed these concerns, highlighting that inadequate planning, resource shortages, and insufficient knowledge discourage developers from engaging with security practices.
To overcome these barriers, Singer [51] suggested non-technical approaches such as gamification and other techniques to enhance the perceived importance and contribution of secure development practices. Incorporating non-technical factors such as social influence, motivation, and persuasion into the implementation of practices can foster greater adoption rates and mitigate developer resistance [51].
Resource constraints are a significant barrier to adopting non-technical practices. Wee [63] noted that organisations frequently fail to provide the time, budget, staff, or guidance required for proper implementation of secure software development methodologies and their practices. Mokhberi and Beznosov [37] also identified specific results of resource shortages, such as a lack of knowledgeable security experts, external penetration testers, and adequate security tools, as critical obstacles to practice and methodology adoption.
In organisations where security is treated as a secondary concern due to tight deadlines, limited budgets, or the absence of rewards or incentives, developers are less likely to invest the necessary time and effort into secure practices [37]. Without applying sufficient resources, the practical

adoption of secure practices remains a challenge despite their recognised importance [37].

2.4.7 Influence of Organisational Maturity on Non-Technical Practice Usage and Adoption

In a study of developers by Fontana et. al. [16], they found that higher agile development maturity correlated towards fostering more non-technical capabilities, such as collaboration, communication, commitment, care, sharing and self-organisation. This shows that organisational maturity plays a significant role in shaping the adoption and effectiveness of non-technical practices to manage these named factors in organisations secure software development processes.
While company size is not a direct determinant of maturity, larger companies often demonstrate greater indicators of maturity due to their need to establish structured processes, often from experience, as they expand and gain market share. An example for this are the findings of Lester et. al. [33] who found that due to the increase in size by organisations the number of communication paths increases and communication becomes more of an issue, this in turn requiring more formalised practices to counteract this.
Several frameworks exist to assess and improve organisational maturity of software development approaches and security practices, the most common being:

- **OWASP Software Assurance Maturity Model (SAMM)**: An open framework helping organisations identify risks and craft strategies to mitigate them [41].
- **Building Security in Maturity Model (BSIMM)**: A model offering standardised terms and practices to guide software security initiatives [6].
- **Cybersecurity Maturity Model 2.0 (CMMC 2.0)**: A framework for assessing and improving the cybersecurity practices of organisations, emphasising scalable maturity levels to ensure sensitive data protection across diverse operational contexts [59].
- **Software Capability Maturity Model (CMM)**: A framework organising evolutionary steps into five maturity levels, building a foundation for continuous process improvement [42].

Mature organisations are more likely to recognise the need/value of addressing human, social and organisational factors in their development processes,. This can be seen in the change of practice types recommended for adoption at higher CMM stages (Stages: 2,3,4,5) [42] who start focusing on managing such aspects to further improve the development process.
By aligning their practices with business needs and fostering a culture of continuous improvement, these organisations ensure that such implemented practices remain effective and competitive [24]. Moreover, higher levels of maturity are seen to mitigate the risks associated with variability in developer capabilities. According to Lavallée and Robillard [31], formalised processes reduce the impact of less proficient developers on project outcomes, creating a buffer that maintains product quality.
The level of an organisation's maturity also influences its security culture. Research by Mokhberi and Beznosov [37] highlights that larger, more mature companies tend to cultivate stronger security cultures. These organisations implement thorough secure software engineering (SSE) and testing processes while fostering an environment where developers view security as a shared responsibility.

Conversely, smaller organisations often demonstrate less mature approaches to security. Developers in these environments face significant challenges, including limited secure coding knowledge and inadequate strategic planning to address competing priorities. These factors lead to poorer security outcomes, with misunderstandings and miss-implementations causing vulnerabilities during the development [37]. Additionally, smaller companies typically lack the resources necessary for effective security implementation. Developers often rely on external sources such as social networks or search engines for guidance, while larger organisations benefit from access to internal security experts who provide more robust support systems [37].
A survey of 154 experienced software project developers confirmed that software process management maturity, as measured by frameworks like the CMM, positively correlates with improved project outcomes [23]. However, not all maturity levels yield observable benefits, suggesting that careful planning and implementation of maturity-related activities are essential [23]. Jiang et al. [23] also noted that while the benefits of increased maturity are evident, achieving higher levels requires significant investment in terms of time and resources, which may not be available to smaller organisations.

2.4.8 Measuring Effectiveness of Non-Technical Practices

The effectiveness of non-technical practices in secure software development remains a challenging area for measurement and validation. Kudriavtseva & Gadyatskaya [28] identified significant gaps in secure development methodologies when it comes to providing evidence for their effectiveness, highlighting a critical need for improvement in software security measurement practices. Similarly, Ita Ryan [46] noted the lack of consensus on universally accepted metrics for evaluating software lifecycle security, which has resulted in difficulties in comparing and estimating the impact of various practices.
A review of methodologies from government and industry further underscores this challenge, as no conclusive evidence of effective measurement practices was found in the 26 prominent ones studied by Kudriavtseva & Gadyatskaya [28]. While assessing the security of software developed in controlled environments is often feasible and repeatable, measuring the implementation of secure development practices in organisational and open-source contexts presents significant difficulties [46].
The complexity of determining whether software is genuinely secure often leads to a focus on evaluating lists of security activities instead of achieving broader security goals [46]. This tendency is particularly evident where compliance-driven practices foster a "box-ticking" approach to security, rather than a culture of meaningful improvement [46]. Consequently, it is imperative to develop methods that evaluate not only the implementation but also the effectiveness of these practices in achieving desired security outcomes.
Research by Votipka et al. [61] further emphasises the limitations of current approaches. Even after participating in a security focused development competition designed to enhance secure coding skills, many teams failed to implement some critical security measures correctly. This highlights that simply introducing non-technical practices, such as security education, is insufficient; there must be robust mechanisms to measure their impact on security outcomes.
Non-technical factors, such as organisational culture and human behaviours, are inherently difficult to measure due to their qualitative nature. Restrepo-Tamayo and Gasca-Hurtado [44] pointed out that the influence of these factors on security practices remains poorly understood and requires further study.
Human factors and organisational dynamics significantly influence the success of non-technical

practices. These aspects are often measured indirectly through surveys, questionnaires, and interviews, as their latent nature makes direct measurement challenging [44]. However, the success of measurement initiatives depends on strong management and stakeholder commitment, both during the initial scoping and throughout the ongoing implementation of measurement and analysis processes [29].

Similarly, Ita Ryan [46] noted that measuring individual secure development practices is fraught with challenges, as debates continue regarding the feasibility and standardisation of such metrics.

A related concern is the complexity of verifying whether secure coding practices are accurately reported or if they yield secure code outcomes at all. As Ita Ryan [46] observed, measuring secure coding efforts is complicated by the lack of consensus on what constitutes success in these initiatives.

Despite these challenges, structured methodologies like SecEval [49] attempt to provide a foundation for assessing security practices within the SDLC by evaluating tools, notations, vulnerabilities, and threats. Establishing measurable indicators of security is viewed as a critical step in creating effective software security metrics, though further discourse on their relevance and validity is essential [24].

Case studies have also been employed in academia to evaluate methodologies, yet they remain underutilised. Kudriavtseva & Gadyatskaya [28] found only one in eight case studies among the 26 methodologies reviewed provided robust justification for their approaches, indicating the need for more empirical research to validate these practices.

3 Research Methodology

3.1 Methodology introduction

This section will look to introduce the methodology applied in this thesis to answer its research aims of: **"Exploring the role and effectiveness of non-technical/auxiliary practices in secure software development, currently used in organisations"** and its sub research questions. It looks to portray and justify the approach and design taken to data collection methods, sample selection, data evaluation methods, ethical concerns and any potential impact on the field of the research it was conducted in.

3.2 Research approach

Qualitative research [10] was chosen as an approach to answer the research questions of this thesis due to the objective and exploratory nature [10] of the topic, which is largely influenced by the interpretation of practitioners of these relevant practices and their perceived effectiveness. Using deductive reasoning, we attempted to create a picture of the current perceptions held by participants on the topic of non-technical practices, which were aimed at being representative of the current state of affairs in relevant organisations software development processes.

3.3 Research design

For this thesis, the data collection was conducted in the form of semi-structured interviews, in order to create a balance between consistency and flexibility, enabling a deeper exploration of non-technical practices in secure software development [10]. This was done due to the ambiguity of the topic, stemming in part from the lack of fixed academic definitions in software development for non-technical practices and a lack of literature on them as a practice sub-group. Due to this, semi-structured interviews were central to this researches design, offering the adaptability needed to investigate the complexity and nuance needed to bridge practitioners different perspectives to non-technical practices.
The interviews were conducted in person or using online services, based on participant preferences, and were subsequently recorded with consent for transcription and analysis. The questions for these semi-structured interviews, were informed by a thorough literature review of prior studies in related fields, in order to address knowledge gaps and were later adapted based on experience and feedback from prior interviews. This literature review was also included in the thesis, in order to provide critical context, examining secure software development methodologies and non-technical practices, to help frame the results collected in order to ensure comprehension and relevance.
To complement the qualitative findings, quantitative methods [52] were used to evaluate exercises conducted with participants, providing measurable insights to enhance the data analysis. Alternative designs, such as experimental or purely quantitative approaches, were deemed unsuitable due to the exploratory nature of the study and were there for not applied.

3.4 Interview questions

The questions used to guide the semi-structured interviews were reviewed by local academics specialising in cybersecurity and secure software development to ensure relevance, they can be

found in the appendix under 8.1 Questions. Based on their feedback, as well as insights from initial interviews, the questions were refined to optimise clarity and effectiveness in relevant data collection.
Additional questions concerning their demographics aimed to add context to participants' responses by gathering information about their expertise, roles, development experience, and their employers' maturity in both organisational and software development practices. This contextual information provided deeper insights into the factors influencing implementation or lack of these practices within their organisations and were used in the thesis.
The complete set of semi-structured interview questions and their alignment with the research questions is detailed in Figure (6): RQ alignment and is presented in the appendix. Two exercises were also incorporated into the study: Exercise One, referred to as "Miro" named after the software used to design it, and Exercise Two, referred to as "Rate the Practice" reflecting its purpose.

3.5 Sample/Participants:

This research aimed to engage professionals with practical experience in secure software development, including roles spanning from developers to senior executives such as CEOs, CTOs, and other key positions influencing organisations development approaches, such as developers and development team leads (Table 1). In total, eleven interviews were conducted for the study.
The participants were situated in Europe and North America and held varying educational backgrounds, including both Bachelors (BSc) and Masters (MSc) degrees in study relevant fields (Table 1). They had between 5 to 15 years of pure practical development experience in organisational settings (Table 1). In this case, employment in the software field was essential, and preference was given to those in leadership roles, as they typically possess greater knowledge of and influence on their organisations development methodologies and processes.
Participants were mainly selected through direct and purposeful sampling to target those with relevant expertise in the field and snowball sampling to identify additional participants through referrals [10]. The goal was to interview participants that could provide insights on trends, similarities, and factors influencing the use of non-technical practices in organisations, and the measurement of their effectiveness in these.
The sample included a diverse group of industry professionals, such as Developers, Development Team Leads, Department leads, CTOs, and CEOs involved in the software development process as seen in Table 1.
Rather than focusing solely on job titles, participants were chosen based on their practical experience and the relevance of their roles to the research topic. We aimed to include individuals from a variety of company sizes, maturity stages, and organisational structures in order to provide a broad perspective of the current application of non-technical practices.
The participants came from a range of organisations, varying in size, maturity, and market focus (Table 1). The participants were from a range of different countries and their organisations operated on local, national and International levels (Table 1.). Insights were shared from multiple development projects, reflecting diverse organisational contexts and development challenges. The study included participants from a diverse set of industries, including:

- Media and streaming services
- Communications provider

ID	Role	Education	Experience (Years)	Country	Organisation size	Company sector	Market	Operations
P0	Developer	BsC	5	Dutch	Large	IT Consulting	Established	International
P1	Development Team Lead	MsC	16	US	Large	Media and Streaming	Market leader	International
P2	CEO	BsC	11	German	Small/ Medium	Software Development	Established	Local
P3	Department Lead	BsC	7	US	Large	Media and Streaming	Market leader	International
P4	CTO	MsC	9	German	Medium	SaaS	Startup	National
P5	Developer	BsC	7	UK	Large	Energy	Established	National
P6	Developer	MsC	7	German	Large	Communications	Established	National
P7	Developer	BsC	5	German	Large	IT Consulting	Established	International
P8	Developer	BsC	6	Dutch	Small/ Medium	Cyber security	Established	National
P9	Developer	MsC	7	German	Large	Aerospace	Established	National
P10	CEO	MsC	5	German	Small/ Medium	Software Development	Established	National

Table 1: Participant Demographics

- IT Consulting
- SaaS (Software as a Service)
- Software development contractors
- Energy provider
- Aerospace
- Cyber security

Additionally, the organisations had different sizes (small, medium, large) and occupied various Market positions in their industries (Table 1) such as:

- Startups – Newly founded organisations focused on innovation and rapid growth, often operating with limited resources and high uncertainty.
- Established Companies – Well-developed organisations with stable operations, structured processes, and a proven market presence.
- Market Leaders – Organisations within an industry that set trends, influence competition, and hold the largest market share.

3.6 Interview process

The interviews were semi-structured, focusing on specific themes related to development methodology usage, the implementation of non-technical practices, and the measurement of their effectiveness. Open-ended questions guided the conversation, allowing for in-depth responses and insights. The interviews varied in duration, lasting between 35 to 85 minutes each.
Audio/Video recordings were used to create matching transcripts, who served as the main data sources, complemented by field notes to capture additional contextual information such as

recurring themes, patterns, and other relevant contextual factors. Transcription was performed using tools such as MS Word and MS Teams, although all transcripts were manually edited for increased usability.
The interviews were conducted between **April 1, 2024, and December 1, 2024**, during which time a variety of perspectives were gathered and analysed.

3.7 Data to be analysed

The primary data collected and analysis in this study was qualitative, with some potential quantitative insights emerging during the exercises and following in the evaluation process.
The data collection process aimed to explore the current use and perceived effectiveness of non-technical practices in organisations, as reported by professionals, and compare these findings to insights from the literature. By doing so, the thesis sought to evaluate whether established non-technical practices identified in the literature are used in practice, are evaluated for effectiveness, and are understood within organisational contexts. Furthermore, the study examined the foundations of these perceptions and their alignment with organisational needs. One key focus was to identify whether practices discussed in the literature are overlooked in organisational settings or if their effectiveness is diminished when applied in real-world scenarios. Additionally, the research sought to uncover whether practical applications have led to innovative approaches for implementing or measuring these practices.
Data saturation was considered in this study, and interviews were continued until sufficient participants from different organisations and positions were consulted and no new relevant themes or insights emerged.
This approach ensuring that the data collected provided a comprehensive understanding of the topic and supported the reliability and depth of the findings necessary.
In the discussion section, the data was evaluated in relation to contemporary research and literature to provide context, uncover new insights, and identify gaps that had not been previously addressed. This comparative analysis helped bridge the gap between theory and practice, offering valuable contributions to the understanding of non-technical practices in secure software development.

3.8 Coding Process

To systematically analyse the transcripts collected from the semi-structured interviews, a coding process was applied to identify key themes and categorise patterns [21]. This involved assigning meaningful labels to ensure a structured, consistent and thorough interpretation of the data [21].
The coding process began with an initial familiarisation phase, which involved repeated reading of the transcripts, listening to audio recordings, and the creation of memos to capture emerging thoughts. This step also included improving the quality of transcripts which required in-depth review of the material. Following this, code development was carried out using an inductive approach to explore the wide range of data from the transcripts, aligning with the explorative nature of the research.
The research employed an inductive and open coding approach [22] in order to ensure that themes and patterns emerged organically from the data rather than being constrained by predefined categories. The initial open coding phase focused on systematically identifying key

findings, utilising in vivo coding [21] to preserve the participants' original words and maintain the authenticity of their perspectives. This process was further complemented by the integration of codes derived from non-technical practices documented in published Software Development Life Cycles (SDLC's) as classified by Kudriavtseva & Gadyatskaya [28]. By combining participant-driven insights with established methodological frameworks, this approach facilitated a comprehensive and in depth analysis of the data.
To manage and organise the collected data, Atlas.ti software [18] was used for coding and data management. The codes were refined over multiple iterations of transcript revisions to ensure they accurately addressed the research questions.
Following this, thematic analysis was applied to the established codes [21]. This entailed the categorisation of data into key themes based on commonality in order providing deeper insights and find emerging trends from different interviews and combine them to extract insights. The relevant Code Book can be found in the appendix under 8.2 Code Book.
Additionally, the results of the exercises completed by participants during the interviews were input into Microsoft Excel for quantification, analysis and visualisation.
The initial coding was conducted by the primary researcher, to ensure the reliability of the findings, a second coder with extensive qualitative research experience was enlisted. Discrepancies in coding were resolved through consensus meetings, with minor adjustments made to the coding scheme for clarity. There by inter-coder agreement was assessed and alignment was reached in meetings with independent coders.
This systematic approach ensured a comprehensive integration of both qualitative and quantitative data throughout the analysis.

3.9 Ethical Considerations

The study adhered to strict ethical guidelines, including the establishment of a consent processes, protocols for maintaining anonymity and confidentiality of data and the participant, and approval from an ethics review board.
All participants were required to sign a consent form that informed them about the aims and character of the research and clearly outlined how their data would be used and anonymised. Participants were informed of their right to withdraw from the study at any point without consequence.
Additionally, they were notified in advance that the interviews would be recorded. All data was stored and analysed in compliance with applicable data protection regulations, including the General Data Protection Regulation (GDPR). All of the individuals interviewed consented to participate in the study and publish the results in the anonymised, aggregated format.
To ensure anonymity, Interviewees were designated as Speaker 1, with the interviewer referred to as Speaker 2 and Employers were anonymised as Company A, and other companies were designated as Company B, C, etc. Further, participants were referred to as "P" for Participants (P0-10), with pronouns ("he," "his," "him") used consistently for all participants, regardless of gender.
The design of the interview questions ensured that no personal or professional details were disclosed. Identifying information was anonymised post-interview, and the named pronouns were consistently applied throughout the research process.

4 Results

4.1 Results Introduction

This research explores the definition, usage, role, effectiveness, and challenges of non-technical practices in secure software development across various organisational contexts. While secure development frameworks and technical solutions to increasing security have gained significant traction, human and organisational factors remain under-explored, despite their crucial role in preventing vulnerabilities and strengthening security in development.
Through interviews with industry professionals, this study captures first-hand insights into how non-technical practices are applied in organisations real-world secure software development processes.
The results chapter is structured as follows: It begins with an overview of the development methodologies and Software Development Life Cycles (SDLCs) used within organisations, providing context for how non-technical practices integrate into these frameworks. Next, it examines practitioners' definitions and general perceptions of non-technical practices, followed by an exploration of commonly used practices. After this their respective perceived effectiveness by practitioners and their organisations challenges in adoption are discussed. The chapter then establishes the impact of organisational maturity on the adoption and application of these practices and concludes with an analysis of how organisations assess and justify their effectiveness.
By addressing the results of the interviews conducted, this section provides comprehensive findings on the role and impact of non-technical practices in secure software development, offering valuable insights for practitioners and policymakers.

4.2 Used software development methodologies

Non-technical practices have been integrated into all different types of software development methodologies such as Agile, Waterfall and DevOps. For this reason we have decided to highlight the general software engineering approaches used by the participants organisations in order to provide context to the environment they have implemented their non-technical practices in.
Non-technical practices, as well as technical practices, in secure software development, act within a combination of and complimenting each other and themselves with the aim of producing a successful final software product.
To fully understand non-technical practices, it is necessary to examine their role and influence within the development process or framework in which they are implemented, thereby establishing their comprehensive impact on development. This is especially important for non-technical practices, as they mostly transcend the whole development process when influencing human, organisational factors and others who act in similar overarching fashion on the development process [28]. For this reason, participants were asked about the current development methodology, formal or informal, they use their practices in order to provide a deeper context on non-technical practices and their influence, implementation and final effectiveness.
During the interviews, it was found that a large number of participants did not follow a formalised methodology. However, all participants stated that their approaches were based on custom methodologies derived from Agile/Scrum, with some incorporating DevOps for further iterative work. Additionally, several issues influencing methodology choices were highlighted,

the most prominent being organisations forced to work in hybrid methodologies or required to follow customer-specified development methodologies influencing the usage of non-technical practices.

4.2.1 Lack of a Formalised Process

A lack of formalised development methodologies was mentioned by eight participants, who described their approaches as informal, flexible, and guided mainly by best practices rather than established frameworks. This informal approach allowed teams to adapt dynamically based on project needs, such as adding additional security, but often resulted in inconsistencies across different projects. P1 explained that their organisation does not follow a specific methodology, saying:

> *"We tend to work agilely, but there's no formalised approach that dictates how we do it." (Interview 1)*

This adaptability was echoed by P9, who noted:

> *"Of course, we follow best practices, but not a specific software development lifecycle." (Interview 9)*

Participants also highlighted the limitations of this informality. For example, P4 acknowledged that while their organisation does not have formal processes, adhering to basic practices is crucial, providing things such as a baseline of security, stating:

> *"We don't really have formal processes in place. I think it's important to at least adhere to some basic practices in general." (Interview 4)*

This lack of formalisation often led to teams relying on iterative trial and error to determine effective practices for each project. While providing a high degree of agility, this informal dynamic approach could disrupt consistency, requiring frequent reassessment of workflows and potentially hindering long-term efficiency.

Key Finding: The lack of formalised methodologies was very common among participants organisations and led to flexible, best-practice-driven approaches, enabling adaptability but causing inconsistencies, a reliance on trial and error, and potential inefficiencies.

4.2.2 Agile

Agile approaches, particularly Scrum, emerged as the dominant development approaches in the interviews, with all participants confirming that their organisations have adopted some form of development based on Agile foundations. This widespread adoption was observed across organisations of varying sizes, types, and maturity levels, reinforcing Agile's status as the industry standard. P1 explained their organisation's Agile-inspired workflow, saying:

> *"We work in sprints, following an agile-like approach. We outline the specifications, conduct some sprints, gather feedback, revise as needed, test, deploy, and then update the documentation. It's a controlled process, but it doesn't fully align with traditional Agile practices." (Interview 1)*

This response highlights how Agile is often tailored to fit the specific needs of different teams and projects. Similarly, P8 reflected on their broader experience across multiple organisations, noting:

> *"Yeah, I've mostly worked with Agile. [. . .] In general, Agile is much more common in the industry, it's what most people tend to use." (Interview 8)*

This perspective underscores the prevalence of Agile methodologies as the default framework in the industry. The compatibility of non-technical practices with Agile's iterative and flexible nature was a recurring theme. Participants emphasised the importance of aligning these practices with the human-centred focus of Agile, which contrasts with more finite, traditional methodologies. By integrating non-technical practices into Agile frameworks, organisations can ensure a seamless fit within their existing workflows while fostering team collaboration and adaptability.

Key Finding: Agile, particularly Scrum, is the dominant methodology, its basis being adopted across diverse organisations. While widely used, Agile is often tailored to specific needs rather than strictly following traditional frameworks, creating custom development approaches. Its iterative nature supports flexibility and collaboration, helping with the integration of non-technical practices.

4.2.3 DevOps

The interviews revealed that DevOps methodologies, particularly CI/CD pipelines, are integral to post-delivery practices for maintaining and improving continuous software products. More than half of the participants mentioned implementing DevOps approaches to some extent, with several emphasising their reliance on CI/CD pipelines for efficient delivery and maintenance. P10 explained the centrality of CI/CD pipelines in their workflow:

> *"We implement this in every project. The focus is on the CI/CD pipeline, which involves systems such as Dev, Stage, and Prod. We always work with a CI/CD pipeline in place." (Interview 10)*

Beyond technical efficiency, participants discussed how DevOps practices also influence non-technical aspects, such as customer communication and system monitoring. P6 shared:

> *"I would consider myself a DevOps professional because we handle things like monitoring and alerting, meaning if our service goes down, we'll notice and have to figure out why. We also manage support channels, so if a user has an issue, they can reach out directly, and we'll see, 'Oh, that's a bug, and we need to maintain it.' Essentially, we maintain our software ourselves." (Interview 6)*

Several participants highlighted the integration of DevOps with Agile methodologies like Scrum to create a seamless blend of technical and non-technical practices. P4 noted:

> *"As a software development team, we follow Scrum for our main product while also adhering to standard DevOps practices regarding cloud infrastructure and deployments. [. . .] On the other hand, there are also various technical practices that I should mention, such as integrating security scans into our CI/CD pipelines." (Interview 4)*

Similarly, P6 described their dual role, stating:

> *"I would describe myself as a DevOps developer because I handle operational tasks, such as managing and deploying Docker containers, while also contributing to development. I see this as DevOps, but our team's way of working follows Agile principles." (Interview 6)*

These findings illustrate how DevOps, particularly CI/CD pipelines, extends beyond technical efficiency to play a pivotal role in shaping non-technical practices. By blending Agile and DevOps methodologies, organisations foster a holistic approach to software development and maintenance, balancing operational efficiency with customer satisfaction and system awareness.

Key Finding: DevOps, particularly CI/CD pipelines, is widely used for software maintenance and evolution, with a strong emphasis on operational efficiency and automation. Many organisations integrate DevOps with Agile methodologies, balancing technical practices with non-technicals such as customer support and feedback through system monitoring to quickly respond to additional requirements or security needs.

4.2.4 Customisation of Methodologies

Many participants highlighted the intertwined use of Agile/Scrum and DevOps in their organisations, emphasising that few rely on the original, unaltered forms of these methodologies. Instead, organisations frequently adapt these frameworks to suit their unique needs and contexts by adding additional practices to increase security or quality.
The interviews revealed that the methodologies implemented by organisations are rarely rigid or "pure." All participants of the study described deviations from formal frameworks to accommodate the specific requirements of their teams and goals. Teams often integrated additional practices into an Agile framework, creating informal, customised networks of best practices or entirely new custom methodologies inspired by Agile and Scrum principles. P4 illustrated this point by saying:

> *"It's all 'home brew.' Although we work traditionally with Scrum, we also have significant deviations from it. I think it's always adapted in a way that works for the company."*

Similarly, P6 explained:

> *"We generally adhered to Agile and just added or adapted things that seemed sensible to us." (Interview 4)*

This informal and flexible approach was commonly described. For instance, P1 noted:

> *"No, we don't actually employ a specific methodology. We tend to work agilely, but there's no formalised 'This is how we do it.' [. . .] It's a controlled process, but it doesn't fully align with traditional Agile practices."* (Interview 1)

For example, participants working to provide software internally for their organisations described this customisation as involving lightweight or streamlined processes. For example, P3 mentioned:

> *"Yeah, we primarily use Agile, but it's a very slim version of it, I would say. So, we don't need to do a lot of the typical processes like having a separate retro or a backlog trimming." (Interview 3)*

Other participants emphasised that adjustments were often driven by experience focusing on human factors, such as improving communication or avoiding burnout. P10 explained:

> *"I hate those endlessly long meetings... In my teams, meetings are very strictly scheduled, and the times are greatly reduced. [...] I regularly cut meetings from classical Scrum because I find them to be bullshit. Instead, I work with smaller teams." (Interview 10)*

These differences in adaptations can be found in most organisations, with P7, who works in consulting, speaking on different organisations he had worked with:

> *"Yeah, usually it's like Agile Scrum, but it really depends. Sometimes they have their own twist, their own fork of it. It's not the most clean implementation." (Interview 7)*

These examples demonstrate how organisations rarely adhere to rigid, formalised frameworks for their development. Instead, they blend, adapt, and streamline practices to suit their unique needs and challenges, to add additional security levels and other adaptations as needed. Due to this custom nature, adopting non-technical practices relies more on individual organisational contexts, their clients and development approaches. Unlike structured methodologies, which provide established guidelines, integrating non-technical practices often requires organisations to conduct additional research and adjustments to ensure compatibility with their customised workflows.

Key Finding: Most organisations follow a customised, self-built development approach rooted in Agile/Scrum, and DevOps. These frameworks are adapted based on experience, best practices, and organisational needs, balancing efficiency, flexibility and security through non-technical practice incorporation to fit specific development needs or customer bases.

4.2.5 Introducing MVPs to SDLCs

An interesting integration into development approaches was observed in the interviews to address human-centric issues, such as miscommunication or disagreements on project scope, during the requirements gathering stage.
This integrated practice was designed to reduce problems in the requirements elicitation phase, where errors in organisational decision-making, awareness, or communication often lead to project delays and increased costs. By introducing a Minimum Viable Product (MVP) with standard mock elements and a front end wraper to collect feedback early in the Software Development Life Cycle (SDLC), organisations aimed to improve clarity and alignment with customer expectations before proceeding with standard development processes.
Two participants from different organisations described how their teams used MVPs to gather customer feedback and refine requirements before fully developing the product using Agile methodologies. P10 explained:

> *"Our goal is therefore to deliver a 'bullshit' frontend as quickly as possible and tell the users: 'Click around a bit.' [...] That means we first build the frontend, and then the users can express their wishes. Based on this, we have the discussion, and finally, the back-end is developed." (Interview 10)*

Similarly, P7 elaborated on the benefits of this approach:

> *"The idea is you basically you build an MVP like a minimum viable product so like really something that barely works and with that you can go to the customer: 'Hey, like, this works, we just have to refine a tiny bit.'" (Interview 7)*

This practice of involving customers early in the design process significantly improves the overall development experience. By providing users with a tangible, interactive version of the product, it fosters better engagement and facilitates clearer communication of their needs. Customers are more likely to provide valuable feedback when they can directly experience the MVP, making the requirements gathering process more precise and human-centric. As P10 highlighted:

> *"In general, I think it's important to give users something as quickly as possible that they can not only see but also experience. This has top priority. It's about involving the users, but it's crucial that they are also willing to engage. Often, they lack motivation and don't participate properly they need to see and feel something tangible, something they can interact with." (Interview 10)*

The introduction of MVPs at the beginning of the SDLC addresses common challenges in the requirements elicitation phase, such as unclear project scope and communication gaps. By aligning project goals more closely with user needs early on, this approach minimises misunderstandings, reduces the likelihood of costly changes later in the process, and ultimately improves the efficiency and effectiveness of software development projects.

Key Finding: Integrating Minimum Viable Products (MVPs) early in the Software Development Life Cycle (SDLC) helps mitigate miscommunication and scope disagreements during requirements gathering. By providing users with an interactive prototype, teams improve engagement, refine requirements, and align project goals with customer expectations, reducing costly revisions later in development.

4.2.6 Hybrid Methodologies: The "Frankenstein" Systems

Several participants expressed frustration with hybrid methodologies that fail to fully embrace the benefits of either Waterfall or Agile approaches. These "Frankenstein" systems often emerge as a compromise between internal team preferences and external client demands. P10 described this:

> *"We also have these unpleasant hybrid Frankenstein forms that are neither truly Agile nor strictly Waterfall."* (Interview 10)

Such hybrids frequently combine structured planning phases (to meet client requirements) with Agile development practices, resulting in a facade of adaptability but often introducing inefficiencies. P0 elaborated:

> *"You always have a planning phase and then you always have a development phase... during development it's agile, but customers don't want to see you saying 'ohh I'm gonna plan, check, do, act' the entire time, because that way they don't have any grip on what you're actually doing." (Interview 0)*

This tension highlights the challenge of reconciling Agile's emphasis on flexibility and iterative progress with clients' preference for predictable and controlled timelines. Hybrid methodologies, while designed to accommodate both approaches, can struggle to deliver the full benefits of either, leading to dissatisfaction and inefficiency in practice.

Key Finding: Forced hybrid methodologies, or "Frankenstein" systems, arise when organisations blend approachers such as Agile and Waterfall, often to meet client demands, giving clients more control but limiting the full benefits of either approach. These compromises are not popular with developers and often leads to inefficiencies, reducing both adaptability, structured planning effectiveness and established security procedures.

4.2.7 Client-Driven Methodology Choices

Client requirements play a significant role in shaping the choice of Software Development Lifecycle (SDLC) practices, often forcing teams to prioritise structured planning and fixed timelines over the flexibility offered by Agile methodologies. This is often done for higher levels of control and adherence to the client organisations security standards for their systems, P7 highlighted this challenge:

> *"Sometimes for clients we have to develop their software in a specific way, but that's usually not something we choose. It's something that's defined by some standards." (Interview 7)*

Strict client guidelines frequently impose rigid meeting structures and exhaustive review processes that align more closely with traditional Waterfall workflows. P10 shared their experience:

> *"For example, at Company A, they have very strict guidelines, and it's hell... they demand two-hour plannings and reviews every two weeks." (Interview 10)*

P10 further elaborated how externally pre-defined project requirements effectively forced their teams to adopt Waterfall methodologies:

> *"Sometimes you are forced to use Waterfall because the requirements gathering has already been done by another firm. Then you just have to do Waterfall." (Interview 10)*

This externally mandated approach often prevents organisations from leveraging their established, standardised practices. Instead, teams are compelled to either scale down to a minimal set of practices or adopt unfamiliar methods to meet client demands. These compromises can undermine the efficiencies typically achieved through refined development processes, leading to reduced productivity and an increased risk of project delays, setbacks or vulnerabilities.

Key Finding: Client requirements can dictate SDLC practices, limiting teams flexibility in methodology choices. Strict guidelines and predefined project structures frequently enforce Waterfall approaches, with rigid planning, review processes, and external standards shaping development workflows. This can disrupt organisations established non-technical practices, reduce efficiency, and increase project risks by disrupting established processes.

4.3 RQ1: "What is the perception on non-technical practices and their impact on Software development?"

4.3.1 Definition of Non-Technical Practices

To better understand their perceptions of non-technical practices, participants were first asked how they defined non-technical practices and how they differentiate them from technical ones. This provided insight into the specific actions and functions they attribute to non-technical practices within the development process and offered context for their views.
The majority of participants defined technical practices as activities that directly influence the visible product. These include tasks such as coding, testing, and encryption, which were frequently described as hands-on actions that affect the functionality and structure of the software. P10 explained:

> *"Technically speaking, everything that impacts the visible product is really part of the technical area. When the developer works on the code, that is the technical work." (Interview 10)*

P1 supported this view, adding that technical practices often involve specific tasks aimed at ensuring the security and proper functioning of the system:

> *"The technical aspects focus on encryption, secure key management, OAuth, and obfuscation. These directly affect the system." (Interview 1)*

Similarly, P5 emphasised that technical practices require a strong understanding of the software and its underlying components:

> *"The technical side is about being cautious about what you're writing and understanding the software. [. . .] Fundamentally, it's about understanding how your system works." (Interview 5)*

These responses illustrate that participants broadly associate technical practices with the direct manipulation and creation of code and the technology behind it. In comparison, most participants described non-technical practices as activities that support technical work without directly interacting with the codebase. These practices were said to provide structure and context for technical tasks, contributing to the overall success of a project. P10 characterised non-technical practices as encompassing efforts outside of direct feature requests:

> *"Everything that is 'non-technical' is essentially the fun we do around it. So, everything that is not an explicit feature request from the customer, I would define these as 'non-technical'." (Interview 10)*

P1 explained non-technical practices in terms of their broader framework, highlighting their organisational value:

> *"It's all about ensuring we consider the 'four Ps': people, process, products, and permissions. These elements create a comprehensive security framework that supports our technical initiatives." (Interview 1)*

P6 added that non-technical practices provide a higher-level overview, contrasting them with the granular nature of technical tasks:

> *"Technical practices are deeper, more granular, like making a function work properly, while non-technical practices are more of a broader overview, like how we design our architecture or move an endpoint into the service." (Interview 6)*

These descriptions reflect a clear separation between the two sets of practices in participants' minds. Non-technical practices were viewed as supporting and influencing technical practices, but not directly impacting the software itself.
Despite this distinction, several participants noted the interdependence of these practices. P0 illustrated this point by describing how non-technical actions could have technical consequences:

> *"A person storing their password on a Post-it note is not technical, but the repercussions are very much technical." (Interview 0)*

In summary, participants understood technical practices as those directly related to the creation or modification of the product, while non-technical practices were perceived as influencing the environment and processes that guide technical tasks. Both were seen as interconnected and essential for successful development.

Key Finding: Participants distinguished technical practices as direct code-related activities affecting the product, while non-technical practices were seen as supporting structures, processes, and organisational frameworks. Despite this distinction, both were recognised as interdependent, with non-technical practices shaping the environment for effective technical work.

4.3.2 Positive Opinion of Non-Technical Practices by Developers

In the interviews, it was highlighted that all participants recognised the value of non-technical practices within software development life cycles to a certain extent. They mentioned that they saw their positive contributions to development, with most sharing that they thought non-technical practices were just as important as technical practices for achieving final success. P9 supports this idea by saying:

> *"Yes, I see it as roughly equally weighted with the technical practices. Of course, it depends on how exactly you draw the line between the two. I would say it's about 50/50, also in terms of importance." (Interview 9)*

Similarly, P3 emphasised the necessity of combining both types of practices, particularly for the inclusion of security into organisations development process:

> *"I think both (technical and non-technical practices) are always needed in order to develop secure code." (Interview 3)*

Some participants expressed that non-technical practices could be even more important than purely technical ones due to their holistic nature and contribution to the overall security environment. P1 highlighted this perspective, stating:

> *"I believe that while core technical practices such as testing and encryption are crucial, non-technical practices can be equally, if not more, important. These practices establish a solid foundation for security from the outset." (Interview 1)*

This sentiment was echoed by P4, who emphasised their indispensability:

> *"Okay, so generally yes, I think it's impossible without them. For me, they are almost more important than the technical practices." (Interview 4)*

As a whole, participants expressed positive views on non-technical practices, recognising their contributions and often emphasising their importance in comparison to technical practices. While technical practices are more tangible and provide immediate results, non-technical practices were appreciated for their long-term and foundational impact on the development process.

Key Finding: Participants recognised the value of non-technical practices in secure software development, often viewing them as equally important, if not more so, than technical practices. The responsibilities covered by non-technical practices mentioned most frequently by participants were their role as Human-Centred, Organisational and Decision-Making Practices. Non-technical practices were seen as foundational, contributing to long-term success, security, and overall development effectiveness.

4.3.3 Benefits of Non-Technical Practices in Managing Human Factors

Participants highlighted their perception of non-technical practices as being responsible for managing and guiding human influence within their organisation's software development life cycle. P0 emphasised that human involvement is a critical vulnerability, explaining:

> *"The human factor is the most vulnerable. I wouldn't write down 'humans are flawed' [. . .] A lot of errors are technical faults, but in general, you can't eliminate all risk. You could build a bank vault that's completely secure, but at some point, someone needs to interact with it, like opening the door. That's when human involvement introduces potential vulnerabilities. And the most vulnerable part of anything is where humans are present." (Interview 0)*

This highlights the recognition that human factors are central to many vulnerabilities in software systems, particularly in security. As P2 added:

> *"That's why social engineering and general defence mechanisms are becoming increasingly important compared to purely technical defence mechanisms." (Interview 2)*

Participants also highlighted the benefits of non-technical practices for addressing these human vulnerabilities and fostering improvements in technical outcomes. For example, P7 shared:

> *"I'd say these non-technical things are more about managing and supporting humans, like everyone makes mistakes, so if you want to increase security and quality, it's good to have them." (Interview 7)*

Similarly, P6 identified their contribution to quality and knowledge distribution in development processes, stating:

> *"Yes, non-technical practices contribute to knowledge distribution and software quality in methodologies. These are the two points, I think, that contribute to things." (Interview 6)*

Participants also emphasised how non-technical practices indirectly improve technical practices by shaping human behaviour and processes. P5 illustrated this, saying:

> *"Non-technical practices are everything that happens around it. For example, security training is a non-technical practice, but it helps enhance your technical practices. It teaches you how to program something without security flaws. Documentation also falls into this category because it guides you on how to properly structure your code." (Interview 5)*

Some participants focused on how non-technical practices address human flaws that could negatively affect development. P8 elaborated:

> *"Non-technical practices, on the other hand, are more about how you behave as an employee at the firm how you handle the information you deal with daily. It could be anything that could potentially lead to a security breach. For example, making employees aware not to discuss company matters in public places where it could be overheard or be detrimental." (Interview 8)*

By addressing human factors such as developer mistakes, communication gaps, misunderstandings, and privacy concerns, participants recognised non-technical practices as essential for fostering reliable, security and higher-quality development processes. These practices were viewed as key to creating an environment where human involvement positively influences the software development lifecycle and its vulnerabilities were managed.

Key Finding: Non-technical practices were seen as crucial for managing human factors in software development, addressing vulnerabilities such as errors in understanding, communication gaps, and other security risks. These practices support human involvement by managing factors such as developer mistakes, communication gaps, misunderstandings, and privacy concerns, ultimately enhancing technical outcomes and creating a more reliable development environment.

4.3.4 Organisational and Decision-Making Practices

Many participants described non-technical practices as operating at an organisational and strategic level, contrasting with the operational focus of technical practices. For example, P4 explained:

> *"That's an interesting question. For me, it all blurs a bit, but I would probably distinguish between two categories: technical measures and organisational measures. Many of the normal processes, like code reviews and manual acceptance testing, I would categorise as organisational, non-technical practices." (Interview 4)*

This perspective highlights how non-technical practices provide a comprehensive, high-level view of development, guiding technical practices throughout the software lifecycle. P6 emphasised the broader design considerations these practices address:

> *"Non-technical is more of an external viewpoint, like saying, 'Hey, do it this way or that way,' and these are more design questions, broader design questions. No technical tool can really tell you that; you need to look at it from a non-technical perspective." (Interview 6)*

Similarly, P8 highlighted their role in promoting situational awareness and guiding action:

> *"OK, so for technical practices versus non-technical practices, what I think is that non-technical practices are more about certain processes or mechanisms that help make us more aware of the situation or guide us to act in a certain way. They tend to operate in a broader sense, rather than focusing strictly on technical specifications." (Interview 8)*

Participants often described non-technical practices as providing structure, coordination, and alignment with organisational goals. P6 elaborated on this point, describing how these practices offer an overarching view:

> *"The other thing is code reviews, which are more about the broader organisational context. Non-technical practices are more of an overview, like how we design our architecture or 'Hey, move the endpoint into the service,' or something like that." (Interview 6)*

Non-technical practices were also recognised for their role in guiding developers and ensuring clarity in communication and processes. P8 summarised this succinctly:

> *"Non-technical practices focus more on how you generally handle information or follow processes." (Interview 8)*

Overall, participants demonstrated a shared/overlapping perception of non-technical practices as being integral to organisational and decision-making activities. These practices were viewed as essential for maintaining secure and structured development processes, ensuring proper planning, and supporting teams in aligning both technical and organisational objectives effectively.

Key Finding: Non-technical practices were consistently viewed as essential for providing structure, coordination, and alignment with organisational goals. They help guide decision-making, foster situational awareness, and ensure clarity in communication, which ultimately supports both technical practices and overall organisational objectives. These practices contribute to a broader strategic view, aiding in the design, planning, and execution of development processes, leading to more secure and higher quality development.

4.3.5 Enhancing Quality, Stability, and Productivity

Another quality of non-technical practices that participants viewed positively was their impact on quality, stability, and productivity. Non-technical practices were described as improving the quality and stability of work produced by organisations that adopt them. Participants highlighted that these practices encourage considering the larger system during development, making individual components easier to integrate and resulting in a higher-quality final product. As P5 stated:

> *"I think, overall, they contribute to the quality and stability of systems. Engineers who take on those non-technical practices inevitably produce a better product and become more well-rounded, thinking beyond just the code itself. In society, of course, we want to earn money, and companies that hire engineers with these skill sets are the ones paying the most." (Interview 5)*

Participants also noted how non-technical practices contribute to awareness and process clarity, particularly in larger or growing teams. P4 highlighted this importance, saying:

> *"It is important to have a good process because not everything can be technically codified or ensured in the pipeline. People should have an awareness of the process, of quality, and so on, and also know how they need to act. This is especially crucial in a growing team. For example, I have over 20 people under me, and they need to have a general idea of the direction they should be heading."* (Interview 4)

In addition to improving quality and stability, participants highlighted how non-technical practices enhance team productivity by improving communication and collaboration. As P7 explained:

> *"All these like team-building things, they're really important. All these like things you do to get the team together and make them talk about their hobbies and like, have like a nice work environment, that's all these non-technical things. But they do help because if developers or any other employees feel really good about what they're working on and their work environment, they're going to be more productive." (Interview 7)*

P1 also emphasised the role of non-technical factors such as culture and open communication in improving team productivity and security:

> *"The culture we've cultivated is also crucial; it's supportive and encourages open communication. Everyone knows that there are no silly questions, especially concerning security. If someone has a concern or needs clarification, our team jumps in without hesitation, providing guidance without judgment. It creates an environment where team members feel comfortable seeking help and sharing knowledge, ultimately strengthening our overall security posture." (Interview 1)*

Participants further recognised that non-technical practices help in organising and streamlining workflows, particularly during product releases. P8 shared their observation:

> *"I think that's why non-technical practices, especially the ones I mentioned like documentation and processes, are so important. If you have those set in place, they contribute a lot to the overall improvement. You can notice this during product releases. When things take longer, it leads to everything piling up at the end, and then everyone rushes to get things sorted out. But if the process is clear from the start, that workload is spread out better." (Interview 8)*

These insights demonstrate the recognition among participants of the role non-technical practices play in enhancing productivity, quality, and stability. They acknowledged how these practices improve technical outputs by fostering better communication, clearer processes, and

stronger team dynamics, ultimately contributing to a more effective, sustainable and secure development process.

Key Finding: Non-technical practices were recognised for enhancing quality, stability, and productivity in software development. Participants highlighted how these practices improve communication, team collaboration, and process clarity, leading to better product integration and higher-quality outcomes. They also help streamline workflows, create supportive team cultures, and ensure smoother project execution, ultimately boosting productivity, security and effectiveness.

4.3.6 AI and the Future of Non-Technical Practices

An interesting perspective shared by participants predicted the increasing importance of non-technical practices in connection with the rise of AI and the automation of more technical tasks. P2 noted:

> *"Within the development methodology, this division (of practices) has very high importance. It's becoming increasingly significant because technical practices are increasingly taken over by AI. This gives you more time to focus on such aspects."* (Interview 2)

This perspective highlights a shift in the future landscape of software development, where the role of non-technical practices may become more central as AI continues to take over more technical tasks. Practitioners are aware of this evolving trend and anticipate that it may affect the current balance of importance between technical and non-technical practices.
Additionally, some participants mentioned that they already incorporate AI into their development processes, particularly for simplifying menial tasks. For example, P10 shared:

> *"Exactly, this is one of the main concepts we are now implementing. We are currently working on developing new methods to structure the feedback coming through the wrapper (Feedback tool) using AI. We are using large language models to sort it and so on, to improve all these processes."* (Interview 10)

This illustrates how AI can enhance non-technical practices, influencing the development process by automating tasks and improving efficiency. As AI continues to evolve, it may further reshape the role of non-technical practices, driving greater emphasis on organisational, strategic, and human-centric aspects of software development.

Key Finding: As AI automates more technical tasks, non-technical practices are expected to become more important to manage them. This shift allows developers to focus on organisational, strategic, and human-centric aspects, with AI already being used to improve processes like feedback structuring.

4.3.7 Negative Opinions of Non-Technical Practices by Developers

Non-technical practices are not regarded solely as positive by practitioners. While all participants acknowledged their advantages in the development process, some also highlighted negative factors that shaped their opinions. Two participants raised concerns about these practices, particularly their perceived impact on trust and priorities.

P7 expressed a strongly negative opinion, especially about non-technical practices that involve security or restrictions. P7 said:

> *"So, in general, I think very negatively of such practices. I think you should put more trust in developers and especially technical people, like anyone who does something technical. You should just put more trust in them. Don't try to lock down their systems with weird security. Yeah, just trust the process, basically." (Interview 7)*

This perspective suggests that non-technical practices, particularly those focused on restrictions, can undermine trust in developers and their technical expertise. Such practices may lead to inefficiencies or frustration, as developers feel constrained by measures that appear unnecessary or obstructive.
P0 also expressed scepticism, emphasising that non-technical practices often receive less attention compared to technical practices. P0 noted:

> *"In the software development cycle, I don't think much attention is given to non-technical practices. The focus is usually on what's considered most important, like testing and making changes after testing." (Interview 0)*

This highlights a view among some developers that non-technical practices may be less relevant than technical ones, particularly those with immediate and tangible effects on their day-to-day work.
These comments illustrate that although non-technical practices are widely recognised for their benefits, a portion of developers remains critical. Their concerns primarily centre on the perceived undermining of trust and the prioritisation of technical over non-technical approaches.

Key Findings: While non-technical practices offer benefits, some developers view aspects of them negatively, when they impose restrictions or shift focus from technical priorities. Concerns by developers include reduced trust, inefficiencies, and lower emphasis on core tasks. This scepticism highlights the need for careful integration to promote understanding and prevent frustration and resistance in organisations.

4.3.8 In depth non-technical practice drawbacks expressed in RQ4

In the results section, Research Question 4 (RQ4) offers a more in-depth analysis of specific negatively perceived aspects of non-technical practices, with a particular focus on their adoption and application as reported by the studies participants.

4.4 RQ2: "Which non-technical practices are commonly used in industry?"

Usage of non-technical practices in organisations, as perceived by participant
There are an undefined number of non-technical practices that can be implemented into organisations SDLC's, due to there being an undefined number of ways that human factors can influence most processes, including software development. To structure the usage of non-technical practices by the participants interviewed we will use the classification of Kudriavtseva

& Gadyatskaya **[28],** which stems from the available 28 methodologies on the topic of SSDM's, that established the criteria and the categories of:

Risk management framework Establishing a structured approach to identify, assess, and manage potential risks during the software development process to mitigate threats. (Vulnerability-bounties, Test-Environments, Risk-Framework, Supply-chain-security, Penetration-testing, HR-Management)

Security metrics Collecting and analysing data related to security practices and incidents throughout the development lifecycle to evaluate and improve security performance. (Security ticket resolution time, Security Incident Frequency, Phishing Simulation Success Rate, Time to Patch Critical Vulnerabilities, Penetration Testing Findings, Third-Party Dependency Risks)

Building a culture of security Promoting a security-first mindset at all stages of development by educating team members on their individual roles in safeguarding the software and its users. (Security-Team&Security-officer, Security-Education, Project-management)

Understanding human behaviour Recognising how decisions and collaboration within the development team, as well as customer actions, can impact software security and quality, and addressing vulnerabilities stemming from human factors. (Analyse-attacker-profiles)

Policies, strategies, standards, and conventions Defining and implementing guidelines, best practices, and compliance standards to ensure secure, consistent, and high-quality software development. (Coding-Conventions & standards, Development-Environment, Quality-Assurance, Technical-Planning/design)

Auxiliary practices of incident or vulnerability response Creating action plans and processes for addressing and mitigating potential incidents or vulnerabilities as they arise during or after development. (Incident-Response-plan, Monitoring)

Communication process and customer responsibilities Establishing clear communication channels between developers, stakeholders, and customers, and defining customer responsibilities to maintain secure and responsible software use. (Business-impact-analysis, Customer-release-risk, Customer-feedback)

Ethics Ensuring that development decisions are guided by principles of responsibility, fairness, and transparency, with a focus on respecting user privacy and avoiding bias. (Ethics-hotline)

Privacy Designing and developing software to protect user data, ensuring compliance with privacy regulations, and maintaining data security throughout the software lifecycle. (Data-access-management, Physical-access-management)

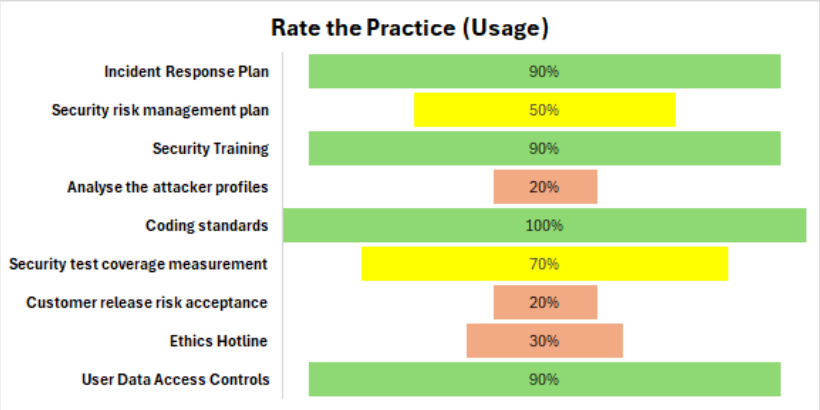

Figure 1: Rate the Practice: Usage Results

Practices collected not categorised by Kudriavtseva & Gadyatskaya [28] : Audits (used by nine people), Manual-Code-review (used by 10 people) , Pair-programming (used by 8), Documentation (used by 8 people)

4.4.1 Rate the practice: Non-technical practice usage

In an exercise presented to the participants, one significant practice from the categories was chosen to represent them and participants where asked if they employed these in their current organisations development process.

Category	Practice	Usage %
Incident or vulnerability response	Incident Response Plan	90%
Risk management framework	Security risk management plan	50%
Building a culture of security	Security Training	90%
Understanding human behaviour	Analyse the attacker profiles	20%
Policies, strategies, standards and conventions	Coding standards	100%
Security metrics	Security test coverage measurement	70%
Communication process and customer responsibilities	Customer release risk acceptance	20%
Ethics	Ethics Hotline	30%
Privacy	User Data Access Controls	90%

Table 2: Rate the Practice: Category Usage Results

The results of this activity, as seen in Table 2, revealed a clear disparity in the usage percentages of certain non-technical practices, which should not directly be applied to the category they represent. Despite differences in organisational size and maturity, the findings indicated that certain non-technical practices remained consistently adopted, suggesting their universal relevance and adaptability across various development environments. There was a relatively even distribution of these practices across the usage scale, with no practice being unused and only a single one, coding standards, being adopted across all of the participants organisations.
.

Policies, strategies, standards and conventions: Coding standards Coding standards were established as the most used non-technical practice by participants, with all of them stating that these were employed in their organisation development process (Fig. 1). They were

the only practice to do so. The fact of coding standards overwhelming presence portraits that non-technical practices are very common in modern development approaches and highlights the simplicity or necessity of implementing them.

Incident or vulnerability response: Incident response plan Incident Response Plan were also present in 90% of development processes (Fig. 1). This high prevalence of preparing responses again shows the influence of non-technical practices but also the current need for organisations to be prepared for incidents and that in a formalised manner.

Building a culture of security: Security Training All participants had engaged in security training/education at some point, though not all currently incorporated it into their development process. As can be seen in Fig. 1 90% of participants currently still did though, they described a variety of formats and topics (e-Learning , security presentations, security workshops), with anti-phishing education being the most mentioned across all participants. This shows the current threat environment in the industry being at a high level across all organisational types and pure technical measures not being able to or the preferred way for organisations to protect them selves.

Privacy: User Data Access Controls User data access control was also present in 90% of organisations (Fig. 1), again pointing to frequent and high level security threat that organisations are combating. Access controls have become vital to prevent malicious actions but also incidents caused in accident by individuals, that should not have had access to certain data or development permissions.

Security metrics: Security test coverage measurement Security test coverage measurement was also clearly present in most development processes, though at 70% not as prevalent as the previously mentioned non-technical practises (Fig. 1). The practice was representative of the collection of general security metrics, which was most often described as being done using the ticket system as a tool.

Risk management framework: Security risk management plan Following this there was a clear drop in the number of adoptions of the practices provided to practitioners. Only 50% of participants stating their organisations formally tracked their developments possible security risks in a fixed management plan (Fig. 1). Risk management is a continuous effort that entrails the monitoring and regrading of current risks as well as the addition of new risks. This additional contentious effort and the subjectiveness of rating and establishing risks were mentioned as possible causes for its mediocre adoption rate.

Ethics: Ethics Hotline Though most were aware of Ethics Hotline's such as numbers or e-mail addresses for practitioners in ethical dilemmas to contact, there was a difference in interpretation by some if this was meant for internal (such as hazing) or external (such as bribery by competitors) ethical dilemmas. 30% of practitioner's organisations employed it in some form or way, with the participants that had them stating that they were nearly never used though (Fig. 1).

Understanding human behaviour: Analyse the attacker profiles This was followed by the analysis of potential attacker profiles, which was slightly less popular with 20% of par-

ticipants organisations employing it (Fig. 1). The analysis of attacker profiles was a practice only rarely known by participants, this was most often justified with either the size or the lack of importance of data they collected. When asked most participants referred to their organisation employing a more blanketed security approach, instead of preparing against individual adversary types.

Communication process and customer responsibilities: Customer release risk acceptance Equally with Analyse the attacker profiles, Customer release risk acceptance was only present in 20% of organisations (Fig. 1). It was highlighted during the exercise that this was often not done by organisations due to the continuous nature of the development which bleeds into maintenance or if the product was designed to be used internally in the organisation. With the rise of agile/scrum and dev ops and a move away from providing complete software, Customer release risk acceptance was in many places mentioned to no longer be applicable.

Key findings: Key findings reveal trends in the adoption of non-technical practices, shaped by organisational priorities. Policies, strategies, standards, and conventions like coding standards were universally adopted (100%), emphasising their fundamental role in development. Incident or vulnerability response practices, such as incident response plans (90%), and building a culture of security with security training (90%) were widely implemented. Privacy practices, like user data access controls (90%), also received strong focus.
However, practices related to security metrics (security test coverage measurement, 70%) and risk management frameworks (security risk management plans, 50%) saw moderate or lower adoption, suggesting challenges in implementation. Ethics practices, such as the Ethics Hotline (30%), and understanding human behaviour (attacker profiling and customer release risk acceptance, both 20%), were less commonly adopted, highlighting these as lower priorities or more complex to implement.
Overall, organisations prioritise immediate security needs and privacy over longer-term, more complex measures, indicating a selective approach to integrating non-technical practices into the SDLC.

4.4.2 Non-technical practices most used by practitioners

The following non-technical practices were the ones applied most often in organisations development processes according to the participants answers to the Miro exercise. This excludes those mentioned in the rate my practice exercise above, who will be covered in-depth in RQ3.

Manual code review Manual Code Review was widely regarded as a fundamental non-technical practice, with the majority of participants endorsing its effectiveness. It was seen as crucial for maintaining code quality, improving security, and fostering knowledge sharing within development teams.
Several participants highlighted the importance of having multiple reviewers assess code before it is merged. P10 emphasised the necessity of ensuring thorough reviews, particularly when working with student developers, to maintain quality:

> *"I would always have two people review the code before approving it. I especially want to use code reviews with working students because they sometimes do this on the side, which is completely fine. But still, I'd always have two people look at the code to improve the approval process."* (Interview 10)

Similarly, P7 described how their organisation had increased the number of required approvals to prevent low-quality code from being pushed:

> *"We had a situation where a lot of low-quality code was being submitted, so we increased the number of approvals from one to two. Now, two people must review and approve the code before it can actually be merged."* (Interview 7)

P6 also emphasised the structured approval process in their team:

> *"In our team of six or seven, we require at least two green check marks before merging any code."* (Interview 6)

Beyond quality control, several participants stressed the security benefits of code reviews. P1, for instance, stated that it was their main security practice, ensuring compliance with internal standards and protecting sensitive information:

> *"The main security practice we focus on is code review. This ensures that the code adheres to our standards, is properly formatted, and complies with our internal security protocols, especially concerning how keys are handled."* (Interview 1)

P3 noted that a well-structured review process can catch issues early, before any external testing is conducted:

> *"A strong code review process with clear standards or guideline documents helps catch a lot of issues before any testing stage."* (Interview 3)

Similarly, P5 described how team-based code reviews helped manage security in large software projects:

> *"We conduct code reviews within our team, where only those familiar with the specific system section review the code. Since our software is large, keeping track of everything is difficult, so security through team-based reviewing has been really beneficial."* (Interview 5)

Beyond security and quality, some participants valued code reviews for their role in knowledge sharing within teams. P6 highlighted how this practice helps distribute expertise:

> *"Code review is a great way to improve quality and share knowledge within the team."* (Interview 6)

Similarly, P4 viewed it as a foundational practice in software development, suggesting that without it, other security measures would be less effective:

> *"I consider code reviews to be a fundamental practice to start with; otherwise, everything else makes little sense."* (Interview 4)

While the specifics of implementation varied, a common theme among participants was the necessity of involving multiple reviewers, ensuring consistency, and integrating code review as a routine part of secure software development.

Key findings: Manual code review is essential for code quality, security, and knowledge sharing. Participants highlighted the importance of multiple reviewers to ensure thorough assessments, to catch issues early, and maintain security standards. It also fosters expertise sharing within teams and shared awareness.

Manual Penetration Testing Manual Penetration testing (pen testing), different from automated penetration testing, was widely regarded as a crucial security practice by participants, with most considering it one of the most effective ways to identify vulnerabilities. However, there were differing opinions on whether it should be conducted mainly internally or externally, as well as on how frequently it should be integrated into the development process. Several participants highlighted the benefits of hiring external cybersecurity firms to conduct pen testing. P7 explained that larger companies typically engage external auditors to test their deployed solutions:

> *"For bigger companies, we usually hire an auditing or cybersecurity company to pen test our deployed solution. We just tell them, 'Hey, this is the scope. Try to break it, try to do something with it,' and then they create a report advising us, 'This is good,' or 'This is really bad, your certificate is outdated,' and so on. They provide a list of issues, which we then work on."* (Interview 7)

P3 supported the idea of external testing but also suggested that internal testing could be beneficial:

> *"Get external penetration testing done. Or even better, do it internally to help improve the organisation's security."* (Interview 3)

P2 took this idea further by advocating for internal hands-on attack simulations, arguing that actively attempting attacks enhances understanding more than just analysing them:

> *"It's important to understand how attacks happen to defend against them. I recommend carrying out attacks yourself because this gives you a better understanding of attacker methods. Analysing attacks is useful, but actively engaging in them is even more effective for learning. However, in this context, it is not the highest priority."* (Interview 2)

Several participants agreed that penetration testing was one of the most valuable security practices. P3 pointed out that penetration tests often reveal more significant security insights than standard function tests:

> *"Often times, a simple penetration test will give you far more information than small function tests."* (Interview 3)

P4 described how regular penetration tests had improved security awareness at the organisational level, particularly among executives:

> *"In my experience, penetration testing is one of the most effective practices. It significantly increases security awareness in the company, especially among management. We used to conduct them irregularly, but now, due to our enterprise partnerships, we are required to do them annually."* (Interview 4)

P4 further explained that frequent penetration tests consistently reveal vulnerabilities, making them a necessary part of proactive security management:

> *"These regular tests have shown that something is almost always discovered, hopefully nothing critical. This gives us a concrete reason to actively work on security topics. In the normal development process, such issues often get overlooked. That's why I consider penetration testing one of the most important non-technical practices."* (Interview 4)

While most participants endorsed penetration testing, some questioned its practicality in day-to-day development. P7 acknowledged its effectiveness but warned against integrating it too frequently in the development lifecycle due to time constraints:

> *"Pen testing is really up there, it's really good. But if it's external, that's great, they can do it separately. However, I don't think it should be part of the normal development cycle because it just takes too much time."* (Interview 7)

Overall, penetration testing was widely viewed as a critical security practice, with many participants advocating for its regular implementation, either through external experts or internal security teams. However, concerns about its integration into daily development workflows highlight the need for a balanced approach.

Key findings: Manual penetration testing is very beneficial for identifying vulnerabilities and raising security awareness, with participant support for both external and internal testing. While valuable, its time-consuming nature means it should be strategically implemented, rather than as part of the daily development cycle.

Pair Programming Pair programming was a practice that received mixed responses from participants. While some saw it as beneficial for onboarding and maintaining coding standards, others expressed concerns about productivity and responsibility.
Several participants highlighted pair programming as a useful tool for onboarding new developers and ensuring consistency in coding practices. P4 explained that as their company rapidly grew, onboarding new employees efficiently became crucial:

> *"Our company is growing rapidly, with almost a new employee joining every month. It's important that they quickly get up to speed and familiarise themselves with our software."* (Interview 4)

To address this challenge, P4's team implemented pair programming as a way to help new developers integrate faster:

> *"New employees need to understand our practices regarding code quality, coding conventions, and possibly security aspects. We've found that pair programming is the best mechanism to achieve this quickly. Team members sit together for a day, program collaboratively, or design a concept together."* (Interview 4)

Similarly, P7 recognised the benefit of pair programming in ensuring alignment among developers:

> *"Pair programming is always nice if you need to check up on all developers, to make sure everyone is on the same page, like, do we have the same thoughts on how to fix certain things?"* (Interview 7)

Despite its benefits for onboarding, some participants felt that pair programming did not necessarily improve productivity. P7 placed it somewhere in the middle in terms of usefulness:

> *"I'd put stuff like pair programming in the middle. It helps in getting a better overview, but it doesn't really boost productivity. It's good for onboarding, but after that, it should be fine."* (Interview 7)

P6 shared a situational approach, indicating that pair programming can be useful for more complex tasks but is not always necessary for smaller ones:

> *"Since we are a well-coordinated team, I do enjoy pair programming but it depends on the task. When we plan, we know the workload, so for larger tasks, we prefer working together, sometimes even in a group of three. But if it's a smaller task and I don't feel like chatting, I just do it alone. So yeah, we use pair programming from time to time."* (Interview 6)

One criticism of pair programming was that it could lead to a dilution of responsibility. P10 expressed concerns that when two developers work together, accountability might decrease:

> *"I'm not really convinced by pair programming because it can lead to a lack of responsibility. If you have two people making mistakes, each one might blame the other. While this isn't part of our culture, the incentive exists. I like the feeling of knowing that responsibility truly lies with one person."* (Interview 10)

Some participants noted that pair programming worked well in combination with code reviews. P4, for example, saw code reviews as more effective than pair programming in improving security awareness:

> *"More important for us are practices like code reviews, where we discuss potential security vulnerabilities and create awareness of common mistakes and how developers can avoid them."* (Interview 4)

However, P4 also acknowledged that pair programming was something their team had recently begun experimenting with:

> *"In recent weeks and months, we've introduced or regularly experimented with pair programming. Previously, each developer worked individually on their specific tasks, followed by code reviews."* (Interview 4)

Pair programming was seen as a valuable practice for onboarding new developers and ensuring consistency in coding and security standards. However, opinions on its effectiveness for productivity and security were mixed. Some participants felt that it was useful only in specific situations, while others preferred code reviews as a better security practice. Additionally, concerns were raised about diluted accountability, suggesting that pair programming might not be suitable for all teams or all types of tasks.

Key findings: Pair Programming was valued for onboarding, ensuring coding consistency and security standards but had mixed opinions regarding productivity. Some felt it didn't significantly boost output and could reduce accountability through shared responsibility. While useful for complex tasks or team alignment, many preferred code reviews for security and efficiency.

Security Champion/Team The concept of a Security Champion or dedicated security team was widely recognised as a valuable non-technical practice among participants. While its implementation varied across organisations, many practitioners highlighted its effectiveness in bridging the gap between security and development.
Several participants emphasised the benefits of having designated security personnel embedded within development teams. P8 described the advantages of this approach, noting the dual role of security specialists:

> *"Each product line has its own security personnel, someone who handles the development team and serves as both a developer and a security specialist. This dual role functions really well, and I think it's a great quality here."* (Interview 8)

Beyond individual product lines, some organisations also employ centralised security oversight. P8 explained how this structure enhances security coordination across different teams:

> *"In addition to that, there is also a person responsible for the overall security of the entire product. However, they tend to be more focused on security rather than development. I think that's one aspect of the process that works really well here, where you have multiple people looking into different aspects and trying to bridge the communication."* (Interview 8)

The idea of a Security Champion, a designated team member responsible for security within a development team was also viewed as beneficial. P9 emphasised its role in maintaining security awareness and leading initiatives:

> *"Another good idea, I think, is to designate someone in the team as responsible for security, a so-called Security Champion, who focuses specifically on security. This person is responsible for driving the security system forward and holding relevant meetings."* (Interview 9)

In addition to dedicated security personnel, some organisations invest in security research teams to stay ahead of emerging threats. P8 highlighted this proactive approach:

> *"That's also a good thing here, they have a research team that continuously looks for new ways to stay ahead of the curve. This is a positive aspect of the process, as they have a dedicated research unit focused specifically on exploring what more can be done in terms of security and how they can improve."* (Interview 8)

Security teams or individuals also support development teams by working to ensure proper documentation and best practices are enforced, as they play a critical role in maintaining security standards and effective responses. P8 emphasised the importance of security teams keeping security guidelines up to date:

> *"Concise documentation is essential because often, people create one massive document that contains everything. As a result, you end up sifting through numerous pages to find the information that actually applies to you. At Company A, our security team is actively working to pull up old documents, update them, and keep them current. This practice is crucial for effective communication and clarity."* (Interview 8)

Collaboration between development teams and security specialists was another recurring theme. P1 and P3 noted that security teams provide guidelines and oversight, ensuring security is integrated early in the development process:

> *"We have a dedicated security team, and our role is to follow templates and best practices they've implemented."* (Interview 1)

&

> *"We often partner with our enterprise security team frequently on the projects that we work on and ensure that they're brought in early on to give them plenty of headroom to review any security controls as needed and provide recommendations that we can implement before we start the project."* (Interview 3)

However, some participants noted that security teams primarily focus on higher-level metrics and compliance requirements rather than direct implementation. P1 reflected on the scope of their security team:

> *"It's probably more within the scope of the Company B security team, since we're large enough that they implement metrics on all of our security processes."* (Interview 1)

Finally, security teams also play an educational role, ensuring employees follow security protocols in day-to-day operations. P5 highlighted the importance of security awareness training:

> *"Yeah, they do. They provide principles we should follow, like understanding the language used in emails and who's asking. For example, if you don't know the name of the person asking for something, you should be suspicious. They ensure we're following the right protocols."* (Interview 5)

Overall, responses indicate that having dedicated security personnel or teams significantly enhances security and adherence to security practices within organisations. Whether through Security Champions, centralised security oversight, or dedicated research teams, these roles help bridge gaps in understanding, implement best practices, and maintain security awareness across development teams.

Key findings: Security Champions and dedicated teams improve security awareness, guide best practices, and ensure early integration and adherence to security measures. They collaborate with development teams, stay ahead of threats, and educate employees on protocols, enhancing overall security.

Why are these Non-technical practices Although each of these practices involves technical expertise, we consider them as non-technical because they fundamentally rely on human collaboration, judgment, and communication.
These practices were all designated as being non-technical by the participants and in all cases, the human element is necessary, making these practices more about process and interpersonal engagement than about pure technical automation.
Manual code reviews, for instance, depend on developers discussing and critiquing code to catch issues that automated tools might miss, spreading awareness and helping unify coding

Non-technical Practice	Category	Hindering	Not Useful	No Opinion	Useful	Very Useful
Incident Response Plan	Used	0	1	0	5	3
Incident Response Plan	Not used	0	0	0	1	0
Security risk management plan	Used	0	0	0	3	2
Security risk management plan	Not used	0	1	0	4	0
Security Training	Used	1	1	0	6	1
Security Training	Not used	0	0	0	1	0
Analyse the attacker profiles	Used	0	2	0	0	0
Analyse the attacker profiles	Not used	0	2	3	3	0
Coding standards	Used	0	0	0	2	8
Coding standards	Not used	0	0	0	0	0
Security test coverage measurement	Used	0	0	0	4	3
Security test coverage measurement	Not used	0	0	1	1	1
Customer release risk acceptance	Used	0	0	0	0	2
Customer release risk acceptance	Not used	1	2	5	0	0
Ethics Hotline	Used	2	0	0	1	0
Ethics Hotline	Not used	2	2	2	1	0
User Data Access Controls	Used	1	0	0	3	5
User Data Access Controls	Not used	0	0	0	1	0
Total		7	11	11	36	25

Figure 2: Rate the practice: results table

approaches through the communication of coding standards. Similarly, manual penetration testing uses a tester's creativity and intuition to simulate real-world attacks not limited to parameters set for technical penetration tests. Pair programming emphasises collaborative problem-solving and knowledge sharing, while security champions or teams focus on embedding a security culture through engagement, responsibility and education.

4.5 RQ3: "Which non-technical practices are perceived as effective and ineffective for practitioners?"

Rate the practice

In the exercise ten out of eleven participants were asked to rank representative non-technical practices on a scale determining their perception of their usefulness on a five point scale in contributing to security and the general development process. . Evaluating the perceived effectiveness of the representative non-technical practices in order of most positive perception in adding benefits to the development process according to participants. Figure 2 (Fig. 2) represents the results for the Rate the practice exercise. It separates each of the selected non-technical practices on a five-stage usefulness scale and into whether the practices was used or not as can be seen in the category column. This means that a participants can have a positive opinion on the effectiveness of a non-technical practices but currently not employ it in their organisation. A row representing the total perception of non-technical practices based on the selected practices was also included but not separated by usage.

4.5.1 Coding Standards

Coding standards were used by all participants in the study (Fig. 2) and were widely regarded as the most effective non-technical practice in the exercise. Overall, 80% of participants rated them as "Very Useful" in positively contributing to the development process, while the remaining 20% found them "Useful." This indicates a consistently positive perception of coding standards across organisations of different types and sizes and among participants in various roles. P9 described the utility of coding standards but cautioned against excessive enforcement:

> *"I would classify coding standards as useful, but you have to be careful not to overdo it. I've seen situations where a linter was used to check compliance with the standards. Some aspects just shouldn't be overdone. Overall, however, coding standards are definitely a meaningful practice if applied correctly."* (Interview 9)

P2 emphasised the value of coding standards for improving security, highlighting their reliability and effectiveness:

> *"Adhering to coding standards is certainly useful for security. Standards are proven and ensure that security aspects are taken into account. 'Robust' in this context means that the standards have proven to be effective, which is ultimately crucial for security in software development."* (Interview 2)

These statements reflect the strong consensus among participants regarding the benefits of coding standards. When applied appropriately, they enhance not only the quality and consistency of code but also the overall security and reliability of the development process.

4.5.2 User Data Access Control

User Data Access Control was rated as the second most effective non-technical practice by practitioners. It was implemented by 90% of participants (Fig. 2) in their development processes, with 60% rating it as "Very Useful" and the remaining 40% as "Useful." Several participants emphasised the importance of restricting access to only what is necessary to perform a specific role. For instance, P1 highlighted the significance of access policies, stating:

> *"As for access policies, they're probably the most important aspect, people should only have access to what they need."* (Interview 1)

Similarly, P5 elaborated on the importance of minimising control to prevent catastrophic errors:

> *"In non-technical practices, it's really about understanding that having the least amount of control available to you is the safest route. For instance, if someone has full access to the production database and starts dropping tables, it's a disaster. So, maybe give them read-only credentials instead. That way, you've implemented a practice that helps ensure you don't disrupt running systems."* (Interview 5)

However, opinions were not universally positive. P7 expressed frustration, describing User Data Access Control as counterproductive:

> *"User data access controls, that's like the worst thing. It's hindering. It's like a very easy way to do it to say it, and we don't use it at all."* (Interview 7)

P4 offered a more nuanced perspective, noting that the relevance and implementation of User Data Access Control depend heavily on the company's stage of development:

> *"I'd say User Data Access Control also strongly depends on the stage of the company. We are at a point where it's becoming relevant and important to implement this strictly. However, it's also a trade-off because people need access to certain things, especially in Customer Care when they have to log into accounts and so on. So, it strongly depends on the company's stage."* (Interview 4)

These responses highlight the varying perspectives on User Data Access Control. While many practitioners recognise its value in improving security and minimising risks, others view it as a hindrance to productivity or see its implementation as highly context-dependent.

4.5.3 Incident Response Plans

Incident response plans were widely adopted and positively regarded by practitioners in the study as can be seen in Fig. 2. The majority of participants rated them as either "Useful" (60%) or "Very Useful" (30%) in their contribution to the development process. However, one participant classified them as "Not Useful," citing specific limitations. P10 highlighted the value of having an incident response plan to reduce dependency on individual team members, stating:

> *"Yes, exactly, we have one. It's definitely useful to have such a plan because it means you're not solely reliant on a single employee. I would classify it somewhere between 'useful' and 'very useful.'"* (Interview 10)

Similarly, P9 pointed out the importance of having a plan in place to provide structure and clarity during stressful situations:

> *"When an incident occurs, it's important to know how to act. Under stress, having a clear plan can be very helpful."* (Interview 9)

P2 further stressed the necessity of preparing for the worst-case scenario and having a clear response strategy to address emergencies:

> *"Exactly, we start with the worst-case scenario. It means that anyone can make a mistake, no matter how skilled we are. We absolutely need this plan in case something goes wrong. The most important thing is having the plan in place when the situation escalates, when the house is burning, so to speak. It just has to be there."* (Interview 2)

However, not all participants viewed incident response plans favourably. P6 questioned their practicality, noting that the variability of incidents often limits the usefulness of predefined plans:

> *"I guess, yeah, but I'd say incident response plans are not really useful because incidents vary a lot. What's the point of the plan if you have to look at each situation individually? The problems are different, and if you knew what problems to expect, you'd solve them beforehand. So, most of the time, it's not that helpful."* (Interview 6)

These perspectives reveal a largely positive reception of incident response plans, with most practitioners valuing their ability to provide structure and reduce stress during emergencies. However, some concerns remain about their adaptability to highly variable situations, suggesting that their usefulness may depend on the context and the nature of incidents encountered.

4.5.4 Security risk management plans

Security Risk Management Plans were perceived as effective by the majority of participants, with 80% rating them as "Useful" and 10% as "Very Useful." Despite these positive perceptions, their implementation was less common, with only half of the participants reporting that such plans were used within their organisations (Fig. 2) . This indicates a gap between recognising the value of risk management and consistently applying it in practice. Several participants emphasised that Security Risk Management Plans are crucial for identifying potential issues early in the development process. P5 highlighted their role in providing a broader perspective, allowing teams to anticipate future problems by considering the larger context of development activities:

> *"It ties into risk management, understanding the problems. Some developers don't like stepping back and planning, but understanding the scope of changes and bugs is crucial to see knock-on effects."* (Interview 5)

Supporting this view, P3 noted that even basic risk preparation in technical documentation can help teams assess potential risks associated with handling sensitive data:

> *"We do some level of risk preparation in our technical documentation to understand what risks are associated with the project, like handling customer or employee data."* (Interview 3)

In larger organisations, the implementation of Security Risk Management Plans often relies on dedicated compliance teams. P7 pointed out that these teams play a key role in enforcing such practices:

> *"In larger companies, you can have a compliance team that sets these things up and makes everyone go through them. I'd say it's useful, and it's still in use."* (Interview 7)

However, not all participants viewed the implementation as effective. P9 argued that while risk management is conceptually important, it is often not sufficiently considered or made accessible to developers:

> *"I believe security risk management is often not sufficiently considered or not summarised in a way that makes it tangible for people."* (Interview 9)

This sentiment was echoed by P0, who linked effective risk management to a company's understanding of its most critical assets:

> *"An easy example, I like to give is you have a Baker, right is most important thing is baking bread and then selling the bread, right? That's the two things he needs to be doing. So if he gets hacked and the I don't know, the ovens go down,*

there should be a backup oven, you know, because he needs to sell bread. And if the banking system goes down, he needs to have some other way. He needs to have a cash register where you can pay with cash. You know that kind of stuff." (Interview 0)

In summary, while participants widely acknowledged the value of Security Risk Management Plans in identifying and mitigating risks, challenges remain in ensuring their consistent application, particularly in smaller organisations where dedicated compliance structures may be lacking. The feedback also highlights the need for clearer, more accessible approaches to risk management to increase its practical relevance across different organisational contexts.

4.5.5 Security trainings

Security trainings were the most widely used in participants organisation with only one not employing it (Fig. 2). 70% saw them as "Useful", one as very useful, yet one participant did describe them as "Not Useful" and one even as hindering. Security education received mixed reactions from participants. Some found it ineffective, while others highlighted its critical importance to security and development when properly implemented. A total of five participants expressed scepticism about its impact, particularly when it is poorly applied and through this lacks engagement by participants. P7 emphasised that security education often fails because people revert to old habits unless they are personally motivated to change:

"For many people, I don't think training or anything else really works. People just fall back into old habits. They need to be personally open to developing in a way that is objectively beneficial or aligns with what the team agrees on." (Interview 7)

Similarly, P8 shared an experience where hiring external security education firms did not lead to noticeable improvements:

"I once spoke to someone who mentioned that companies often hire specialised security training firms, but they found that the results before and after were the same. There was no noticeable benefit from the training. Honestly, I partly agree with that." (Interview 8)

However, some participants argued that security education is not inherently ineffective, but its implementation is often flawed. P3 explained that the delivery method determines its effectiveness:

"Security training, I would say, has been ineffective in the past. But it depends on how the training is conducted. If it's just a series of basic online videos saying, 'As a business, we require you to take this training,' people will do the bare minimum to complete it or try to bypass it entirely." (Interview 3)

P4 highlighted that keeping security education short and engaging improves its impact:

"The key thing is that our training is kept short and engaging, provided by an external company. I think that's a good approach. But ultimately, it's about creating a basic level of awareness, people don't need to become experts." (Interview 4)

Despite recognising its value, some participants found certain aspects of security education frustrating, particularly repetitive topics such as phishing awareness. P6 expressed this sentiment:

> *"Security training... I guess it's useful, but it annoys me. The phishing stuff is always so tedious. We've had these sessions over and over again."* (Interview 6)

These perspectives suggest that while security education is widely implemented, though its effectiveness depends on engagement, relevance, and delivery methods. Despite concerns about effectiveness, six participants strongly emphasised that security education is essential for improving awareness and security preparedness. P9 noted that many developers lack formal training in security, making structured education necessary:

> *"Training people is essential. Just because someone can programme doesn't mean they've had formal training in software security, especially if they are self-taught."* (Interview 9)

Beyond technical skills, some participants stressed the importance of security awareness, particularly in areas like social engineering. P2 highlighted that human vulnerabilities often outweigh technical risks:

> *"Social engineering training is absolutely essential learning how to protect yourself from being manipulated by others. This is often even more important than technical measures or firewalls. Social engineering is one of the biggest threats."* (Interview 2)

In addition to awareness, participants also pointed out that security education should be tailored to the specific needs of developers. P3 recommended that teams take structured courses that focus on relevant technologies and past security vulnerabilities:

> *"The focus should be on properly educating the team, getting them to take structured courses on the languages and frameworks they use, as well as past security vulnerabilities."* (Interview 3)

These insights suggest that security education is most valuable when it is structured, relevant, and focuses on both technical and behavioural aspects of security. Several participants described hands-on approaches that help reinforce security awareness. P4 shared that post-mortem discussions after security incidents are an effective way to learn from real-world mistakes:

> *"After any major incident, whether it's a system failure or a security breach, we always conduct post-mortems. We explain exactly what happened. It's important that these incidents don't happen in isolation but are shared with the whole team. We go through the timeline, discuss our response, and consider alternative approaches. This is crucial for raising awareness, as the average developer doesn't typically encounter critical security incidents in their day-to-day work. And ideally, such incidents shouldn't happen too often."* (Interview 4)

Similarly, P6 described how regular tech sessions help reinforce security awareness by discussing security-related topics as a team:

> *"We have a tech session every two weeks. What is that? We meet as a team, note down topics we want to review together, or go over something someone has implemented. Sometimes, only two people have seen the change during a code review, but it's something central to our work, so we go through it as a team."* (Interview 6)

These approaches highlight that security awareness is most effective when it is integrated into regular workflows, rather than being a separate, one-off activity. Beyond structured education, some companies build security awareness by gradually increasing employees' responsibilities. P1 explained how their manager supports skill development while allowing employees to take on new challenges:

> *"My new manager has been excellent at what they call 'release of responsibility' from the educational world. They encourage me to take on tasks that match my abilities but also push me slightly outside my comfort zone. This approach allows me to explore new responsibilities while still having the support to test my understanding and growth."* (Interview 1)

This suggests that progressive responsibility and hands-on experience can help employees build a lasting security mindset beyond traditional education methods. While security education is widely recognised as necessary, its effectiveness is debated. Some practitioners believe that choice of focus and poorly structured training reduce its impact, while others argue that customised, engaging, and practical approaches make a significant difference. Hands-on methods such as post-mortems, tech sessions, and structured learning programmes were seen as more effective than generic training videos. Ultimately, security education is most impactful when it is continuous, context-specific, and actively encourages responsibility.

4.5.6 Customer release risk acceptance

Customer release risk acceptance was a practice that was seldom employed by practitioners (Fig. 2) and had a very mixed rating. Three participants who did not employ the practice rated it as negative with two placing it in "Not Useful" and one in hindering, while two practitioners who did employ the practice placed it in "Very Useful". Due to this the practice can be categorised as positive, yet due to the large number of abstentions by the rest of the participants this is hard to determine for sure. Participants highlighted the main advantage of this approach: shifting residual risk from the company to the customer. P9 described how transparency about security risks plays a crucial role in this process:

> *"It is very sensible to communicate security aspects transparently, especially regarding the risks that exist. The question, of course, is how to handle these risks. Essentially, the residual risk is transferred to the customer, which could be beneficial for the company if the customer accepts it well. However, it is difficult to make a general judgement on this."* (Interview 9)

Similarly, P10 explained that while they do implement this approach, it is often driven by time constraints and customer demands:

> *"Yes, we do this. At the beginning, we assess the risks, and if things need to move quickly, customers sometimes give us requirements that are difficult to implement. We then communicate clearly that this is suboptimal. It is probably very sensible because the responsibility no longer lies with us."* (Interview 10)

However, other participants raised concerns about the practicality of this approach. P3 argued that Customer Release Risk Acceptance is more relevant for contractors or vendors rather than in-house development teams:

> *"Yeah, I wouldn't be able to do that because I think that seems like it's more for a contractor or vendor providing software."* (Interview 3)

Likewise, P2 pointed out that while it may be beneficial for businesses, it could be disadvantageous for customers:

> *"As a business owner, I find this very useful, whereas for the buyer, it could be more of a hindrance. I can't really say in what situation this would be useful for the end customer, but for me as a business owner, it definitely makes sense."* (Interview 2)

In summary, Customer Release Risk Acceptance remains a highly polarising practice. While some practitioners see it as a beneficial way to delegate responsibility, others view it as impractical or disadvantageous for end users. The mixed perspectives highlight the importance of context, while it may work well in some business models, it is not universally applicable.

4.5.7 Analysis of attacker profiles

The analysis of attacker profiles was a non-technical practice that was employed by less than half of the participants (Fig. 2) and was perceived as very mixed, but in total with a negative perception when contributing to the development process. Many participants (30%) did not have an opinion, 40% saw the practice as "Not Useful", while 20% saw it as "Useful" though it has to be observed that the people the participants describing it as "Useful" did not employ the practice while half of those placing it in "Not Useful" the practice where the ones currently employing it.
Many participants viewed the analysis as unnecessary within their development environments. P7 expressed this sentiment clearly, highlighting that such activities fall outside the core focus of their team:

> *"Analysing attacker profiles, that's something I think external cybersecurity providers should do. For us, it's not useful; it just wastes time. We don't do that. Either we fix the issue, or we don't understand how the attacker got there. That's someone else's job."* (Interview 7)

Similarly, P9, who had direct experience with attacker profiling, found the practice limited in its practical application, noting that it did not significantly enhance preparedness:

> *"I don't find the analysis of attacker profiles particularly useful. I've applied it myself and realised that you should prepare for all types of attacks, regardless of the profiles you create. It's interesting for understanding attack vectors, but I don't think you should rely on it."* (Interview 9)

However, some participants acknowledged potential value in specific contexts. P8 pointed out that industries like banking benefit more from attacker profiling due to the direct correlation between security breaches and financial loss:

> *"Some companies, especially banks, focus on profiling attackers because they can immediately link vulnerabilities to monetary losses. For product companies, it's harder to quantify the impact since losing user data doesn't have the same immediate effect as losing bank account details."* (Interview 8)

Despite this theoretical value, P8 also admitted uncertainty about its practical implementation, citing resource constraints as a significant barrier:

> *"It seems like it could be really useful in theory, but in practice, it's hard to do unless you have the right data and resources to track patterns effectively. It sounds valuable, but implementing it well is a different challenge."* (Interview 8)

Lastly, P4 emphasised that broad, comprehensive security measures are often prioritised over detailed attacker profiling, as the latter may not significantly reduce risk across diverse attack surfaces:

> *"You have to cover everything anyway. We focus more on broad protection rather than analysing specific attacker profiles. That's actually more important in my view."* (Interview 4)

In summary, while the Analysis of Attacker Profiles holds theoretical appeal, its practical value is questioned by many practitioners. In most environments broad security coverage was prioritised due to the complexity of implementation and the diversity of attacks faced by organisations. The practice appears more relevant in high-risk industries like finance, but for most participants, the resource investment required outweighs the perceived benefits.

4.5.8 Ethics Hotline

Ethics hotlines were a seldom employed non-technical practice, as can be seen in Fig. 2, with only three participants indicating they were used in their organisation. They were the worst received practice with the highest numbers of categorisation as "Hindering" (40%) and for "Not Useful" (20%), though they did receive categorisations as "Useful" by two participants, one of which employed the practice. A strong sentiment of rejection was evident among participants who viewed the practice as unnecessary or even detrimental. P10 was particularly dismissive, associating the implementation of ethics hotlines with organisational failure:

> *"Such nonsense. Companies that introduce this will soon go bankrupt. No, I think it's bullshit. We've never used it, and I'd categorise it as completely hindering."* (Interview 10)

Similarly, P7 found the concept counterproductive, suggesting that ethical concerns should be resolved through direct communication rather than formal reporting channels:

> *"I feel that's more like hindering. Just talk to people, and it's going to be fine. I don't need to talk to HR. No one needs that."* (Interview 7)

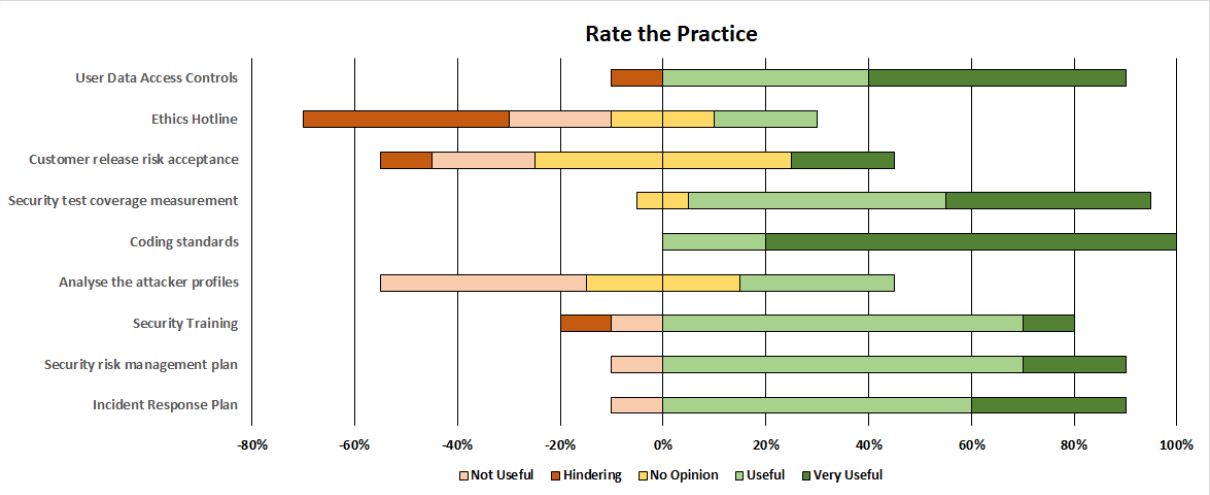

Figure 3: Rate the practice: effectiveness Likert chart

Even participants with a more neutral stance expressed doubts about its effectiveness in practice. P9 acknowledged the idea's theoretical value but questioned whether employees would genuinely feel comfortable using it without fear of repercussions:

> *"I think the idea is fundamentally good, but I'm not sure it's practical. I'd rate it as not very useful because I think people probably wouldn't use it. And if they did, they might get into trouble with their boss."* (Interview 9)

In some cases, organisations had implemented alternative mechanisms, but these were often minimalistic. P1 mentioned having a basic whistleblower system, though it lacked the formal structure of an ethics hotline:

> *"The ethics hotline, not very useful in my opinion. We do have a kind of whistleblower option, but it's more like an email system. If you have a concern, you just send an email."* (Interview 1)

However, not all feedback was entirely dismissive. P4 saw potential value in large organisations where hierarchical barriers might hinder open communication, suggesting that such hotlines could serve as an alternative route to report managerial misconduct:

> *"I'd say it's more for very large companies. If you need something like that, there's already something seriously wrong in the organisation. But it's good to have an alternative contact method when managers aren't doing their jobs properly."* (Interview 4)

In conclusion, ethics hotlines were largely viewed as irrelevant or counterproductive, especially in smaller or more informal development environments. While there was some recognition of their potential role in larger organisations with complex hierarchies, the prevailing perception was that open communication channels within teams rendered such formal mechanisms unnecessary.

4.5.9 Exercise conclusion

In total we can see that most non-technical practices were ranked in the realm of "Useful" and "Very Useful" as can be seen in Fig. 3. Four practices received a categorisation as "Hindering" by a participant, with one receiving 57% of the total "Hindering" votes and only a 30% usage rate (Fig. 3) , which could point towards a bias against the individual practice and not necessarily the category it represents. Votes for practices categorising them as "Not Useful", were also low in comparison to positive votes, though in this category there was a close to even split between practitioners that used the practice and those that did not, making it more representable.
From the results we can also see that most exercises that received positive ratings were also more likely to be employed by practitioners, considering a few notable exceptions (Fig. 2) . This could show that organisations adopt practices that they perceive as contributing positively to their development more often, or that they are seen as useful as their benefits have been experienced by them. The total as seen in Fig. 2 also provides a positive perception of the selected group of non-technical practices, with "useful" being the highest scoring category followed by "very useful" which were significantly higher than the neural or negative scores.

Key findings: These results show an overall appreciation by practitioners for the abilities of non-technical to positively contribute to the development process and highlights some practices and possible categories that can be considered essential and nearly standardised in modern software development according to the results of participants ratings. Other practices received negative evaluations combined with described disuse, while some were only applicable to organisations with certain organisational structures or project requirements.

4.5.10 MIRO

Without prior introduction or suggestions of possible non-technical practices, participants were asked to name and rank the most important non-technical practices based on their opinion of effectiveness to positively contribute to development.
These results were then grouped according to the context they were named in and were then assigned to categories, some practices are found in multiple categories as they were named such context during the interviews. These categories were then ranked in Table: 2 according to the ranking of original named practice named and the frequency they were named across different participants.
Participants were also asked to give reasoning for the ranking and the importance for the practice they perceived as non-technical and most important, as well as evidence for this. These are the combined responses of the practices per category and their provided evidence for their importance.
These results in Table: 3 show the awareness of participants to non-technical practices in their SDLCs and the broad range of practices in use. The differentiation in terms for different non-technical practices shows a common lack in unified knowledge on the matter, highlighting a more casual approach on the implementation of these practices not based on academic or industrial definitions.

Awareness and Education
Raising awareness and providing education ensures that teams understand security risks, follow best practices, and use tools effectively. Frequent human errors, such as phishing attacks (the

Rank	Category	Practices Included	Common Theme
1	Awareness and Education	Awareness, Security education, Peer review, Security champion, Security metrics in pipeline, Tech sessions, Training, Knowing your tools	Focus on raising awareness and educating individuals on best practices, security, and industry standards.
2	Risk Management and Planning	Risk management, 3rd party risk management, Business continuity plan, Planning, Access control, PMP, Security team, Pen testing	Emphasis on preparation, understanding potential risks, and creating plans to mitigate them.
3	Documentation and Process Structuring	Documentation, QA process, Code review, Continuous integration	Maintaining proper records and structured processes for better project and task management.
4	Team and People Management	Team building, People (ensuring overlap), Equal footing on vision/aims, Shared work approach, No single points of failure, Time management techniques, Acknowledgement / Gamification	Strategies to improve team cohesion, communication, and alignment within a group.
5	Standardisation and Best Practices	Coding standards, Minimum security practices, Testing, Following government regulations, Baseline of security practices, Automated testing	Maintaining consistency and ensuring a baseline of quality and security through established practices.
6	Feedback and Continuous Improvement	MVP for customer feedback process, Feedback structuring, Continuous integration	Incorporating feedback and iterating on work for continuous improvement.

Table 3: Miro Results

most common form of cyber-attack), highlight the need for security awareness. Regular training, peer reviews, and knowledge-sharing sessions have proven to foster a culture of security and reduce mistakes.

Risk Management and Planning
Proactively managing risks and planning ensures the organisation is prepared for challenges and compliant with regulations. Government fines for third-party risk mismanagement and non-compliance demonstrate the importance of structured risk management. Business continuity plans are supported by statistics showing improved recovery rates. Penetration testing consistently uncovers areas for improvement, and experience proves that planning and access control mitigate large-scale issues.

Documentation and Process Structuring
Clear documentation and well-structured processes reduce errors, ensure consistency, and enable faster issue resolution when things go wrong. Real-world experience shows that manual code reviews, QA processes, and continuous integration improve adherence to standards, catch issues early, and maintain high-quality deliverables. Proper documentation significantly reduces downtime during failures.

Team and People Management
Effective team and people management promotes resilience, collaboration, and productivity. Overlapping skills, shared goals, and eliminating single points of failure prevent knowledge loss during personnel fluctuations, as experience consistently proves. Techniques like time management (e.g., tools like JIRA) improve focus and stamina, while acknowledgment and gamification boost morale and productivity. Team building further enhances communication and team efficiency.

Standardisation and Best Practices
Establishing standards and best practices ensures security, quality, and compliance. Government regulations enforce fines for non-compliance, making adherence essential. Coding standards, best practices and code reviews consistently reduce errors, improve code quality, and prevent security gaps. Experience confirms that following baseline security practices and enforcing minimum security guidelines avoids sacrificing security for faster delivery.

Feedback and Continuous Improvement
Continuous feedback and improvement processes lead to better product outcomes and adaptability. User engagement improves when MVPs provide tangible, interactive elements, motivating users to provide actionable feedback. Continuous integration ensures early detection of errors and maintains high-quality standards. Experience proves that structured feedback processes foster innovation and align deliverables with user needs.

Conclusion
Much of the evidence provided concerning non-technical practices was based on participants experience and anecdotal insights. Participants demonstrated a strong understanding of the importance of their non-technical practices, offering logical reasoning for their perceived value, even if quantifiable evidence was lacking. The findings highlight a generell but fragmented understanding of non-technical practices in SDLCs, emphasising the need for greater awareness

and structured implementation. Key areas such as security awareness, risk management, documentation, team management, standardisation, and continuous improvement all contribute to a more resilient and secure development process. Practitioners supported the idea that fostering education, enforcing best practices, and encouraging structured feedback, organisations can enhance security, compliance, and overall software quality.

Key Findings: Although a wide array of non-technical practices are employed within organisations SDLC, the exercise revealed a clear emphasis on raising awareness and education (Table 2). Participants ranked practices such as security training, peer reviews, and security champions most highly, underscoring their importance in security and mitigating human error. They also emphasised proactive risk management, thorough documentation, effective team management, standardised best practices, and continuous feedback (Table 2). However, the varied terminology and reliance on anecdotal evidence indicate a lack of unified standards, suggesting that these crucial practices are frequently implemented in an ad hoc manner rather than following established academic or industrial frameworks.

4.6 RQ4: "What are the most common challenges faced by organisations when adopting non-technical software practices?

Cost of Non-technical practices (RQ4)
Tough non-technical practices are acknowledged as positively contributing to software development, in terms of security and otherwise as reiterated in the conducted interviews, there is often only a limited number of these practices adopted by organisations into their development processes. This begs the question as at what is stopping these organisations from implementing more of these practices to improve their development and subsequent products. In our interviews we discovered that among the obstacles to implement non-technical practices in organisations are factors in the realm of: organisational, money, time, people, complexity and non-compliance.

4.6.1 Funding and Resource Constraints

The largest theme that was established across the majority of participants was the significant challenge posed by funding and resource limitations in implementing non-technical practices into organisations development process. Participants frequently highlighted that a lack of resources was a major obstacle for their organisations in doing so. P8 explained his experience with the push by organisations to introduce non-technical practices and the resource limitations they face preventing them from such:

> *"A lot of organisations want to do the right thing, but they lack the resources and the money to do it. Currently, there is a non-technical practice that we want to set up, but we lack the money and the resources to do that."* (Interview 8)

This sentiment was echoed by many participants, who stated that, given adequate funding, they would readily adopt more non-technical practices to improve their development processes. When asked about obstacles preventing the implementation. P9 added that the obstacle of long-term funding that often leads to high costs of non-technical practices:

> *"I think that there's often a lack of money and time. In many projects, it might not even be strictly necessary, but in larger projects, the question arises of who bears the costs for continuous maintenance and updates. That can be a significant obstacle."* (Interview 9)

This highlights not only the issue of initial implementation cost but the compounding expenses that appear over extended periods create significant obstacles not only to their adoption but sustained employment.
What should also be considered is that the allocation of available resources occurs at an organisational level, where there might be conflicting interests in the name of profitability. P0 emphasised the limited willingness of organisational to make investments in the security of their development process and non-technical practices in it, stating:

> *"Companies tend to invest just enough in security, only as much as is necessary to avoid potential financial losses. But if there's a risk that could cost them more, that's when they'll increase security measures."* (Interview 0)

Similarly, P3 underscored how prioritisation of resources within an organisation often determines whether non-technical practices are seen as necessary and subsequently are adopted:

> *"It all costs money, so it's really just based on your Chief Security Officer, or whoever is at the company, to determine that priority and see how much we're going to spend on security."* (Interview 3)

Further explaining:

> Regular pen testing is nice, but if you don't have a normal system for it, it is pretty expensive. (Interview 3)

These insights highlight how resource constraints often lead organisations to prioritise immediate business needs over non-technical practices, despite their potential long-term value. Practitioners are acutely aware of the resource demands associated with non-technical practices, from implementation to ongoing maintenance. Many believe that the challenges for adoption stem from organisational factors and their leaderships view that these resources could be better allocated to cheaper technical tools and practices, which typically yield more immediate and tangible benefits.

Key findings: Funding and other resource constraints limit the adoption of non-technical practices. Companies often prioritise immediate needs and cost-effective technical solutions, investing in long term non-technical security only when faced with significant risks or additional requirements set by leadership or customers.

4.6.2 Time

The second largest factor mentioned by an equally high number of interview participants was a lack of time in their development process to integrate non-technical practices.
Many developers complained about the short development timelines they face and how it affects their desire to produce securer and higher quality work. As P3 puts it:

> *"Yeah. Give me all the time in the world and I'd be happy to do anything to make sure that we have zero security incidents and vulnerabilities. I think most people would be happy to do that." (Interview 3)*

This is mirrored by P6 stating:

> *"Normally, developers would actually like to do this, because they want to write good software and feel confident that when they push something live, it won't break. But when the boss says it has to be finished by tomorrow, that makes it difficult." (Interview 6)*

This shows that developers want to produce good and secure software and would love to implement non-technical practices but are unable to do so due to tight deadlines forcing them to focus on output producing practices. As P10 states an example of the dilemmas he faces on implementing practices in projects with short deadlines:

> *"You can certainly implement high level practices to ensure quality and so on, but ultimately there's simply not enough time. I'd like to make it nicer, but as I said, the festival is mid-February, and we need to be done by mid-January. The contract will be signed next week. One wonders where you would even fit in proper code reviews, because it's going to be a really stressful project." (Interview 10)*

In contrast, from leadership position concerning a balance between security and timely delivery P4 stated:

> *"Definitely. It's (security culture) ideal if you have something like that, but I think it's also hard to establish. It's always a balancing act. As a company, you only have limited time, and you have to weigh how much you're willing to 'sacrifice' for security." (Interview 4)*

Finding this balance is of investment and output was commonly mentioned by participants when consulted on why they did not implement more non-technical practices. Participants stated non-technical practices require time to be completed and do often not show immediate results, but rather bring improvement over a longer period of time. This makes them hard to justify and puts them first on the chopping block when time is a missing resource. As put by P2:

> *"A major challenge is that in stressful projects, one tends to prioritise technical practices due to time pressure. In such situations, people often think short-term and neglect the non-technical practices that are important in the long run. When stress is high, people fall back on what is quickly implementable and achievable, following the motto: "It doesn't matter, we just have to get this done now," with a focus on time and costs." (Interview 2)*

P1 supported this concern but framed the issue as a choice between addressing these costs early or dealing with greater challenges later, explaining:

> *"Well, the problem is that it takes more time (to develop products). But it's either time now or time later, your choice." (Interview 1)*

These responses reveal that even when the benefits of non-technical practices are recognised, they are not always perceived as providing a sufficient return on investment for the time invested. Instead, they are sometimes regarded as a diversion of time and effort that could be more effectively applied to technical practices offering visible, immediate improvements necessary when faced with short development times.

Key findings: Time constraints were a major obstacle to adopting non-technical practices. Developers wanted to implement security measures but faced tight deadlines, leading to the prioritisation of technical tasks with more immediate benefits. Non-technical practices, offering long-term benefits, were often overlooked in favour of short-term solutions in the face of tight development deadlines which shifted development priorities.

4.6.3 Lack of Developers

Closely tied to the issues of time and money is the challenge of lacking sufficient personnel, such as developers, to handle the additional tasks introduced by non-technical practices. P7 highlighted this issue, stating:

> *"The huge problem was that if all developers are really occupied, no one has time to check code. Their merge requests just stay in this queue forever, then they get outdated, and you have to do merge reviews, rebase, or merge new changes on top of your current branch it's just a huge mess." (Interview 7)*

Implementing additional practices often requires dedicated staff to take responsibility for these tasks. This includes ensuring team members have both the skills and the time necessary to effectively integrate and maintain non-technical practices. P0 emphasised this gap, stating:

> *"Developers don't usually focus on improving the development process, even though it would be beneficial. Many developers, unsurprisingly, don't care much about regulations." (Interview 0)*

Without adequate staffing or engagement, these practices risk either negatively impacting the development process or being discarded altogether. Over time, they may also lose credibility among developers, resulting in neglect and ultimately failing to add value to the organisation.

Key finding: Limited staff and availability hinder the implementation of non-technical practices. Developers are often busy preforming technical tasks, causing delays and neglect of non-technical practices, which can lose value if not maintained, and ultimately fail to benefit the organisation.

4.6.4 Organisational factors

Justifying and implementing additional non-technical practices across multiple development frameworks is a significant organisational challenge, one that becomes increasingly complex with larger companies, development teams, and diverse products. P8 highlighted the administrative hurdles involved, stating:

> *"Almost all of them know the problems, almost all of them know the solution. The only thing that's lacking is they cannot actually put those solutions in place to solve the problems because there is a lot of overhead and a lot of admin stuff that needs to go through before they can even implement the solution."* (Interview 8)

Leadership must have a thorough understanding of their development processes and lifecycle to successfully integrate the right non-technical practices. These practices require in-depth knowledge of the organisation's processes, products, and the specific non-technical methods being adopted. P0 emphasised this disconnect, noting:

> *"It's just again, a crossroads between people who are technical, and they understand the problems, and then the people who make the rules and enforce the rules that have to, you know, deal with the consequences."* (Interview 0)

However, striking the right balance between invested resources, process improvement, and project requirements remains critical. For organisations with varied or evolving products, this balance can be particularly challenging. P7 reflected on the difficulties, stating:

> *"I still think most of the time it's just too much, like we have extra bureaucracy on top of our development efforts, and it doesn't work."* (Interview 7)

The key to effective adoption lies in aligning organisational priorities and resources while minimising unnecessary bureaucracy, ensuring that non-technical practices enhance rather than hinder development efforts. Leadership has to be involved in the incorporation of non-technical practices and understand their value and requirements for them to grant the necessary resources and permissions.

Key findings: Implementing non-technical practices is challenging due to administrative hurdles, lack of leadership understanding, and conflicting priorities. Balancing resources and processes while minimising bureaucracy is key to successful adoption, which requires leadership support to be accomplished.

4.6.5 Complexity of integration and application

The additional complexity in the adaptation process of and the added complexity that are introduced by non-technical practices to the development process were frequently criticised by participants.
Due to the complexity of implementing non-technical practices properly and in the right place, it requires significant effort from developers and leadership. Thereby diverting their attention from tasks that directly contribute value to the company. P9 elaborated on this challenge, saying:

> *"Yes, this is difficult because it may not be implemented regularly or correctly. In addition, it needs to be adapted for each department or development team so that it truly fits them and can be implemented effectively."* (Interview 9)

The complexity of adding non-technical practices to a development process is seconded by P10 stating:

> *"The implementation of non-technical practices costs money and nerves." (Interview 10)*

This highlights the added difficulty of not just adopting non-technical practices but doing so in a way that ensures they are effectively integrated into the development process, which demands additional resources and careful attention.
Following their introduction, the addition of these practices were also seen as complicating output producing tasks with the additional steps. Adding these steps requires more time to complete these processes, compounding the existing time constraints many developers mentioned they were affected by. Thereby trough the increased number of steps to properly integrate non-technical practices into development process and differences in required skills for these it becomes more complex for developers to complete their work. As P9 noted:

> *"The problem often is that adding new elements increases complexity. This frequently leads to these practices being used less in the industry because there is either not enough time or money for them."* (Interview 9)

Practitioners agreed that if the time or money is not present to complete these more complex processes, organisations do not implement them and practitioners do not use them as they add no or diminished value. Even though their long-term benefits, such as enhanced security, were acknowledged P3 expressed frustration with the complexity non-technical practices added to the development process, saying:

> *"It sucks, because a lot of the time it's really just like, 'hey, it makes a lot of things more difficult from developing code to testing to deploying it in production.' These security features make it more difficult for you to do those things, but it saves you in the long run from having to deal with security incidents."*(Interview 3)

Another concern raised was the restrictive nature of certain practices, such as strict security protocols, which can limit developers' ability to access tools or customise their setups, ultimately reducing productivity. P7 illustrated this point with an example:

> *"Right. Because like for example, we still have some people like that, especially like the trainees. And for example, they get these Windows laptops with like 1000 anti-malware programs, and they can't install apps themselves and stuff like that. And it's just bad because you can't be productive. Like if you can't get the tools or the setup you need yourself because you're used to it, you're just going to be very unproductive. So in that sense, they just stop it."* (Interview 7)

These responses highlight that while the long-term advantages of non-technical practices are recognised, their immediate impact on the complexity of day-to-day development tasks often leads to frustration among practitioners. This perceived trade-off between complexity and security underscores the challenge of balancing short-term efficiency with long-term stability, leading to hesitations in organisations concerning the implementation of non-technical practices.

Key findings: Non-technical practices complicate processes through additional steps, requiring more varied expertise and consuming extra time and resources to complete them. While they offer long-term security benefits, developers find them frustrating due to additional restrictions and required knowledge to effectively implement the non-technical practices. This causes non-technical practices to not be adopted or dropped due to improper application due to added complexity, if the pre-existing application knowledge is not present.

4.6.6 Lower Productivity

A key concern raised by participants about non-technical practices was their perceived impact on productivity. Several participants questioned whether the benefits of these practices, such as enhanced reliability, justified the extended development timelines they often entail. P7 voiced this concern, stating:

> *"I wouldn't even say that they really improve the development process because on the one side you have these side sort of 'hey, it's going to be more reliable,' but what's the importance of reliability and like focusing on these rules set, if it means that the development takes half a year longer? Long term, right, you always have to keep that in mind." (Interview 7)*

This sentiment was echoed by P2, who reflected on the pressures of meeting deadlines. They observed that non-technical practices are frequently the first to be de-prioritised when time constraints become critical:

> *"A major challenge is that during stressful projects, there is a tendency to prioritise technical practices due to time pressure. In such situations, people often think short-term and neglect the long-term important non-technical practices. When stress is high, you fall back on what can be implemented quickly and is feasible, with the mindset: 'It doesn't matter, we just need to get this done now,' focusing on time and costs." (Interview 2)*

These perspectives highlight a recurring tension in software development, while non-technical practices may offer security and long-term advantages, they are often viewed as expendable under the immediate pressures of deadlines and resource constraints. These are seen as additional elements that can be added post-delivery and are there for cut out of the development process to be added later, defeating the purpose of continuous security in development.

Key findings: Participants expressed concerns that non-technical practices could reduce productivity by extending development timelines through the addition of steps in the development. While these practices improve consistency and reliability, due to this they are often de-prioritised under time pressure, as teams focus on short-term goals over long-term benefits such as quality and security.

4.6.7 Non-compliance

A key challenge identified in the interviews was the inconsistent adoption and application of non-technical practices within software development processes by developers. While these practices are designed to enhance development outcomes by addressing aspects like documentation, training, and compliance, their success depends heavily on meaningful engagement from practitioners. However, many developers expressed frustration with practices they perceived as tedious or irrelevant, leading to non-compliance. As P7 explained:

> *"You should always check: 'OK, how useful is it?' Because if you force people to do it every month, they will just click through it and get bored. And it just doesn't work." (Interview 7)*

Security training emerged as a particularly contentious example of ineffective non-technical practices. Many participants described these sessions as poorly designed and disengaging. P8 highlighted the widespread lack of motivation:

> *"The thing is with training, what I do is I just skip through all the videos. I cannot be bothered for the life of me, and I think a lot of people do the same thing. [. . .] The reason it's not useful is the way the trainings are done is really, really bad." (Interview 8)*

This sentiment was echoed by P9, who remarked on the common tendency for employees to complete such trainings without meaningful participation:

> *"For example, with training sessions we have to attend, most people just click through them. I can completely understand that." (Interview 9)*

The perception of these practices as overly rigid or impractical often led developers to bypass them entirely. P7 described how cumbersome security procedures sometimes result in shortcuts:

> *"It's very common in companies where security procedures are just so hard and such a pain for the developers that they will just take shortcuts." (Interview 7)*

Participants also pointed to the broader consequences of non-compliance, citing examples where neglecting non-technical practices contributed to critical failures. P7 provided a hypothetical example to illustrate this risk:

> *"For something like Crowdstrike, maybe one developer just pushed it to production like, 'Hey, I don't want to go through this whole compliance stuff.' They knew, 'It works, everyone's happy.' But this time, it didn't really work that well." (Interview 7)*

These challenges suggest that developers often struggle to see the direct benefits of non-technical practices, particularly when they are perceived as time-consuming or disconnected from immediate development goals. As P0 observed:

> *"Developers don't usually focus on improving the development process, even though it would be beneficial. Many developers, unsurprisingly, don't care much about regulations." (Interview 0)*

Participants emphasised that non-technical practices must be fully implemented and monitored to deliver value. P1 noted the importance of comprehensive adherence to these practices:

> *"Non-technical practices have to be fully featured. You need a review at the end to ensure that you've followed the secure practices laid out. [. . .] If no one reviews for security gaps in the software process, it falls short." (Interview 1)*

The findings highlight that organisations must address non-compliance by redesigning non-technical practices to be more practical, engaging, and clearly aligned with development goals. By doing so, they can mitigate the risks associated with non-compliance and ensure that these

practices are fully integrated into workflows, with developers actively and willingly engaging with them.

Key Findings: Non-technical practices often suffer from inconsistent adoption due to lack of engagement and perceived irrelevance or complexity. Developers frequently bypass security training and compliance tasks, seeing them as tedious or impractical with no little benefits to the development. Poorly designed and integrated non-technical practices encourage shortcuts, increasing security risks, when practices are skipped or not completed. To improve adherence, organisations must make these practices more practical, engaging, and aligned with development goals for developers to engage with them consistently and improve developed.

4.6.8 Lack of engagement

Lack of engagement was mentioned as a problem when implementing and applying non-technical practices, especial when their contributions to development were not obvious.
To solve this, gamification was identified by multiple participants as an effective way to engage practitioners in non-technical practices. By integrating elements of reward and recognition, gamification can make these non-technical practices feel more rewarding and less disconnected from core development efforts. P2 emphasised:

> *"Everyone still has a little child within us. We work based on reward systems, dopamine, serotonin. We need that 'well done' from others that's just how it is. Praise and recognition are an incredibly important practice." (Interview 2)*

By motivating individuals through rewards and a sense of achievement, gamification can make the benefits of non-technical practices more tangible, encouraging participation and enthusiasm. P5 supported this, stating:

> *"Yeah, it's like, don't just flip through 50 pages and then have a small quiz or something where you just don't care about it. I want it to be fun and presented in a way where I'm not going to start yawning and having my eyes roll back in my head. I think it's worth investing in good security training a bit." (Interview 5)*

Participants also suggested that gamification could foster engagement without creating unnecessary competition. P2 proposed:

> *"It would make sense to establish a ranking system within the company that doesn't create competitive pressure among employees, as is often the case with scoring systems. Instead, internal goals could be set, similar to agile development and Scrum, so that everyone can track their progress without having to compare themselves to others. For instance, if you complete 100 of 800 work packages by the end of the year, you could receive additional compensation. That would be a form of gamification that motivates individual employees to engage more deeply with their toolbox."(Interview 2)*

In addition to fostering engagement, gamification can boost productivity and profitability by encouraging employees to perform necessary but monotonous tasks more enthusiastically. P2 summarised this idea, stating:

> *"When we were just talking about fun and gamification, it helps ensure that everything doesn't feel so serious and formal. Ultimately, overall improvement is just a means to generate more revenue for the company." (Interview 2)*

In conclusion, gamification offers a promising approach to increasing engagement with non-technical practices by making them more interactive and rewarding. By incorporating elements of recognition and progress tracking, it can enhance motivation without creating unnecessary competition. This approach not only improves participation but also boosts productivity and overall company success by making sure developers engage with less tangible aims such as security and quality.

Key Findings:
Gamification enhances engagement by making non-technical practices more interactive and rewarding for developers. It increases motivation, reduces resistance, and improves productivity while fostering a positive culture and work environment. This leads to more secure products through personal investment of developers into projects.

4.6.9 Overload of non-technical practices

The adoption of non-technical practices and their full implementation is a challenge that has many factors that should be regarded throughout different organisational types and stages. Adding non-technical practices for the sake of including them can not only waste resources but be detrimental to compliance and development efforts.
Finding the balance for the correct, right amount and placement to reach a balance of the invested resources and the benefits they provide is the task of every development team and their management. As P6 stated:

> *"In some projects, I would like to make improvements, but I think my team has found a good balance. We have a well-balanced ratio between the requirements and the practices we've implemented, so I don't currently see a need for specific changes."* (Interview 6)

Though many interview participants proclaimed to want to integrate more non-technical practices, finding a balance is important to prevent bloated over regular processes that hinder the development process more than they improve it. Due to this, on some occasions it might be better for organisations to place their valuable resources where they are needed most, instead of adding unnecessary non-technical practices with no added value for the sake of having them. As P8 highlights:

> *"You might as well save up those resources and use them somewhere else. Then, when you realise, OK, yeah, now we actually need to take this to the next level, that's when you bring those resources back in."* (Interview 8)

Finding the right balance in implementing non-technical practices is essential to maximising their benefits without overwhelming development processes. While most developers see value in these practices, others, at perceived higher maturity stages, found that excessive implementation can be counterproductive. They found that organisations should prioritise their resources effectively, introducing non-technical practices where they add real value rather than

implementing them for their own sake. Fining the balance between productivity, safety and bureaucracy is something that teams have to learn from their own experience, especial when they rely on custom frameworks that don't highlight practice synergies.

Key findings: Finding the right balance in implementing non-technical practices is essential to maximise their benefits without overwhelming development processes. comprehensive understanding of practices and experience are key to ensuring that non-technical practices support, rather than hinder, productivity and security through recess implementation.

4.7 RQ5: "Why do organisations perceived as having a higher degree of maturity by developers integrate more non-technical practices in their secure software development process?"

4.7.1 Availability of Resources

The availability of resources plays a significant role in the implementation of non-technical practices. Larger organisations with higher maturity levels often have invested more financial, human, and time resources to support these practices, enabled by their higher market share or profits. For instance, P1 emphasised the importance of resource availability by stating:

> *"I'd recommend them (non-technical practices) for most organisations, but the key factor is having the resources."* (Interview 1)

A recurring concern mentioned by several participants is the lack of money and time as key obstacles to implementing non-technical practices, particularly for smaller projects or startups. P9 highlighted this challenge and the compounding long term costs of implementing non-technical practices, noting:

> *"I believe that there is often a lack of money and time. In many projects, it may not necessarily be required, but in larger projects, the question arises of who will cover the costs for continuous maintenance and updates. This can represent a significant obstacle."* (Interview 9)

On the other hand, established companies typically have the means to allocate more resources toward non-technical practices such as training, documentation, and process improvements. P1 acknowledged the disparity between mature organisations and startups, remarking:

> *"It's a completely different mindset. We have the resources, which makes my job easier, but unfortunately, startups don't."* (Interview 1)

Organisations with higher maturity levels tend to have invested in more diverse sets of practices, which are then often already standardised within their processes based on experience. As P4 explained:

> *"But now, I believe we have more resources to move further in this direction, professionalise ourselves, potentially hire people, build teams, and implement more practices."* (Interview 4)

The insights demonstrate that while resource availability is essential, the gap between mature organisations and startups creates significant disparities in their ability to implement and sustain non-technical practices effectively. Participants from less mature organisations were willing to incorporate additional non-technical practices, but were more selective in their investment of resources. Where as large organisation were willing to invest as they saw a higher reciprocal in terms of benefits for their invested security or longevity.

Key Findings: Larger, more mature organisations have more financial, human, and time resources to support non-technical practices, making implementation easier. Smaller organisations, particularly startups, face challenges due to limited resources, often struggling to cover ongoing maintenance and expansion, selecting to prioritise more technical practices.

4.7.2 Focus on Long-Term Goals Over Immediate Gains

Interview participants highlighted the contrast between mature organisations and startups in their approach to long-term goals versus immediate gains. participants explained that in their experience, mature organisations often prioritise sustainability and long-term security over short-term speed or market capture. In contrast, startups may sacrifice quality and security to move quickly and increase output speed. As P4 explained:

> *"I think this is always a challenge, especially as a startup. But over time, it might become easier, particularly if you haven't received significant external funding for a long period. In that case, you really have to focus on the essentials. For a startup, it's primarily about achieving product-market fit and acquiring customers. You always have to carefully consider how to make the best use of your time."* (Interview 4)

Participants from larger and more established organisations described a focus on formalised non-technical practices that help minimise risks and maintain product quality over time. P1 emphasised this in comparison to less mature organisations, stating:

> *"That's the big difference between a large corporation like Company B, which has the resources and a long-term view, and a startup, which is more focused on capturing market share or being first to market. Startups often don't have the time and rely on a 'better to ask forgiveness later' approach."* (Interview 1)

Further commenting on the difference in practices separating the maturity types and the effect on cost and security P1 elaborated:

> "Small companies often prioritise immediate sales over thorough processes, thinking they can address issues later. They tend to forgo essential practices for quicker gains, which can lead to higher expenses down the line. While I understand the need to gain market share, from a security standpoint, it's much more effective to establish strong practices from the beginning." (Interview 1)

This focus on immediate gains was often observed in the way startups handle development and scaling, this was echoed also by other participants. P5 recalled a period of rapid growth and its impact, noting:

> *"There was a time where the company grew really, really quickly, there was a lot of code and development that was going straight through, and so the code is essentially at a point where it needs to be maintained and mature."* (Interview 5)

Younger organisations frequently mentioned short-term growth as their primary goal, with plans to formalise processes and adopt non-technical practices once they reach a sufficient size or level of maturity. This highlights the differing priorities between organisations depending on their stage of development, with startups balancing immediate needs against aspirations for long-term stability.

Key Findings: Startups often focus on rapid growth and market capture, sacrificing quality and security, while more mature organisations prioritise sustainability, security, and formalised non-technical practices. This difference in focus can result in startups forgoing essential practices in favour of speed, which may lead to higher long-term costs and security issues.

4.7.3 Greater Organisational Structure and Formalisation

Higher maturity often correlates with more formalised processes, including documentation, training, and regular evaluation. As organisations scale, they must adopt structured practices to maintain consistency and reliability, especially with increased staff size and turnover. P4 highlighted this need, explaining:

> *"We're starting to integrate more professionalised processes because our size demands it."* (Interview 4)

Mature companies implement frameworks such as sprint reviews, retrospectives, and standardised security measures. These frameworks address the challenges of sharing risks and procedures in larger, multi-team organisations, where informal communication methods used in smaller startups are no longer effective. As P4 described:

> *"That works well up to a certain point, but the more you scale, the more difficult it becomes. We now have four teams, and it's no longer as easy because you lose comparability at the organisational level."* (Interview 4)

Participants from larger, older organisations often recounted a broader range of non-technical practices that had been adapted based on experience and formalised over time. This evolution reflects organisational maturity, where previously assumed or ad-hoc approaches are replaced by structured frameworks. As P4 noted:

> *"At a certain point, metrics and mature processes become necessary to manage and streamline workflows."* (Interview 4)

Non-technical practices become increasingly critical as organisations grow because they manage human factors, ensuring coordination among teams, measuring performance, and maintaining consistency. For example, metrics development and process evaluations play a pivotal role in addressing the complexity and fragmentation that arise with scale.
In contrast, P1 commented on the disadvantages of larger, more mature companies citing the fragmentation and its prevention of further learning:

"What limits our organisation from learning more is largely due to our size and fragmentation" (Interview 1)

This is echoed by P8 commenting on the hindrance of bureaucracy that occurs at certain organisational stages:

"This has been an issue in my current company and in past roles, where administrative tasks often bog down development. Processes that could realistically be completed in a day can end up taking a full business week due to unnecessary delays. Regularly refining these processes could streamline workflows significantly, freeing up valuable time and resources." (Interview 8)

This further shows the downsides of over formalised structures, often overburdened with redundant practices that compound bureaucratic effort hindering agile development. As shown above, participants comment on the constant need for revaluation of practices to stay efficient and prevent waste.

Key Findings: As organisations mature, they implement more formalised processes to ensure consistency and coordination, but this can lead to fragmentation and bureaucratic delays that hinder agility. Regular revaluation of these practices is necessary to maintain efficiency, balance growth with flexibility and adapt to changes in the security landscape.

4.7.4 Experience and Awareness of Risks

Participants from more mature organisations often reported of past challenges or security incidents that motivated the implementation of preventive non-technical practices, such as security training, root cause analyses (RCA), and corrections of errors (COE) procedures. P3 described one such approach, noting:

"Company B has a process called a COE, or a correction of errors. It's sort of like an RCA, where you do the root cause analysis, the five whys, and everything else. But then you also build an action plan for fixing it in the future."(Interview 3)

This experience accumulated over time enabled these organisations to recognise the importance of proactive measures and the need for certain non-technical practices. P8 described a process of awareness risk causing changes in a mature organisation:

At Company A, this (problem of bottlenecks) has been recognised, and it's actually quite mature in this regard. The heads of teams understand that there is a heavy reliance on certain individuals. It becomes natural for others to seek answers from these people. However, they are now working to transition into a more process-dependent organisation, reducing the need to rely on individuals. (Interview 8)

By adapting their development approaches through awareness and experience to incorporate these practices, they positioned themselves to respond more effectively to potential incidents and continuously improve over time. P6 described the process of learning from experience and it becoming more mature explaining:

> *"It (our development process) became increasingly effective. Clearly, of course, you try things out, so it might happen that you say, "Hey, let's try it this way or that," and then you realise that everything is disrupted and doesn't quite fit. In that case, you can, of course, adjust it again. But yes, it actually kept improving because everyone became more in sync."* (Interview 6)

This shows the adaptation and improvement that happens over time in organisations, as they mature and learn from their experiences and implement non-technical practices based on them. Currently it seems as though organisations learn more from their own experiences than from others or academic knowledge, needing longer to do so but creating a more personalised process through experimentation.

Key Findings: Organisations highlighted how past challenges or security incidents drove the implementation of preventive non-technical practices creating more mature development processes, resilient to vulnerabilities. These experiences helped organisations recognise the importance of certain non-technical practices, leading to a focus on continuous improvement in their development processes.

4.7.5 Regulatory and External Pressures

As organisations evolve and grow, they often face pressure to conform to stricter government regulations or follow industry standards, in order to compete, that require formalised security practices. This includes non-technical measures, such as compliance documentation and training programs, which become essential to meet these expectations from stakeholders and regulators. P4 highlighted this need, stating:

> *"We've now reached a size where more and more documentation is being externally required of us. To be honest, we haven't done much in this regard so far, because up until now, it has essentially been handled by individual people. I'm the primary point of contact when something goes wrong, and I have a plan in my head for how I would need to act. However, this plan is not documented in writing." (Interview 4)*

This sentiment was echoed by P10, expanding though that these pressures diminished their productivity:

> "I suspect that the bigger we grow, the more likely we will need something like this, especially if customers demand it from us. At the moment, we still have peace regarding such matters and can focus on what truly delivers value." (Interview 10)

This shows the evolution of organisations into more mature versions and the pressures that force them to adopt certain non-technical practices to comply with external stakeholder pressures. The specific requirements for compliance and the needed non-technical practices vary depending on the project or customer. P7 explained:

> *"We follow some TÜV standard [58] or if it's like a government project, we have to follow government guidelines on security. But it really depends on the project."* (Interview 7)

The importance of compliance grows with an organisation's size and responsibility, necessitating the installation of non-technical practices to ensure adherence to regulations and avoid repercussions. P0 emphasised this shift, noting:

> *"I would include regulations and standards as important, like NIS 2 [13] being one example, and DORA [14] as another. Nowadays, CTOs or directors get personally fined for not obeying those laws, which is an interesting shift."* (Interview 0)

This demonstrates how external pressures not only drive the adoption of non-technical practices, but also highlight the increasing accountability required from leadership in larger organisations, requiring them to become more mature in their security.

Key Findings: As organisations mature, they face greater external pressures to adopt formal security practices to comply with regulations and industry standards due to the projects they develop. Larger, more mature organisations must prioritise compliance, as they becomes increasingly accountable for meeting these requirements.

4.8 RQ6: "What methods are used by organisations to estimate the effectiveness of non-technical practices in the context of developing secure software?"

4.8.1 Effectiveness of Software Development Life Cycles (SDLCs)

In the interviews participants generally reported that the effectiveness of their general software development approach and its security was not explicitly measured through any kind of fixed or regular processes. Instead, evaluations often relied on informal assessments or developer feedback prompting specific changes in methodology. As P1 explained that quantitative evaluation of the development process was uncommon:

> *"Basically, when discussing the effectiveness of development, we aren't really measuring it quantitatively." (Interview 1)*

The only instance of a structured evaluation came from P10, who described how their organisation measured the impact of introducing a mock back-end process. By reviewing past projects and assessing the distribution of effort across different components, they demonstrated a clear reduction in development effort:

> *"The evidence is even measurable [. . .] Change efforts have been halved with the new approach (of providing customers with a mock back-end). A review of past projects shows back-end and frontend efforts are generally balanced at 50/50." (Interview 10)*

This case stood out as the only example of a comprehensive evaluation of a devlopmen approach effectives, likely because it followed a recent and significant methodological change. For most participants, the evaluation of development effectiveness in security and otherwise relied on subjective assessments, such as personal experience or intuition. P4 highlighted the reliance on "gut feelings" and leadership experience, explaining:

> *"I found it extremely difficult to find reliable metrics. As long as I, as CTO, don't feel that something is going in the wrong direction, I think everything is quite good." (Interview 4)*

Customer satisfaction was also cited by P4 as a key indicator of an effective methodology. Delivering projects efficiently and meeting customer expectations in quality, cost and security were viewed as proof that the approach was working:

> *"I would say the proof of this is that we continue to deliver, and that at an acceptable speed, and our customers are satisfied." (Interview 4)*

In summary, while formal evaluations of development effectiveness were rare, participants identified customer satisfaction and qualitative assessments in the form of feedback as important indicators of success. The single quantifiable example provided by P10 demonstrates the potential for structured evaluation methods to provide clear evidence of improvement though, particularly in cases of significant methodological changes.

Key Findings: Most participants reported that the effectiveness of their SDLCs was not formally measured, relying instead on subjective assessments such as personal experience or customer satisfaction. While formal evaluation methods were rare, customer feedback and the continued delivery of projects on time were commonly viewed as indicators of success.

4.8.2 Quantitative methods of evaluating the effectiveness of non-technical Practices

Quantitative metrics or numeric proof of effectiveness of implemented non-technical practices related to managing human, organisational factors and security were rare in the interviews. Most quantitative metrics mentioned by participants to evaluate the success of their implemented practices were based on standard engineering metrics of their code base and focused on general software engineering. These key performance indicators (KPI's) collected in the development process were compared before and after the implementation of non-technical practices, to gauge the effectiveness of said practices. Most of these metrics were related to development performance, such as ticket throughput, story point burn down charts, ticket response times, and system health metrics like system uptime frequency of tickets returned from testing, with some focusing on security such as the number of vulnerable dependencies in the code created. These KPIs serve as proxy measures for evaluating the success of general development approaches and the impact of non-technical practices on security and otherwise. Providing an example of how managers use metrics to encourage improvements in team performance and adherence to security guidelines through team comparisons, P6 stated:

> *"You have the software development teams, and they have a manager, and that manager has a boss who looks at a table, and it shows that one team has 10 artifacts, and they all have only 10% test coverage. On the other hand, the other team's artifacts have 80% test coverage. Then, the manager asks the first team and its leader, 'Hey, why do you only have 10% test coverage?' And then it happens automatically you'll feel compelled to improve that as a result." (Interview 6)*

Most of the quantitative metrics were collected using a ticket system such as in JIRA, which was used by all developers interviewed, or other related tools. These tools were stated by

participants to provide a structured way to gather and analyse data on the effectiveness of development, security and general. P10 described the active tracking of ticket return rates as a way to monitor the effectiveness of the development process:

> *"One thing I didn't mention earlier is the rate at which a ticket comes back. When we deliver a ticket, it goes to the testers, and we receive feedback on it. I actively track this as well, specifically per employee. It helps me get a sense of how the process is running." (Interview 10)*

Referring to aspects of general software development, this was also the case in P6's organisation who said:

> *"How can you measure whether you're improving? Within a Scrum process, you might estimate tickets in Story Points and then observe whether your velocity increases or decreases. I think that's the main metric for assessing how efficient your team is."* (Interview 6)

This shows a strong usage of technical and automated tools to manage the contentious measurement of effectiveness in organisations. All participants explained they used JIRA as a tool to manage their development, though there was a disparity in the usage of the tool to quantify effectiveness of development and certainly individual practices. Security metrics were, for example, not frequently collected in these organisations, with half of the participants mentioning though that this might be a future improvement they could consider implementing. P1 explained this lack of measurement in fields not directly related to productivity or quality explaining:

> "We don't have to do too many code rewrites for security. That's probably the only observable metric we have, but we don't actually record or log that metric anywhere." (Interview 1)

A different participant highlighted the use of tickets also extended to creating metrics on tracking vulnerability trends and assessing their severity:

> *"This is, of course, both the quantity and quality of errors. There should simply be fewer errors over time, and they should also be less significant that is, less severe errors. This can easily be measured using the ticketing system." (Interview 2)*

For those who did collect such metrics, ticket systems served as a valuable tool for tracking and analysing trends. As one participant explained:

> *"You can assign a security label to a ticket, and by the end, your labels will provide data. Over the course of the year, you'll get a curve that, ideally, trends downward. That way, you know the system is becoming more secure." (Interview 2)*

These KPIs served as proxy measures for evaluating the success of development approaches and gauging the impact of non-technical practices, even though they were not specifically designed for this purpose. By adapting standard engineering metrics, participants mentioned that their organisations found a pragmatic way to approximate the effectiveness of their non-technical

initiatives. This benchmarking showed improvements or declines in product and development security, quality and otherwise.

Key Findings: Quantitative metrics for evaluating non-technical practices were rare, with participants using standard engineering metrics such as ticket throughput and test coverage as proxies. Management tools like JIRA helped track team performance trough tickets, though security-specific metrics were less common, with some participants considering them as future improvements. These metrics provided insights into development effectiveness, even if not directly tied to non-technical practices, affecting their reliability.

Quantitative assessment of security training The non-technical practice most frequently evaluated using quantitative metrics was security education, which was regularly assessed by ten out of eleven participants. The success and effectiveness of security education was evaluated through multiple methods in organisations for which quantitative metrics were collected according to participants, including:

Retention Assessment: Participants were required to achieve a certain percentage of correct answers at the end of training sessions on the presented material, which was typically presented in the form of slides or videos. P2 explained the process, its meaning and outcomes:

> *"A monthly security quiz that is repeated to refresh knowledge. This quiz is the tool, and the practice behind it is the repetition and verification of security knowledge. This way, I ensure that employees still have the important security guidelines in mind. If someone does not pass the quiz, they would first have to attend training again before being allowed to program." (Interview 2)*

Similarly P0 commented on the process, stated the variance in these assessments, but voiced his discontent with this methods effectiveness:

> *"So far, I've completed a few, and the quality varies. Some are really good, while others are quite poor. Sometimes, it's just a video followed by the simplest multiple-choice quiz imaginable. Other times, they're more interactive, with a video that asks questions like, 'What would be the right course of action in this situation?' or 'Can you identify five different risks?' However, even those are all multiple-choice and fully automated, so their actual effectiveness is debatable. Still, it's definitely better than nothing." (Interview 0)*

Phishing simulations: Internal security teams sent phishing emails to participants, tracking the percentage of successful reports. Failure to report a phishing attempt resulted in additional training on the topic. P0 stated his experience:

> *"And they would send out a Phishing email from the company, and if you report them through outlook, for example, then you would get an email saying, "Great job, you reported a phishing email" and if you if you mess up then you also get an email saying like "hey, you probably shouldn't click that link blah blah"." (Interview 0)*

Real-world incident tracking: The number of successfully prevented and not prevented attacks on systems. This gave evidence of the level of security knowledge and could be compared in a before and after model when security education was implemented. P5 shared his experience with the gathering of quantitive results:

> *"So yeah, one thing we measure is incident reports, like phishing attacks or data breach attempts. We report these to our security team, and they gather statistics about how often people are encountering these threats and reacting appropriately, like reporting it or avoiding clicking on it. For example, we can track how many people clicked through a phishing attempt or if their laptops were infected with malware, which would then trigger a response process to handle it. These types of metrics help quantify the threat levels." (Interview 5)*

Capture the Flag: One Participant stated that his organisation tested the effectiveness of security training on physical access management to workspaces by placing on object, such as a flag, in the workspace and having an unauthorised person sent to retrieve it in order to see if they would be prevented from doing so. P0 explained this practice:

> *"And then where you have like, you have also have physical security, you know combined with technical for example using a pass to get into certain parts of the building, they would have people doing tests where they would put a flag in the middle of a of a room and they will be like our hey, somewhere in this month some someone's gonna come over and try and steal the flag without it, without a without being allowed into the building actually." (Interview 0)*

In conclusion, the evaluation of non-technical practices, especially in security, through quantitative methods in software development is still limited, but present in certain areas. Most metrics used to assess the success of these practices are indirect, relying on standard engineering KPIs such as ticket throughput, story point burn-down, and system health metrics. These proxies offer valuable insights into the broader impact of non-technical practices, though their primary focus is on development performance rather than evaluating the practices themselves. Security-related practices, such as security education, were most frequently assessed using quantitative methods. These included retention assessments, phishing simulations, real-world incident tracking, and even physical access tests. Despite the variety of approaches, challenges remain in achieving consistency and ensuring the effectiveness of these metrics, with some participants expressing concerns over the depth of the evaluation tools used. Overall, while quantitative metrics are being applied, especially in security, further development and standardisation are needed to better assess the true impact of non-technical practices within the software development process.

Key Findings: Quantitative metrics for evaluating non-technical practices in software development were rare, with most organisations relying on engineering KPIs like ticket throughput as indirect measures. Security training was most commonly assessed quantitatively through methods such as quizzes, phishing simulations, and tracking real-world incidents. While these approaches provided useful insights, challenges remained in ensuring their consistency and effectiveness. Overall, though quantitative methods are being applied, further development is needed to accurately assess the impact of non-technical practices.

4.8.3 Qualitative methods of evaluating the effectiveness of Non-Technical Practices

Qualitative evaluation on the effectiveness of non-technical practices was the most common approach and was mentioned to be conducted in one way or another by ten out of eleven participants.

Case studies Though there may be discrepancies due to varying external circumstances, case studies emerged as the most effective way of comparing development approaches, including methodologies and the adaptation and implementation of non-technical practices. Among the participants, six mentioned conducting case studies as a means to evaluate and refine these practices.
Participants from larger, more mature organisations frequently emphasised the use of comparative case studies. These involved comparing two teams with similar circumstances but different development methodologies or practices. P2 illustrated this idea, saying:

> *"You simply have to compare projects and apply these non-technical aspects anew in each project. Conduct field studies within the company or have different project teams work in parallel. These teams are each given a methodology and rotations. In the end, you have various projects with different people and approaches. You can then effectively present and analyse this in a matrix." (Interview 2)*

However, participants noted that comparative case studies require significant resources, including comparable teams, sufficient time, and organisational support, which are more commonly available in larger companies. In smaller organisations, single case studies were more commonly employed. These focused on analysing a team's performance before and after introducing new non-technical practices.
P2 emphasised the importance of allowing sufficient time for case studies to yield meaningful results:

> *"To build a matrix and simply record the trial periods within the company, you really need to stick with it for a year; you can't test a practice for just two weeks and then say it didn't work. That doesn't work." (Interview 2)*

Similarly, P5 highlighted how teams could derive continuous improvements from lessons learned in previous projects:

> *"Yeah, it's more of a continuous process. Projects where something went very well, they look at how can we integrate that into other projects. Or projects where there was a security incident, and then based on that, there's a new rule for everyone." (Interview 5)*

Overall, while comparative case studies offered valuable insights for organisations with the resources to implement them, single case studies provided a practical alternative for smaller companies. Both approaches underscore the need for sustained effort and reflection to effectively evaluate and refine non-technical practices in software development.

Key Findings:

Case studies were used to refine non-technical practices, with larger organisations favouring comparative studies, while smaller companies relied on single case studies. Both approaches highlighted the need for sustained effort and continuous improvement in the form of assessment of non-technical practices and their contribution to security.

Audits and external evaluations Participants reported that their organisations frequently relied on audits, often conducted by external entities, to assess the effectiveness of their development practices, including non-technical initiatives. These audits provided structured evaluations that highlighted gaps in processes and suggested improvements, thus supporting the refinement of non-technical practices. P8 emphasised how audits require a detailed review of documentation and operational plans, which can expose weaknesses in organisational practices:

> *"There are audits that happen in every firm, like security audits, and the security auditor usually asks for different kinds of documents, such as your disaster recovery plan, mitigation plans, and vulnerability management plans. Sometimes companies don't have these plans ready."* (Interview 8)

Audits were also viewed as a practical tool to validate and enhance the implementation of non-technical practices. P0 elaborated on how external reviews offered specific recommendations that contributed to organisational improvement:

> *"But when it comes to what we're doing for ourselves, there are enforced audits conducted by external organisations. These audits generate reports highlighting the gaps and offering recommendations for the cyber team. For instance, suggestions like, 'Hey, you should be improving your contingency plans' or 'You should be conducting security tests more frequently.'"* (Interview 0)

Start-up participants noted that external audits were particularly useful in ensuring their non-technical practices were effective and aligned with industry expectations. Investors often required these audits to assess processes and provide feedback on areas where non-technical practices, such as team coordination or process transparency, could be improved. P4 described their experience with investor-driven audits:

> *"As part of the funding round, the investor obviously also looks at this and evaluates our company. They go really in-depth and examine our processes. They've also found some areas for improvement, but overall, it's going well. I would say, for our stage, it's already very good."* (Interview 4)

Participants highlighted the positive impact of external audits on improving the quality of non-technical practices, particularly in areas like security and privacy management. Regular evaluations were seen as a critical component of maintaining and refining these practices. P7 underscored the importance of frequent external reviews in ensuring non-technical practices remained effective over time:

> *"Specifically for security and privacy practices, I'd say again, these sorts of external tests are really important. Maybe once a year, once every six months."* (Interview 7)

Through their structured approach, audits and external evaluations acted as catalysts for organisations to strengthen their non-technical practices. By providing a clear framework to assess compliance and identify areas for enhancement, these evaluations were instrumental in ensuring the effectiveness of practices aimed at improving human and organisational factors. Though expensive, they provided insights of compliance and refocused efforts based on neutral perspectives, which was appreciated by participants.

Key factors: Internal and external audits were frequently used in organisations to evaluate and improve non-technical practices, offering insights into gaps and areas for improvements. For start-ups, audits were essential for aligning with industry standards and satisfying investor requirements. Regular audits, particularly in areas like security and privacy, played a key role in maintaining the effectiveness of non-technical practices in these fields.

Feedback Loops Participants identified internal and customer feedback as key methods for evaluating the effectiveness of their development methodologies and the non-technical practices within them. Regular feedback loops, involving both customers and practitioners, were a staple in most organisations represented in the interviews. This feedback was valued as it provided insights from the end-users and team members, the primary stakeholders affected by the practices.
Given that most participants' organisations followed agile or Scrum methodologies, feedback loops were often already integrated into their workflows. These informal mechanisms were praised for their ability to capture contextual and dynamic evaluations of practices. P2 highlighted the value of feedback from team discussions and its role in assessing development approaches:

> *"You simply evaluate project results, conduct feedback rounds, discuss everything in the team dynamic, and then just observe how things develop over time." (Interview 2)*

Several participants emphasised that informal feedback was often more reliable than metrics, which could be misinterpreted or lack contextual depth. P4 supported this view, prioritising retrospectives and team discussions over quantitative indicators:

> *"So far, I would say that informal feedback is significantly more important. As part of Scrum, we also do retrospectives where all developers say what went well and what didn't go so well. This informal feedback from the team is, for me, actually a much more important indicator than any metrics, which are often hard to define." (Interview 4)*

Internal feedback from specialised teams, such as security teams, was also considered an effective means to refine processes and implement non-technical practices. Participants noted that these teams often conducted thorough reviews and provided actionable recommendations to improve organisational practices. P3 described how feedback from a security team helped evaluate the success of implemented practices and make further adjustments based on recommendations:

> *"As for evaluating success, we typically rely on Company B's security team for feedback and advice. They conduct a full review and, at the end, suggest improvements, advising us on what we should do differently." (Interview 3)*

Similarly, P1 highlighted the reliance on internal expertise to gauge the success of new initiatives:

> *"We use the security team and our experience to see if implementing something was successful. They conduct a full review and suggest improvements, advising us on what we should do differently." (Interview 1)*

Feedback loops were particularly valued for their adaptability and relevance, enabling organisations to continuously assess and refine their non-technical practices. By combining insights from internal teams, customers, and practitioners, these loops provided a dynamic and holistic evaluation framework, supplementing or even surpassing traditional metrics in their effectiveness.

Key Findings: Feedback from both internal teams and customers was crucial for evaluating non-technical practices. Many organisations, especially those using agile, integrated feedback loops, with informal team discussions and retrospectives valued more than quantitative metrics. Security teams also contributed by offering regular reviews and recommendations. These feedback loops provided a dynamic and continuous way to assess and improve non-technical practices and their effectiveness.

5 Discussion

5.1 Introduction

This chapter examines the role of non-technical security practices in software development, focusing on their implementation, challenges, and effectiveness across different organisations. While secure development frameworks have gained traction, human and organisational factors remain critical, yet often overlooked components of software security.
By analysing insights from industry professionals, this discussion explores how these non-technical practices are applied and adopted in real-world development settings, their perceived impact, and the barriers to their adoption. Understanding these factors is essential for refining industry standards, strengthening security strategies, and ensuring the development of more secure and resilient software.
Additionally, this discussion considers the broader implications of these findings, including their potential to shape policy recommendations and regulatory guidelines. Addressing the effectiveness of non-technical security measures can provide valuable insights for a wide range of stakeholders, from individual developers and smaller organisations to larger enterprises striving to enhance their security posture.

5.2 Key findings and results

SDLC

In our study, participants highlighted that organisations typically use informal development approaches, adapted to suit their specific needs, often prioritising flexibility but through these ad hoc approaches facing inconsistencies. Agile was universally recognised as the industry standard, with all participants consulted employing it, but customising the basic approach by integrating practices, including non-technical ones, to fit their development styles. DevOps, particularly CI/CD pipelines, were also commonly used for continuous projects to streamline development and deployment, when taking on additional operational responsibilities**.** A novel approach mentioned was the introduction of MVP's to the development process, these were seen as a valuable tool to help refine customer requirements early and reduce late-stage changes and significantly reduce development effort. However, participants pointed out issues, they were occasionally forced into using hybrid SDLCs, which merge Waterfall and Agile, especially when client-driven demands impose rigid structures, limiting flexibility and disrupting established workflows which lead to inefficiencies.

RQ1

In our study, participants distinguished technical practices as direct code-related activities directly affecting the product, while non-technical practices were seen as supporting structures for the management of human and organisational factors. Despite this distinction, both practice types were recognised as equally important and interdependent, with non-technical practices shaping the environment for effective technical work.
Non-technical practices, particularly in areas like security, were valued for their long-term impact, improvement of awareness, reduction of errors, increase of resilience and the consistency they provided. They were also mentioned as contributing to quality, stability and productivity of development, by for example streamlining workflows, preventing bottlenecks, and fostering collaboration. Individual participants also highlighted negative opinions, stating concerns

about the implementation of non-technical practices, including: reduced trust, inefficiencies, and lower emphasis on core tasks.
Looking ahead, participants noted that as AI automates more technical tasks, non-technical practices will become even more important, helping to shape strategy and oversee development. After technical, non-technical practices were predicted to be next, with some participants organisations already using AI to enhance these practices, improving efficiency and decision making.

RQ2

We found that some non-technical practices were implemented indeterminate of organisational differences, core security measures such as coding standards, incident response plans, and security training are close to unanimously implemented by organisations, highlighting their fundamental role in secure software development. While practices such as security risk management and metrics tracking are valued, their inconsistent adoption suggests challenges related to perceived necessity and resource demands. In contrast, attacker profile analysis and customer release risk acceptance see low adoption, being found as niche practices, indicating a preference for broad security frameworks over targeted threat modelling.
Beyond these, we found manual code reviews and penetration testing to play a critical role in ensuring security and quality of development, by sharing security standards and testing for possible vulnerabilities. Practices like pair programming and manual code reviews facilitate knowledge sharing and help with onboarding, while security teams and Security Champions help integrate security as a focus into development workflows. However, balancing security with productivity remains a challenge, particularly regarding the integration large scale and complex non-technical practices. These findings underscore the industry's focus on shared awareness, preparedness and education, alongside ongoing efforts to embed security into development processes without compromising efficiency.

RQ3

The exercise revealed that practitioners generally perceive non-technical practices as effective, with most rating a selection given as "Useful" or "Very Useful". In particular, practices to increase awareness and education, risk management, and planning were seen as effectively enhancing development processes by preventing errors, ensuring consistency, and fostering a culture of security. Additionally, practices such as business continuity planning and manual penetration testing were considered important in preparing organisations for challenges and ensuring regulatory compliance.
Effective team management emerged as a key factor, with factors such as skill overlap, shared goals, and the usage of time management tools, such as JIRA, contributing to improved collaboration and productivity. Standardisation measures, including coding standards and baseline security practices, were regarded as vital for maintaining quality and compliance with security requirements, being rated overall as "Very Useful". Moreover, practices like managing continuous integration and structured customer feedback were effective at identifying issues early and aligning deliverables with user needs. However, some non-technical practices received mixed or negative perceptions, being seen as not applicable or contributing to secure development, such as ethics hotlines. Also, although certain measures were effective in specific contexts, their informal application and lack of unified terminology meant that justifications of their effectiveness were mostly anecdotal. We found an absence of formal definitions and variable adoption across contexts, this suggests that some practices may be less universally applicable

than others.

RQ4

Participants highlighted several challenges in adopting and maintaining non-technical practices, including funding constraints, time pressures, and organisational issues. Limited resources and prioritisation of immediate returns over long-term security were major barriers, with leadership decisions playing a key role in securing funding. Tight deadlines forced developers to prioritise speed over quality and security practices, causing superficial applications, while staffing shortages and skill gaps hindered implementation of new non-technical practices.

Organisational hurdles such as administrative overhead and leadership disconnects further complicated adoption. Strict security protocols, though beneficial long-term, were seen as productivity obstacles in the short term, leading to non-compliance and workarounds by developers faced with recourse constraints.

To address these issues, participants suggested using gamification to increase engagement and adopting a balanced, selective approach to integrating non-technical practices. Gradual implementation of non-technical practices, leadership inclusion and better resource allocation were recommended to maintain productivity while enhancing security and quality.

RQ5

We found that organisations perceived as mature, integrate more non-technical practices into their secure software development processes, due to several factors. They have more resources, financial, human, and time to support these practices, while smaller organisations face challenges with limited budgets and pressure to quickly gain market share. Additionally, mature companies focus on long-term goals, prioritising sustainability and security over short-term speed and market capture, unlike startups.

As organisations grow, they develop structured processes to maintain consistency across larger teams and departments in terms of security and quality. Experience with past challenges drives the adoption of non-technical practices, improving processes over time. Regulatory pressures also play a role, as larger organisations are under more scrutiny to comply with industry standards and legal requirements, making practices like documentation and security training essential to comply with them. Overall, resource availability, long-term focus, experience, organisational structure, and regulatory pressures contribute to the higher adoption of non-technical practices in more mature organisations.

RQ6

Interviews revealed that organisations rarely conduct structured evaluations of their Software Development Life Cycles (SDLCs). Most rely on subjective assessments, such as leadership intuition and customer satisfaction, rather than formal metrics. The only example of a quantitative evaluation came from one organisation, who were able to do so due to comparing a significant change in methodology.

For non-technical practices, quantitative metrics were primarily derived from standard engineering KPIs. These served as proxy measures for effectiveness of implemented practices by evaluating team performance rather than the practices themselves. Security-specific metrics were infrequent, though some participants mentioned tracking vulnerability trends or ticket return rates as indicators of development quality.

Qualitative evaluations were more common, with case studies, audits, and feedback loops serving as primary methods. Larger organisations used comparative case studies to assess different

development approaches, while smaller ones relied on single-team evaluations. External audits played a key role in highlighting gaps and driving improvements, particularly in security and privacy. Internal and customer feedback, often gathered through agile retrospectives and team discussions, was widely valued for providing actionable insights.
Overall, while quantitative evaluations were limited and often indirect, qualitative approaches provided organisations with practical ways to assess and refine their non-technical practices. Further development and standardisation of assessment methods could improve the measurement of these practices' effectiveness in secure software development.

5.3 In-depth discussion

5.3.1 SDLC

In our study, participants highlighted that organisations typically use informal development approaches, adapted to suit their specific needs, prioritising flexibility but through ad hoc implementation facing inconsistencies. Agile was universally recognised as the industry standard, with all participants consulted employing it, but customising on top of the basic framework by integrating practices, including non-technical ones, to fit their development styles. DevOps, particularly CI/CD pipelines, were also commonly used for continuous projects to streamline development and deployment when taking on additional operational responsibilities. A novel approach mentioned was the introduction of MVPs, who were seen as a valuable tool to help refine customer requirements early, reduce late-stage changes and significantly reduce development effort. However, participants pointed out that they were occasionally forced into using hybrid SDLCs, which merge Waterfall and Agile, especially when client-driven demands impose rigid structures, limiting flexibility and disrupting workflows that lead to inefficiencies.
The literature underscores the increasing significance of security in modern software development, with Secure Software Development Methodologies (SSDMs) evolving since early 2004 to integrate security across all SDLC phases [27]. These methodologies aim to embed security at every stage, addressing vulnerabilities that may arise due to inadvertent errors or malicious activities, with most being based on Waterfall [28]. However, despite the growing emphasis on secure software practices, our findings suggest that many organisations deviate from structured SSDMs in favour of custom agile frameworks that blend different practices creating their own methodologies.
Prior research highlights that formal SSDMs often fail to gain widespread adoption due to their rigid structures and complexity, causing most organisations to adopt custom frameworks [63]. Developers frequently express frustration with hybrid SDLCs, often viewing them as inefficient adaptations that dilute the benefits of Agile or DevOps [46]. Khan et al. [27] point out that software security often falls short because many organisations rely on ad-hoc security practices rather than integrating security systematically throughout the development lifecycle. The literature also emphasises that software security remains challenging due to the disconnect between theoretical frameworks and real-world practices, leading to inconsistencies in implementation [27].
While previous research focuses on the lack of universal adoption of SSDMs, our study highlights that organisations actively attempt to balance security with operational efficiency by customising their SDLCs toward this. Our results reinforce that many organisations prefer selecting specific security activities in response to their needs, rather than following a structured methodology, similarly as what was noted by Wee [63] in his observations on the applications of SSDLC's.

Interestingly, while Agile and DevOps were widely embraced in our study, security integration remained inconsistent, echoing Khan et al.'s [27] concern that high-security integrity software is still uncommon despite ongoing advancements in SSDMs. Our study further suggests that client-driven constraints often impose rigid structures, making it difficult for teams to implement security measures effectively, an issue not explicitly explored in prior literature. The introduction of MVP's for requirements collection was also an approach previously not found in our consulted literature, providing an incentive for further research into the approach due to its significant effectiveness reported by participants.
Overall, our findings suggest that while organisations strive to integrate flexibility into their SDLCs, security is often treated as an afterthought rather than a core component of the development process. The inefficiencies observed in hybrid models indicate a need for clearer guidelines on balancing Agile adaptability with structured security requirements and enhancing communications with customers. Future research should focus on refining secure development approaches to align with real-world industry needs, ensuring that security practices are seamlessly integrated into agile workflows rather than perceived as an external burden. Additionally, fostering collaboration between academia and industry could help bridge the gap between theoretical SSDMs and their often customised practical application, ultimately improving software security without compromising development efficiency.
Sub Research Questions:

5.3.2 RQ1: "What is the perception of non-technical practices and their impact on software development?"

In our study, participants distinguished technical practices as direct code-related activities affecting the product, while non-technical practices were seen as supporting structures, processes, and organisational frameworks. Despite this distinction, both were recognised as interdependent, with non-technical practices shaping the environment for effective technical work.
Non-technical practices were valued for their long-term impact, improving awareness, reducing errors, and enhancing team productivity, contributing to overall security. They were also mentioned as contributing to quality, stability and productivity of development, by for example streamlining workflows, preventing bottlenecks, and fostering collaboration. Individual participants highlighted some negative opinions, stating concerns including: reduced trust, inefficiencies, and lower emphasis on core technical tasks.
Looking ahead, participants noted that as AI automates more technical tasks, non-technical practices will become even more important, helping to shape strategy and oversee AI development. After technical, non-technical practices were predicted to be next, with some participants organisations already using AI to enhance these practices, improving efficiency and decision-making.
There are no fixed definitions in the research of what encompasses non-technical practices in software development, as existing literature primarily focuses on individual practices or categories which are covered under the blanket term of non-technical practices.
Writing by Rhona & Flin [15], Restrepo-Tamayo & Gasca-Hurtado [44] and Matturro et al. [35] mainly reference to non-technical skills, such as communication, teamwork, and leadership, rather than non-technical practices that facilitate these skills in software development.
There are overlaps in our findings and the existing research though, such as Lacher et al. [30] describing non-technical skills are the cognitive, personal, and social skills that complement technical skills, and therefore contribute to people's performance. This aligns with participants

perspective in this study who viewed non-technical practices as structured processes that actively enhance technical practices and also mention these other aspects.
Furthermore, while previous research primarily discusses individual competencies, these findings highlight a broader organisational perspective, showing how non-technical practices such as security training, risk management, and coding standards contribute to development security and success.
A key distinction between this study and previous research is the emphasis on practices rather than skills and their place in development frameworks. While prior studies identify essential non-technical competencies, participants in this research perceived non-technical practices as the mechanisms that strengthen these competencies within software development. As AI takes over more technical tasks, developers are increasingly shifted towards organisational, strategic, and human-centric processes to manage software projects using AI effectively. This shift suggests that non-technical practices will become even more critical as technology evolves, influencing the application of AI in aspects such as security and quality.
These findings highlight the growing recognition and positive perception of non-technical practices as supporting structures, processes, and organisational frameworks vital to success and security in software development. By managing human and organisational factors and thereby improving technical practices, these practices create a more efficient, resilient, and productive development environment. However, concerns that an excessive focus on non-technical practices could introduce inefficiencies, reduce trust, or distract from core technical work should be acknowledged. This suggests that while non-technical practices are essential, they should be carefully integrated, ensuring they support rather than hinder software development processes.

5.3.3 RQ2: "Which non-technical practices are commonly involved in secure software methodologies and are used in industry?"

Our findings reveal that organisations implement a broad spectrum of non-technical practices within their SDLCs, largely influenced by organisational and human factors. Key practices such as coding standards are universally adopted (100%), underscoring their fundamental role in adding the required security and quality to development. Similarly, incident response plans and security training are widely implemented (90% each), and privacy measures like user data access controls also receive strong emphasis (90%), showing their importance across different organisations. In contrast, practices related to security metrics (70%) and risk management frameworks (50%) show moderate adoption, while ethics measures and efforts to understand human behaviour (e.g., attacker profiling, customer release risk acceptance at 20% each) are less common. This pattern suggests that organisations prioritise broad security needs and privacy over more complex and specific measures, adopting a selective approach to integrating these non-technical practices into their development processes.
The literature underscores the critical importance of clear security policies, robust review processes, and comprehensive training programmes in secure software development. Mokhberi and Beznosov [37] and Zahan [66] advocate for the use of structured security metrics and manual code reviews, while Apvrille and Pourzandi [3] demonstrate that peer reviews significantly reduce bug rates and identify vulnerabilities earlier. Further, research by Singer [51] and Naveed and Kausar [48] highlights the impact of organisational factors such as communication, knowledge management, and leadership commitment on the success of agile and secure development processes. Restrepo-Tamayo and Gasca-Hurtado [44] also emphasise that strong interpersonal relationships and effective communication are key to achieving high-quality software outcomes.

While much of the literature focuses on individual non-technical skills (communication, teamwork, leadership), our findings reveal that structured non-technical practices, which operationalise these skills are selectively implemented in industry. The close to universal adoption of practices like coding standards and incident response plans indicates that these are seen as indispensable and inexpensive to implement. Conversely, practices that are more complex or resource-intensive exhibit lower adoption rates, suggesting they are either perceived as less immediately relevant or as providing fewer returns, particularly in organisations with limited resources or lower maturity levels.
Overall, the selective yet consistent adoption of non-technical practices by organisation highlights their essential role in secure software development. Organisations that invest in these practices demonstrate a commitment to embedding security throughout the development lifecycle, balancing universally applied measures with tailored approaches to meet customer needs. This integration not only enhances security and operational efficiency but also fosters a resilient development environment capable of adapting to evolving threats. However, the lower adoption rates for more complex practices suggest that organisations must carefully consider resource allocation and implementation strategies to ensure these measures support rather than hinder overall productivity.

5.3.4 RQ3: "Which non-technical practices are perceived as effective and ineffective for practitioners?"

The exercise revealed that non-technical practices are widely seen as effective, with most presented to participants being rated as "Useful" or "Very Useful." Practices focused on enhancing awareness, security knowledge, risk management, and planning were particularly valued for preventing errors, ensuring consistency, and fostering security. Techniques such as business continuity planning and penetration testing were identified as valuable at preparing organisations for incidents and ensuring compliance. In addition, effective team management, standardisation practices, and feedback mechanisms were noted for enhancing collaboration, productivity, and continuous improvement. However, mixed perceptions emerged for some practices due to a lack of unified terminology and varying interpretations, suggesting that many non-technical practices are applied informally and their implementation based largely on personal experience rather than formal evidence.
The literature underscores that human and organisational factors are central to secure software development. John et al. [24] remind us that "software is developed for people and by people," emphasising the importance of human dynamics. Villegas & Hurtado [34]and Restrepo -Tamayo and Gasca-Hurtado [44] further highlight that effective communication, interpersonal relationships, and motivation are critical to project success Additionally, organisational research by Sommerville [53] and Al-Mushayt [1] indicates that structured processes such as staff training, knowledge sharing, and clear standards are strongly correlated with improved software quality. Also case studies, such as the one by Harischandra et al. [19], demonstrate that practices like knowledge sharing and staff training significantly influence development outcomes.
While prior research tends to focus on non-technical skills such as communication, teamwork, and leadership our study shows that practitioners perceive structured non-technical practices as the means to operationalise these skills. The literature confirms that these organisational practices are vital. However, our findings also highlight inconsistency in their application, which may stem from ambiguous definitions and varying interpretations across organisations.

This divergence suggests that while the theoretical benefits of non-technical practices are widely acknowledged, practical implementation remains uneven, with some practices being underutilised or executed in a "box-ticking" manner.
Overall, our findings highlight that non-technical practices are over all effective at securing software development, yet their effectiveness is undermined by inconsistent application and unclear standards. User data access control, security coverage measurement, coding standards, security training, security risk management and indecent response plans were all seen as very effective being rated as such by least 80% of participants. As effective contributions to development, pair programming, security champions/teams, manual code review and penetration testing were most often mentioned for their contributions to awareness and secure development successes. To maximise the benefits of these practices such as: improved security, enhanced collaboration, and increased productivity, organisations should work towards formalising and standardising their non-technical processes. This approach would ensure that the underlying human and organisational factors are consistently managed, ultimately leading to more robust and secure development outcomes. Future research should aim to clarify the terminology and develop standardised metrics for evaluating the effectiveness of these practices in real-world settings.

5.3.5 RQ4: "What are the most common challenges faced by organisations when adopting non-technical software practices?"

Participants identified key challenges in adopting and maintaining non-technical practices, including budget constraints, tight deadlines, and organisational barriers. Many organisations prioritise short-term gains over long-term investments such as security and quality, limiting resources for implementation of non-technical practices. Leadership decisions play a crucial role in planning for long-term success and securing funding and time for these practices, requiring leadership awareness and comprehension. Additionally, staffing shortages and skill gaps hinder adoption, while bureaucratic overhead and poor communication between management and development teams create further obstacles. Once implemented, strict security protocols, though beneficial long-term, can often slow down workflows, leading developers to find workarounds rather than fully integrating best practices if they see them as not effectively contributing to development.
Our findings align closely with the literature on the challenges of adopting non-technical practices in secure software development. Mokhberi and Beznosov[37] report that resource constraints such as limited budgets, time, and skilled personnel are a major obstacle, compounded by competing priorities and the absence of clear government guidelines or organisational policies. Similarly, Singer [51] and Stoddard & Kwak [29] note that even well-documented processes may fail without strong institutional support, as developers often treat security as a secondary concern when immediate returns are prioritised by leadership. Moreover, Nikitina and Mattsson [40] emphasise that successful implementation requires significant cultural change and adjustments in stakeholder behaviour, while Wee [63] underscores how inadequate staffing and resource shortages further exacerbate adoption challenges. Singer [51] suggests that non-technical strategies such as gamification can help boost engagement and overcome resistance, highlighting the need for social integration and motivation to ensure effective adoption.
While the literature highlights broad challenges, our findings provide deeper insight into their practical impact. Participants stressed that tight deadlines often force teams to prioritise speed over security and quality, a reality less emphasised in academic studies. Additionally, they

noted a disconnect between leadership and the needs of technical teams, making adoption and compliance difficult despite formal policies. Our results reveal that the informal, inconsistent application of non-technical practices along with the lack of clear definitions creates further complications, which can have effects of security, quality and productivity.
To overcome these challenges, organisations must allocate resources strategically and develop clear policies that balance immediate project needs with long-term security goals. Strong and committed leadership is essential to integrate security practices without overburdening teams with redundant non-technical practices. We found that gradual implementation, engagement strategies like gamification, and regular feedback can help ease adoption and prevent non-compliance. Ultimately, success depends on fostering a supportive organisational culture, improving awareness and communication across all levels, ensuring that non-technical practices become an integrated and sustainable part of software development that is understood and accepted by its applicants.

5.3.6 RQ5: "Why do organisations perceived as having a higher degree of maturity by developers integrate more non-technical practices in their secure software development process?"

We found that mature organisations integrate more non-technical practices into their secure software development processes and do so for several reasons. Firstly, they have greater financial, human, and time resources that enable them to invest in long-term security and sustainability, in contrast to smaller organisations that often prioritise short-term gains and rapid market capture. As companies grow, they need to develop structured processes that maintain consistency across larger teams, using experience from past challenges to drive the adoption of non-technical practices such as comprehensive documentation, security training, and formalised risk management. Regulatory pressures also play a significant role, as larger organisations must more often rigidly comply with industry standards, legal and customer requirements making the integration of certain practices necessary to meet compliance goals.
Our findings align with previous research that emphasises organisational maturity as a key factor in adopting non-technical practices. Fontana et al. [16] found that higher agile maturity correlates with enhanced non-technical capabilities like collaboration, communication, and self-organisation. Similarly, Lester et al. [33] observed that as companies grow, increased communication complexity necessitates more formalised practices to maintain effective coordination. Furthermore, established maturity frameworks such as OWASP SAMM, BSIMM, CMMC 2.0, and CMM provide structured approaches for improving security practices, while John et al. [24] and Lavallée and Robillard [31] highlight that mature organisations benefit from formalised processes that buffer the impact of less proficient developers as the developer count increases in the organisation. Mokhberi and Beznosov [37] note that larger companies tend to foster stronger security cultures, and Jiang et al. [23] provide empirical evidence linking higher process management maturity with improved project outcomes.
While the literature broadly recognises the advantages of organisational maturity, our findings add nuance by showing that mature organisations not only possess more resources but also actively prioritise long-term security and sustainable practices. Our participants indicate that in these environments, non-technical practices are seen as integral to maintaining quality and compliance, rather than as optional add-ons. This contrasts with smaller organisations, where immediate technical deliverables often take precedence, and where the lack of structured processes and a unified security culture can lead to inconsistent practice adoption. Thus, our

results extend previous research by highlighting the interplay between resource availability, organisational culture, and regulatory pressures in driving the integration of non-technical practices.
Overall, the data suggest that organisational maturity is a indicator for the effective adoption of non-technical practices in secure software development. Mature organisations, with their robust resources and structured processes, are better positioned to implement comprehensive security measures that go beyond technical fixes. This not only strengthens their overall security posture but also improves project outcomes by mitigating risks associated with inconsistent practices and skill gaps. For smaller organisations, these findings underscore the importance of investing in organisational development and process maturity to build a culture that values non-technical long-term security and continuous improvement practices.

5.3.7 RQ6: What methods are used by organisations to estimate the effectiveness of non-technical practices in the context of developing secure software?

Our findings indicate that organisations rarely conduct structured evaluations of their Software Development Life Cycles (SDLCs) to measure their effectiveness or that of the non-technical practices used in them. Instead, they predominantly rely on subjective assessments such as leadership intuition and customer feedback to gauge impact. Quantitative evaluations are mostly limited to standard engineering KPIs, such as ticket throughput, story point burn-down or system uptime, which serve as indirect proxies rather than directly capturing the effectiveness of these practices by comparing values before and after. Only a few of the organisations evaluated employed security-specific metrics, such as tracking vulnerability trends or security ticket return rates, while qualitative methods such as case studies, external audits, and agile retrospectives remain the primary evaluators of effectiveness for non-technical practices and their adoption success.
These empirical observations reflect the challenges highlighted in the literature. Kudriavtseva & Gadyatskaya [28] identified significant gaps in secure development methodologies, emphasising that the lack of evidence of their effectiveness, which creates a critical need for improved measurement practices. Similarly, Ita Ryan [46] notes that the absence of universally accepted metrics for evaluating software lifecycle security leads to difficulties in comparing and estimating the impact of non-technical practices. The literature further reveals that non-technical factors and human behaviours, such as organisational culture and security awareness, are inherently difficult to quantify and are often measured indirectly via surveys and interviews [44]. Research by Votipka et al. [61] also illustrates that even structured security training may fail to produce the expected security outcomes, reinforcing the limitations of current quantitative approaches. Frameworks like SecEval [49] do call for measurable security indicators and John et al. [24] call for systematic approaches to approach this issue, yet these adoptions in practice remains minimal.
While the literature predominantly criticises the absence of formal, universal effectiveness and security metrics, our findings reveal that organisations are attempting to measure effectiveness through pragmatic, albeit indirect, methods. The heavy reliance on engineering KPIs and qualitative feedback indicates a passive approach, or one where the focus is on enumerating security activities rather than on capturing their true impact. This observation aligns with Ita Ryan's [46] critique of compliance-driven evaluations, which tend to prioritise procedural completeness over genuine security improvements. Additionally, the challenges of measuring

non-technical practices in diverse contexts, such as organisational and open-source settings, as noted by Restrepo-Tamayo and Gasca-Hurtado [44], further underline the gap between theoretical frameworks and practical implementation.
In summary, while organisations do attempt to assess the effectiveness of non-technical practices, their methods remain largely informal and indirect, failing to capture the security-specific impact of these initiatives. This gap underscores the urgent need for the development of structured, standardised measurement frameworks that integrate security-specific metrics into agile and DevOps workflows and individual non-technical practices. Enhanced evaluation methods, supported by closer collaboration between industry and academia, could transform qualitative feedback and proxy indicators into actionable, empirical outcomes to measure security and other factors. Ultimately, establishing robust evaluation mechanisms is critical for advancing secure software development and ensuring that non-technical practices contribute effectively to overall organisational security.

5.4 Recommendations

5.4.1 High Feasibility

Formal Yet Agile Strategy
Recommendation: Set clearly defined goals, methodologies, and roles, while maintaining flexibility to adapt to changing circumstances.
Why: Our research highlighted the need for a balance between structured planning and adaptability. Participants voiced that clearly defined goals and roles in development enhance alignment and accountability, yet they need the freedom to adapt, allowing teams to adjust to changing circumstances.
Benefits: Provides clarity, aligns teams with business objectives, enhances accountability, yet allows for rapid adjustments to market or project changes. This is an established process that requires minimal resources beyond documentation and team alignment, ensuring seamless integration into existing workflows.

Continuous Improvement
Recommendation: Continuously evaluate processes, collect feedback, and test new approaches for ongoing improvement.
Why: Our research revealed that teams who consistently evaluate their processes and incorporate feedback are better able to identify inefficiencies and drive innovation. Participants noted that this low-cost, ongoing approach not only enhances quality and security over time but also keeps the development process aligned with evolving standards and innovations.
Benefits: Identifies inefficiencies, fosters innovation, improves quality and security over time, and keeps the SDLC aligned with evolving standards. It's a low-cost, ongoing activity that can be embedded into existing workflows, fostering continuous refinement with minimal investment.

Formalised & Documented Practices
Recommendation: Create and maintain clear, up-to-date documentation with well-defined technical and non-technical practices.
Why: Many participants in mature organisations, emphasised that clear, up-to-date documentation not only enhances process adherence and accountability but also reinforces security by providing standardised guidance on risk response, mitigation, and secure practices. They insist that maintaining such comprehensive documentation is essential, underpinning organisational

resilience and comprehensibility of operational and security measures. This is especial useful with newer developers entering the organisation and enforces process not personal bound development.
Benefits: Enhances clarity, consistency, and accountability, and provides a solid reference for training and process adherence. This recommendation is straightforward to implement and can be achieved with existing resources, requiring minimal effort to maintain.

Enhance Team & Security Collaboration
Recommendation: Implement effective team management strategies, such as promoting skill overlap, aligning shared goals and embed security-focused collaboration practices such as pair programming and regular manual code reviews as part of the development.
Why: Participants often noted that bottlenecks significantly hindered their development, prompting them to actively enhance skill and knowledge sharing to not only make development more resilient but also disseminate security knowledge. This approach spreads awareness through communication, and creates continuous security oversight through shared knowledge and skills.
Benefits: Boosts productivity, enhances communication, improves knowledge sharing, ensures continuous security oversight, and embeds security considerations throughout the development process. It has high impact with low to moderate resource investment, achievable within existing team structures.

Secure Leadership Buy-In
Recommendation: Advocate for leadership support to allocate resources for non-technical practices, emphasising their long-term benefits over short-term gains.
Why: Participants revealed that leadership was seen as very important for the implementation of non-technical practices, the planning and assignment of resources for non-technical practices. Organisations lacking leadership commitment and understanding of non-technical practices were prevented from implementing and maintaining them effectively.
Benefits: Ensures strategic alignment, secures necessary funding, and fosters an organisational culture that values sustainable improvements. While it requires some advocacy, it is typically achievable through internal communication and alignment with leadership goals.

Address Skill Gaps
Recommendation: Provide targeted training programmes to equip teams with the necessary skills to use non-technical practices, removing complexity, improving current development practices and fostering smoother adoption of future practices.
Why: Our research highlighted that skill gaps in often hinder the effective adoption and application of new non-technical practices. Participants noted that structured training programmes help bridge these gaps, enhancing team efficiency and skills by providing users with the knowledge to effectively apply these practices.
Benefits: Enhances team capabilities, reduces the learning curve, and builds a more versatile and competent workforce, to develop more secure software. The feasibility of this recommendation is high as training programs can be implemented with relative ease through workshops, courses, or mentoring, requiring moderate investment.

Use Gamification for Engagement
Recommendation: Incorporate gamified elements to encourage adherence to best practices

in non-technical tasks, particularly in security and workflow management.
Why: Participants revealed that from their experience incorporating gamified elements can significantly boost team engagement and adherence to best practices. Participants observed that gamification not only makes routine tasks more engaging but also enhances vigilance in critical areas such as security and workflow management.
Benefits: Increases team motivation, improves practice adoption, and adding gamified elements makes routine tasks more engaging. Basic gamification is achievable with existing tools and resources but requires creativity and planning, making it relatively easy to implement if designed with the organisational culture in mind.

5.4.2 Medium Feasibility

Optimised Workflows & Continuous Improvement
Recommendation: Optimise organisational workflows by eliminating bottlenecks and integrating structured feedback mechanisms and continuous integration.
Why: Many participants found in their organisations that inefficient workflows and hidden bottlenecks hinder timely delivery, quality and security responses. Respondents noted that eliminating these obstacles and integrating structured feedback with continuous integration streamlines operations and fosters a culture of continuous improvement.
Benefits: Increases efficiency, enables early issue detection, enhances product security, and fosters a culture of ongoing learning and process optimisation. This recommendation requires commitment and effort from various teams to remove bottlenecks, and integration of feedback mechanisms can be resource-intensive at first.

Long-Term & Security-Driven Practices
Recommendation: Implement standard security practices alongside robust security awareness programmes and prioritise long-term initiatives when resources allow such.
Why: We found that standard security practices combined with robust security awareness programmes are essential for mitigating risks and ensuring sustainable development. Respondents stressed that although these initiatives require a moderate investment in time and resources, their long-term benefits in enhancing team productivity and product security make them indispensable.
Benefits: Reduces risks, improves team productivity, enhances product security, and ensures sustainable development outcomes. While non-technical security practices are essential, implementing them may require time, investment in tools, and comprehensive training, which demands a moderate commitment of resources.

Integrated Evaluation & Standardised Assessment
Recommendation: Develop structured evaluation frameworks for non-technical practices that incorporate both quantitative and qualitative assessment methods.
Why: Structured evaluation frameworks combining quantitative and qualitative methods yield objective insights and facilitate consistent benchmarking. Respondents emphasised that this balanced approach enhances decision-making and continuous improvement, despite requiring a moderate investment in planning and integration.
Benefits: Provides objective, data-driven insights, ensures uniform assessment across teams, facilitates benchmarking, balances numerical analysis with contextual insights, enhances decision-making, and supports continuous improvement. This requires planning, integration with exist-

ing tools, and standardisation, all of which require a moderate level of resources and coordination, but increases in complexity with individualisation and depth.

Quantitative & Security Metrics
Recommendation: Design specific, measurable indicators for non-technical practice effectiveness and introduce security metrics for development.
Why: Absence of clear performance benchmarks for non-technical practice effectiveness and security can obscure issues and improvement opportunities. Participants noted that while establishing specific, measurable indicators and security metrics can be resource intensive, it significantly enhances accountability, risk management and vulnerability detection.
Benefits: Offers clear performance benchmarks, enhances accountability, improves vulnerability detection, and ensures focused security improvements. Setting up systems to monitor and track performance of practices can be resource intensive, requiring significant time and effort to establish and maintain these metrics, yet they provide significant benefits.

5.4.3 Low Feasibility

AI Integration into Non-technical Practices
Recommendation: Evaluate and, where beneficial, integrate AI tools to support and enhance non-technical practices, aiding in decision-making and workflow automation.
Why: AI tools are predicted to automate many technical and enhance non-technical practices, making introduction and maintenance less complex and recourse intensive. Respondents emphasised that the complexity and substantial investment in technology and expertise required for AI initiatives render them challenging to implement, despite their strategic potential.
Benefits: Provides data-driven insights and improvement suggestions, streamlines repetitive tasks and supports more informed strategic management, in terms such as security. AI integration is resource-intensive and requires significant investment in technology, expertise, and time for successful adoption, making it a high-complexity initiative.

6 Limitations

Throughout this research, several limitations were encountered that may have influenced the findings and their interpretations. Several measures were implemented to limit these challenges, as outlined below. In order to interpret and apply the findings, the context and limitations of the study in which they were gathered should be acknowledged and understood.

6.1 Limited Generalisability

One of the primary limitations of this study is the limited applicability of the findings. Given the relatively small sample size, the results may be specific to the organisational contexts of the participants and may not be generalisable to all organisations secure software development approaches. While the study provides valuable insights into how non-technical practices are perceived and implemented, these findings cannot be assumed to represent the experiences or challenges faced by all organisations, particularly those operating in different industries, regulatory environments, or at varying levels of maturity. To mitigate this limitation, we selected a diverse set of organisations and cross-referenced our findings with existing literature to contextualise our results within the broader field.

6.2 Interviewer Bias

Throughout the data collection process, we remained aware of the potential for interviewer bias [20] to influence participants responses. Despite efforts to remain neutral and objective, our own perspectives on non-technical security practices may have shaped the phrasing of questions, the interpretation of responses, or our interactions with participants. This could have introduced bias into the data, influencing how the effectiveness of these practices was perceived and reported. To limit this bias, we employed a semi-structured interview protocol, conducted a mock interview, and held regular debriefing sessions to reflect on and adjust our approach throughout the study.

6.3 Participant Bias

Participant bias [20], including selective recollection and social desirability bias, may have affected the reliability of the data. Participants may have unintentionally misremembered details or presented non-technical practices within their organisation in a more favourable light, particularly when discussing security culture, leadership engagement, or compliance efforts. This introduces the risk that some accounts may be incomplete or unintentionally inaccurate, impacting the validity of the findings. In response, we assured participants of confidentiality and emphasised the importance of natural feedback, which helped to encourage honest and accurate responses.

6.4 Small Sample Size

The small sample size posed a challenge to the diversity and representativeness of the data. While the study provided in-depth qualitative insights into the role of non-technical practices in secure software development, the findings cannot be assumed to reflect broader organisational trends. Organisations differ in size, structure, and maturity as well as other factors, and the

perspectives captured in this study may not be fully applicable to other contexts. This limitation may be particularly relevant for sectors with strict regulatory requirements or for organisations with significantly different development approaches. We tried to limit this by interviewing participants from a wide range of organisational contexts, positions and organisations to create a broad field of opinions.

6.5 Subjectivity and Interpretation

A key limitation of this qualitative research lies in the subjective nature of both data collection and analysis. The interpretation of interview data is inherently influenced by our perspectives, making complete objectivity difficult to achieve. Although independent coding was conducted to mitigate bias, other researchers may interpret the same data differently, leading to variations in conclusions. We attempted to address this through independent coding and discussions on interpretation thereby enhancing the credibility of our analysis.

6.6 Sensitivity of the Topic and Participant Disclosure

A key limitation of this study was the sensitivity of the topic, which may have affected the level of openness and disclosure from participants. Discussions surrounding non-technical security practices often intersect with compliance and organisations security policies which may have led participants to withhold certain details due to confidentiality concerns. This reluctance may have resulted in cautious or partial responses, limiting the depth of insights into challenges such as resistance to security training, gaps in leadership support, or inefficiencies in security governance. To counter this, we implemented strict confidentiality protocols and ensured a secure environment for data collection, which encouraged participants to share more openly within the bounds of ethical research practice.

6.7 Conclusion: Limitations

We aimed to conduct this study as thoroughly and objectively as possible, but some limitations remain. Despite our efforts to reduce bias, analyse data carefully, and encourage honest responses, factors such as the small sample size, subjective interpretations, and the sensitive nature of the topic may have influenced the findings. These challenges are common in qualitative research and should be considered when interpreting the results, though we took important steps to limit these limitations and provide useful insights into non-technical practices in secure software development. However, we believe this study offers useful insights into non-technical practices in secure software development used in organisations and highlights important areas for future research.

7 Conclusions

This study explored practitioner's perspectives on non-technical practices, their effectiveness and adoption in different organisations secure software development approaches, as well as how these organisations assessed and justified their effectiveness. The data collected sheds light on how these practices are used in the field, and the further discussion on how their application aligns with insights from existing literature.

We found that most organisations use custom and informal Agile or DevOps-based SDLCs, tailored with non-technical practices. Participants mentioned issues with Hybrid Agile-Waterfall models and dictated development approaches though, which often disrupt existing workflows. The findings also revealed that most practitioners shared similar definitions of non-technical practices and viewed them as equally important for contributing to the development process and its security, alongside technical practices. As AI continues to automate technical tasks, the importance of non-technical practices is expected to grow, with these practices likely becoming the next focus.

The paper provides a comprehensive documentation of all non-technical practices used across participants organisations. It also highlights practices such as Manual code review, Manual Penetration Testing, Pair Programming and Security Champion/Team as some of the most used non-technical practices. In addition, results from exercises showed Coding standards (100% usage), Security Training, Incident Response Plan, User Data Access Controls (90% usage) with near universal adoption across organisations and others.

In terms of effectiveness, Awareness and Education, Risk Management and Planning, Documentation and Process Structuring, Team and People Management, Standardisation and Best Practices, Feedback and Continuous Improvement were named as categories with the most important and effective practices, in this respective order. The exercise showed that high adoption correlated with perceived effectiveness, though even some less adopted practices from the exercise were also seen as effective.

We found multiple barriers to the implementation and application of non-technical practices in organisations, including lack of funding, time, low developer employment, organisational hurdles, high practice complexity, reduced productivity, non-compliance, low engagement and practice overload.

In terms of non-technical practices in different organisations, mature organisations were found to adopt more non-technical practices due to access to resources and compliance needs. Startups focused more on providing output and market share, more often disregarding non-technical practices in favour of direct output and gaining market share faster.

Evaluation of non-technical practices was found to be mostly qualitative, based on retrospectives and audits. Quantitative metrics are rare and often indirectly associated to practices. Organisations tend to adopt and measure these practices informally, relying on experience and intuition rather than formal evaluation.

The findings from this thesis offer several actionable recommendations for organisations aiming to strengthen their secure software development processes through non-technical practices. Non-technical practices shared across different organisational types are highlighted which organisations can pull inspiration from their own methodology to increase their secure development processes.

Issues in adoption and application that other organisations have faced are also highlighted for them to prepare against and review their own development methods for, to increase their own organisations success.

This study underscores the importance of recognising not only the value of non-technical skills but also the need for systematic methods to measure and justify their effectiveness in practice. Future research should focus on developing and validating robust methodologies for quantifying the impact of non-technical practices. Comparative studies across organisations at varying stages of maturity could further help establish whether practices discussed in the literature are overlooked or if their effectiveness diminishes when applied in real-world scenarios. Additionally, investigating innovative approaches for implementing and measuring these practices could contribute to the establishment of standardised metrics and best practices that would benefit the field as a whole.

RQ: "Exploring the role and effectiveness of non-technical/auxiliary practices in secure software development, currently used in organisations."

Non-technical practices are widely recognised by developers and managers as playing a crucial role in organisations secure software development (SSD) by effectively managing human and organisational factors and creating an environment in which technical practices can deliver secure, high-quality software. While most of these practices are considered highly effective, a few are not, with the success often being contingent on the context and rigour in which they are implemented. Often non-technical practices are only evaluated qualitatively and informally though, making it hard to justify their implementation and maintenance.

References

[1] Omar S Al-Mushayt. *An empirical investigation of factors influencing the successful treatment of organisational issues in information systems development*. PhD thesis, © Omar Saeed Al-Mushayt, 2000.

[2] Reza Alavi, Shareeful Islam, and Haralambos Mouratidis. An information security risk-driven investment model for analysing human factors. *Information & Computer Security*, 24(2):205–227, 2016.

[3] Axelle Apvrille and Makan Pourzandi. Secure software development by example. *IEEE Security & Privacy*, 3(4):10–17, 2005.

[4] Renana Arizon-Peretz, Irit Hadar, and Gil Luria. The importance of security is in the eye of the beholder: Cultural, organizational, and personal factors affecting the implementation of security by design. *IEEE Transactions on Software Engineering*, 48(11):4433–4446, 2021.

[5] Hala Assal and Sonia Chiasson. 'think secure from the beginning' a survey with software developers. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pages 1–13, 2019.

[6] Black Duck Software. What is BSIMM? (Building Security In Maturity Model), 2006. Accessed: 2024-12-06.

[7] Forbes Business Council. The high cost of security vulnerabilities: Why observability is the solution, 2023. Accessed: 2025-02-27.

[8] Deloitte. Secure software development lifecycle, 2023. Accessed: 2025-02-27.

[9] Udaya Mohan Devadas and Yovini Yashodara Dharmapala. Soft skills evaluation in the information technology and business process management industry in sri lanka: skills, methods and problems. *International Journal of Economics Business and Human Behaviour*, 2(3), 2021.

[10] Barbara DiCicco-Bloom and Benjamin F Crabtree. The qualitative research interview. *Medical education*, 40(4):314–321, 2006.

[11] Neil F Doherty and Malcolm King. The consideration of organizational issues during the systems development process: an empirical analysis. *Behaviour & information technology*, 17(1):41–51, 1998.

[12] Tore Dyba. An empirical investigation of the key factors for success in software process improvement. *IEEE transactions on Software Engineering*, 31(5):410–424, 2005.

[13] European Commission. Nis2 directive, 2025. Accessed: 22 March 2025.

[14] European Insurance and Occupational Pensions Authority (EIOPA). Digital operational resilience act (dora), 2025. Accessed: 22 March 2025.

[15] Rhona Flin and Paul O'Connor. *Safety at the sharp end: a guide to non-technical skills*. CRC Press, 2017.

[16] Rafaela Mantovani Fontana, Isabela Mantovani Fontana, Paula Andrea da Rosa Garbuio, Sheila Reinehr, and Andreia Malucelli. Processes versus people: How should agile software development maturity be defined? *Journal of Systems and Software*, 97:140–155, 2014.

[17] Gartner, Inc. Gartner predicts nearly half of cybersecurity leaders will change jobs by 2025. Technical report, Gartner, 2023.

[18] ATLAS.ti Scientific Software Development GmbH. Atlas.ti - the qualitative data analysis & research software, 2024. Version 24.2.1.32227.

[19] Jayani Harischandra and Sunesh Hettiarachchi. Organizational factors that affect the software quality a case study at the engineering division of a selected software development organization in sri lanka. In *2020 IEEE 7th International Conference on Industrial Engineering and Applications (ICIEA)*, pages 984–988. IEEE, 2020.

[20] Donald C Hildum and Roger W Brown. Verbal reinforcement and interviewer bias. *The Journal of Abnormal and Social Psychology*, 53(1):108, 1956.

[21] Judith A Holton. The coding process and its challenges. *The Sage handbook of grounded theory*, 3:265–289, 2007.

[22] Bassam A. Hussein and Kristin H. J. Hafseld. Impact of organizational factors. In *The 7th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications 12-14 September 2013, Berlin, Germany*, 2013.

[23] James J Jiang, Gary Klein, Hsin-Ginn Hwang, Jack Huang, and Shin-Yuan Hung. An exploration of the relationship between software development process maturity and project performance. *Information & Management*, 41(3):279–288, 2004.

[24] Michael John, Frank Maurer, and Bjørnar Tessem. Human and social factors of software engineering: workshop summary. *ACM SIGSOFT Software Engineering Notes*, 30(4):1–6, 2005.

[25] Jason B Kaminsky. Impact of nontechnical leadership practices on it project success. *Journal of Leadership Studies*, 6(1):30–49, 2012.

[26] Hossein Keramati and Seyed-Hassan Mirian-Hosseinabadi. Integrating software development security activities with agile methodologies. In *2008 IEEE/ACS International Conference on Computer Systems and Applications*, pages 749–754. IEEE, 2008.

[27] Rafiq Ahmad Khan, Siffat Ullah Khan, Habib Ullah Khan, and Muhammad Ilyas. Systematic literature review on security risks and its practices in secure software development. *ieee Access*, 10:5456–5481, 2022.

[28] Arina Kudriavtseva and Olga Gadyatskaya. Secure software development methodologies: a multivocal literature review. *arXiv preprint arXiv:2211.16987*, 2022.

[29] Young Hoon Kwak and Jared Stoddard. Project risk management: lessons learned from software development environment. *Technovation*, 24(11):915–920, 2004.

[30] Lisa Lacher, Gursimran Walia, Kendall Nygard, Fabian Fagerholm, Max Pagels, and Jürgen Münch. A behavior marker for measuring non-technical skills of software professionals: An empirical study. *International journal of software engineering and knowledge engineering*, 25(09n10):1733–1738, 2015.

[31] Mathieu Lavallée and Pierre N Robillard. The impacts of software process improvement on developers: A systematic review. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 113–122. IEEE, 2012.

[32] Mathieu Lavallée and Pierre N Robillard. Why good developers write bad code: An observational case study of the impacts of organizational factors on software quality. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 1, pages 677–687. IEEE, 2015.

[33] NG Lester, F George Wilkie, Donald McFall, and MP Ware. Investigating the role of cmmi with expanding company size for small-to medium-sized enterprises. *Journal of Software Maintenance and Evolution: Research and Practice*, 22(1):17–31, 2010.

[34] Liliana Machuca-Villegas and Gloria Piedad Gasca-Hurtado. Towards a social and human factor classification related to productivity in software development teams. In *Trends and Applications in Software Engineering: Proceedings of the 8th International Conference on Software Process Improvement (CIMPS 2019)*, pages 36–50. Springer, 2020.

[35] Gerardo Matturro, Florencia Raschetti, and Carina Fontán. A systematic mapping study on soft skills in software engineering. *J. Univers. Comput. Sci.*, 25(1):16–41, 2019.

[36] Microsoft. Windows SDK: Windows Filtering Platform, 2024. Accessed: 2025-03-08.

[37] Azadeh Mokhberi and Konstantin Beznosov. Sok: human, organizational, and technological dimensions of developers' challenges in engineering secure software. In *Proceedings of the 2021 European Symposium on Usable Security*, pages 59–75, 2021.

[38] Nachiappan Nagappan, Brendan Murphy, and Victor Basili. The influence of organizational structure on software quality: an empirical case study. In *Proceedings of the 30th international conference on Software engineering*, pages 521–530, 2008.

[39] National Institute of Standards and Technology. Secure Software Development Framework (SSDF), Version 1.1. Technical Report NIST SP 800-218, NIST, 2022. Accessed: 2025-03-08.

[40] Natalja Nikitina and Mira Kajko-Mattsson. Guiding the adoption of software development methods. In *Proceedings of the 2014 international conference on software and system process*, pages 109–118, 2014.

[41] OWASP Foundation. Software Assurance Maturity Model (SAMM), 2003. Accessed: 2024-12-06.

[42] Mark C Paulk, Bill Curtis, Mary Beth Chrissis, Charles V Weber, et al. *Capability maturity model for software*. Citeseer, 1991.

[43] Laleh Pirzadeh. Human factors in software development: a systematic literature review. Master's thesis, CHALMERS UNIVERSITY OF TECHNOLOGY, 2010.

[44] Luz Marcela Restrepo-Tamayo and Gloria Piedad Gasca-Hurtado. Human aspects in software development: a systematic mapping study. In *International Conference on Collaboration Technologies and Social Computing*, pages 1–22. Springer, 2022.

[45] Luz Marcela Restrepo-Tamayo and Gloria Piedad Gasca-Hurtado. Non-technical factors in software engineering within the context of industry 4.0. In *New Perspectives in Software Engineering*, pages 89–103. Springer, 2024.

[46] Ita Ryan, Utz Roedig, and Klaas-Jan Stol. Measuring secure coding practice and culture: A finger pointing at the moon is not the moon. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 1622–1634. IEEE, 2023.

[47] Carolyn B Seaman and Victor R Basili. Communication and organization in software development: An empirical study. *IBM Systems Journal*, 36(4):550–563, 1997.

[48] Naveed Shehzad and Maryam Kausar. Organizational factors impacting agile software development -a systematic literature. *American Scientific Research Journal for Engineering, Technology, and Sciences*, pages 2313–4402, 12 2021.

[49] Adam Shostack, Matthew Smith, Sam Weber, and Mary Ellen Zurko. Empirical evaluation of secure development processes (dagstuhl seminar 19231). In *Dagstuhl Reports*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[50] Stacy Simpson. Safecode whitepaper: Fundamental practices for secure software development 2nd edition. In *ISSE 2014 Securing Electronic Business Processes: Highlights of the Information Security Solutions Europe 2014 Conference*, pages 1–32. Springer, 2014.

[51] Leif-Gerrit Singer. *Improving the adoption of software engineering practices through persuasive interventions*. Lulu. com, 2013.

[52] RL Snyder and David L Bish. Quantitative analysis. *Modern powder diffraction*, 20:101–144, 1989.

[53] Ian Sommerville and Tom Rodden. Human, social and organisational influences on the software process. *Software Process*, 4:89–100, 1996.

[54] Statista. Spending in the global security technology and services market by segment, 2025. Accessed: 2025-02-27.

[55] Bjarne Stroustrup. Bjarne stroustrup's quotes. `https://www.stroustrup.com/quotes.html`. Accessed: 2025-01-16.

[56] O TI, D ECI, M AKIN, A LE, and D ERS. *The non-technical skills for surgeons (NOTSS) system handbook v1. 2*. NOTSS, 2006.

[57] Micheal Tuape and Yirsaw Ayalew. Factors affecting development process in small software companies. In *2019 IEEE/ACM Symposium on Software Engineering in Africa (SEiA)*, pages 16–23. IEEE, 2019.

[58] TÜV SÜD. Prüfung der softwarequalität, 2025. Accessed: 22 March 2025.

[59] U.S. Department of Defense CIO. About the Cybersecurity Maturity Model Certification 2.0 (CMMC 2.0), 2020. Accessed: 2024-12-06.

[60] Verizon. 2024 data breach investigations report (dbir). Technical report, Verizon, 2024. Accessed: 2025-01-15.

[61] Daniel Votipka, Kelsey R Fulton, James Parker, Matthew Hou, Michelle L Mazurek, and Michael Hicks. Understanding security mistakes developers make: Qualitative analysis from build it, break it, fix it. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 109–126, 2020.

[62] Patrick E Waterson, Chris W Clegg, and Carolyn M Axtell. The interplay between cognitive and organizational factors in software development. *Human—Computer Interaction: Interact'95*, pages 32–37, 1995.

[63] Alex Wee, Arina Kudriavtseva, and Olga Gadyatskaya. "have heard of it": A study with practitioners on adoption of secure software development frameworks. In *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 626–633. IEEE, 2024.

[64] World Economic Forum. The global risks report 2024. Technical report, World Economic Forum, 2024.

[65] Shundan Xiao, Jim Witschey, and Emerson Murphy-Hill. Social influences on secure development tool adoption: why security tools spread. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, pages 1095–1106, 2014.

[66] Nusrat Zahan, Shohanuzzaman Shohan, Dan Harris, and Laurie Williams. Do software security practices yield fewer vulnerabilities? In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 292–303. IEEE, 2023.

8 Appendix

8.1 Abbreviations and Acronyms:

AI – Artificial Intelligence
BSc – Bachelor of Science
BSIMM – Building Security In Maturity Model
CEO – Chief Executive Officer
CI/CD – Continuous Integration / Continuous Deployment
CMM – Software Capability Maturity Model
CMMC 2.0 - Cybersecurity Maturity Model 2
COE – Correction of Errors
CTO – Chief Technology Officer
DevOps – Development and Operations
DORA – Digital Operational Resilience Act
GDPR – General Data Protection Regulation
ICT – Information and Communications Technology
IoT – Internet of Things
KPI – Key Performance Indicator
MSc – Master of Science
MVP – Minimum Viable Product
NIS2 – Network and Information Systems Directive 2
OWASP – Open Worldwide Application Security Project
PMP – Project Management Plan
QA – Quality Assurance
RCA – Root Cause Analysis
RQ – Research Question
SAAS – Software as a Service
SAMM – Software Assurance Maturity Model
SDLC – Software Development Life Cycle
SSD – Secure Software Development
SSDM – Secure Software Development Methodology
SSDLC – Secure Software Development Life Cycle
SSE – Secure Software Engineering
TÜV – Technischer Überwachungsverein (German Technical Inspection Association)

8.2 Questions:

Demographic information:
Education:
Years of experience:
Type of Employment: (Freelancer/Consulting/Full-time)
Size of company:
Company sector:
Market Position:
Introduction:

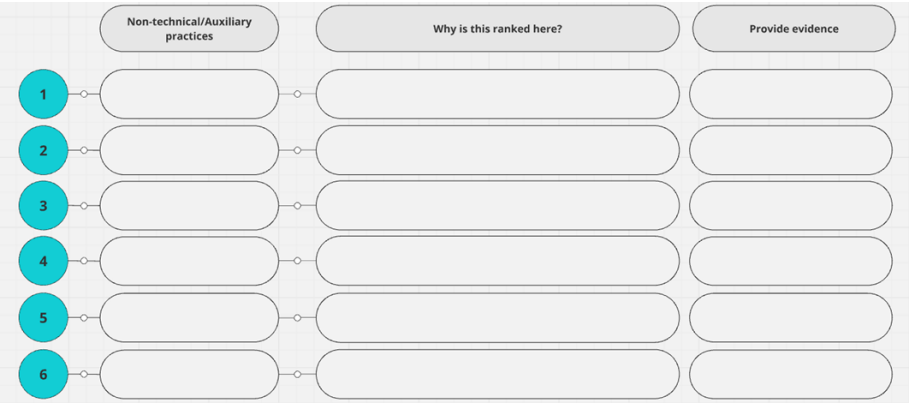

Figure 4: Miro

1)How would you describe your role and its importance within the organisation's structure and objectives?
2)Could you outline your main responsibilities, concerning software development and security?
SSDM's/SSDL's
3)What SDLC models have you worked with (e.g., (Waterfall, Agile, DevOps)?
4)Does your organisation employ a custom or adopted methodology in its development processes?
5)Can you describe the development process you follow and how mature do you think it currently is?
5a)What evidence do you use or do to gauge the maturity of your methodology?
Participant Auxiliary/Non-Technical practices Opinion:
6)What activities did you implement in your development process to improve the security and quality of development?
7)How would you interpret Auxiliary/Non-Technical practices in software development and how do they differ from technical practices?
8)Can you give examples of auxiliary/non-technical practices you have encountered or used during a project?
Exercise (Miro)
8b) Can you rank the non-technical practices have you used in the past; how do they rank in their effectiveness?
9)How do you perceive the importance of Auxiliary/Non-Technical practices as part of development methodologies?
9a)How do you think they contribute to the overall improvement of the development process?
9b)How can you observe this improvement and is it measures?
10)What specific tools *(JIRA, Microsoft Threat Modelling Tool)* and practices related to Auxiliary/Non-Technical practices, do you interact with in your development to ensure secure software development?
Exercise (Rate the practice)
Definitions:
Hindering: This rating indicates that the practice not only fails to contribute positively but actually creates obstacles or drawbacks in your work.
Not Useful: This rating is for practices that have no significant positive or negative impact. They neither help nor hinder your work.

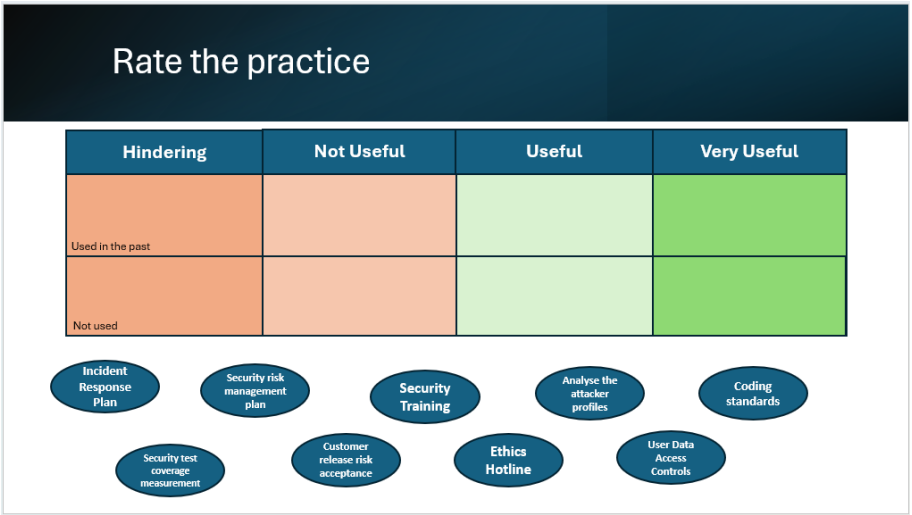

Figure 5: Rate the practice

Useful: Practices rated as useful are beneficial and contribute positively to your work. They help in some way, but may not be critical
Very Useful: This rating is for practices that are extremely beneficial and play a crucial role in your success. They are essential to achieving your goals effectively.
Effectiveness:
11)How does your organization currently measure the effectiveness of development practices used during the development process?
12)Can you provide examples of key performance indicators or metrics used to assess the impact of security and quality practices on software development outcomes?
13)In your experience, what methodologies or tools does your organization rely on to evaluate the success of implemented security and privacy measures, and
13a)How do these assessments inform future development efforts?
Conclusion:
14)What would non-technical practice would you recommend other organisations to have in their development methodologies?
15)What is stopping your organisation from implementing more Auxiliary/Non-Technical practices to improve software security?
16)If given the resources, would you advocate for the incorporation of additional Auxiliary/Non-Technical practices into your organization's development?
17)Could you recommend me anyone that might be interested discussing the topic in an interview?
Thank you for attending the interview.

8.3 Code Book:

Project: s3731545 Thesis Bruno Smith (Snapshot 2024-11-30 13:36:33)
Report created by Bruno Smith on 25.02.2025
RQ 0 (Introduction: SDLC and demographics in Organizations)
15 Codes:

- **Demographics**

ID	Section:	Key Takeaway:	Answered (Y/N):
1	Introduction		
2	Development approaches		
3	Auxiliary/Non-Technical practices		
4	Miro		
5	Rate the practice		
6	Effectiveness		
7	Conclusion		

  - Demographics: Role in the company
  - Demographics: Market position
  - Demographics: Education
  - Demographics: Company Size
  - Demographics: Experience
  - Demographics: Company sector
  - Demographics: Competitor
  - Demographics: Dev customer
  - Demographics: Employment
- **SDLC usage**
  - SDLC usage: Waterfall
  - SDLC usage: Agile
  - SDLC usage: SDLC
  - SDLC usage: Dev Ops
- **Human factor**

RQ 1 (Non-tech usage)
15 Codes:

- **Auxiliary practices of incident or vulnerability response**
  - Auxiliary practices of incident or vulnerability response: Monitoring
  - Auxiliary practices of incident or vulnerability response: Incident Response plan
- **Building a culture of security**
  - Building a culture of security: Security Team
  - Building a culture of security: Security officer
  - Building a culture of security: Security Education
  - Building a culture of security: Project management
  - Building a culture of security: Communication

- **Ethics** Ethics: Ethics hotline
- **Policies, strategies, standards, and conventions**
  - Policies, strategies, standards, and conventions: Project Priorities
  - Policies, strategies, standards, and conventions: Security Responsibility
  - Policies, strategies, standards, and conventions: Gov Regulations
  - Policies, strategies, standards, and conventions: Technical design
  - Policies, strategies, standards, and conventions: Product
  - Policies, strategies, standards, and conventions: Security focus
  - Policies, strategies, standards, and conventions: Planning the implementation and deployment of secure development
  - Policies, strategies, standards, and conventions: Quality Assurance
  - Policies, strategies, standards, and conventions: Coding Conventions and standards
  - Policies, strategies, standards, and conventions: Development Environment
  - Policies, strategies, standards, and conventions: Audit
- **Privacy**
  - Privacy: Access management
  - Privacy: Physical access management
- **Risk management framework**
  - Risk management framework: Test Environments
  - Risk management framework: Social Engineering
  - Risk management framework: Supply chain security
  - Risk management framework: HR Management
  - Risk management framework: Risk Framework
  - Risk management framework: Penetration testing
  - Risk management framework: Vulnerability Bounties
- **Security metrics**
  - Security metrics: Quality of Code
  - Security metrics: Quality Gates
  - Security metrics: Metrics captured by devs
  - Security metrics: Security Test Coverage
- **Understanding human behavior** Understanding human behavior: Analyze attacker profiles
- **Opinion on non-technical practices**

  - Opinion on non-technical practices: Negative opinion on non-technical practices
  - Opinion on non-technical practices: Positive opinion on non-technical practices
- **Perceived as effective and ineffective on Non-techs (RQ3)**
  - Perceived as effective and ineffective on Non-techs (RQ3): Perceived as effective
  - Perceived as effective and ineffective on Non-techs (RQ3): Perceived as ineffective
- **Tool**

RQ 2 (Measuring Non-tech)
11 Codes:

- **Communication process and customer responsibilities** Communication process and customer responsibilities: Customer feedback
- **Policies, strategies, standards, and conventions** Policies, strategies, standards, and conventions: Audit
- **Qualitative**
  - Qualitative: Review
  - Qualitative: Case Study
  - Qualitative: Performance
  - Qualitative: Best Practice
- **Effectiveness Measurement Non-technical practices**
- **Effectiveness SDLC**
- **External Security**
- **Ticket**

RQ 3 (Non-tech effectiveness opinion)
10 Codes:

- **Auxiliary practices of incident or vulnerability response**
  - Auxiliary practices of incident or vulnerability response: Monitoring
  - Auxiliary practices of incident or vulnerability response: Incident Response plan
- **Building a culture of security**
  - Building a culture of security: Security Team
  - Building a culture of security: Security officer
  - Building a culture of security: Security Education
  - Building a culture of security: Project management
  - Building a culture of security: Communication

- **Communication process and customer responsibilities**
  - Communication process and customer responsibilities: Customer feedback
  - Communication process and customer responsibilities: Business impact analysis
  - Communication process and customer responsibilities: Customer release risk
  - Communication process and customer responsibilities: Awareness
- **Ethics** Ethics: Ethics hotline
- **Policies, strategies, standards, and conventions**
  - Policies, strategies, standards, and conventions: Project Priorities
  - Policies, strategies, standards, and conventions: Security Responsibility
  - Policies, strategies, standards, and conventions: Gov Regulations
  - Policies, strategies, standards, and conventions: Technical design
  - Policies, strategies, standards, and conventions: Product
  - Policies, strategies, standards, and conventions: Security focus
  - Policies, strategies, standards, and conventions: Planning the implementation and deployment of secure development
  - Policies, strategies, standards, and conventions: Quality Assurance
  - Policies, strategies, standards, and conventions: Coding Conventions and standards
  - Policies, strategies, standards, and conventions: Development Environment
  - Policies, strategies, standards, and conventions: Audit
- **Privacy**
  - Privacy: Access management
  - Privacy: Physical access management
- **Risk management framework**
  - Risk management framework: Test Environments
  - Risk management framework: Social Engineering
  - Risk management framework: Supply chain security
  - Risk management framework: HR Management
  - Risk management framework: Risk Framework
  - Risk management framework: Penetration testing
  - Risk management framework: Vulnerability Bounties
- **Security metrics**
  - Security metrics: Quality of Code
  - Security metrics: Quality Gates

  - Security metrics: Metrics captured by devs
  - Security metrics: Security Test Coverage
- **Understanding human behavior** Understanding human behavior: Analyze attacker profiles
- **Rate the practice**

RQ 4 (Non-tech adoption)
10 Codes:

- **Cost of Non-techs (RQ4)**
  - Cost of Non-techs (RQ4): time
  - Cost of Non-techs (RQ4): money
  - Cost of Non-techs (RQ4): people
  - Cost of Non-techs (RQ4): complexity
  - Cost of Non-techs (RQ4): organizational
  - Cost of Non-techs (RQ4): balance
- **Dev timeline**
- **Gamification**
- **Non-compliance**

RQ 5 (Non-tech maturity)
6 Codes:

- **Demographics**
  - Demographics: Company Size
  - Demographics: Market position
- **Policies, strategies, standards, and conventions** Policies, strategies, standards, and conventions: Security focus
- **Dev timeline**
- **Legacy software**
- **Maturity and Non-techs (RQ5)**

RQ 6 (Non-tech opinion)
3 Codes:

- **Definition of non-technical practices** Definition of non-technical practices: Definition of non-technical practices
- **Definition of Technical practice**
- **Opinion on non-technical practices**
  - Opinion on non-technical practices: Negative opinion on non-technical practices
  - Opinion on non-technical practices: Positive opinion on non-technical practices

8.4 Interview Questions to Research question alignment

Knowledge & Attitude toward Cybersecurity/Software Development
DI) Education
DI) Years of Experience
DI) Employment Type
1) How would you describe your role and its importance within the organisation's structure and objectives?
2) Could you outline the key areas of your responsibilities concerning software development and security protocols within the organisation?
6) Can you give some examples of development practices?
7) How would you interpret Auxiliary/Non-Technical practices in software development and security?
7A) How do these practices differ from core technical practices in software development?
RQ1 "Which non-technical practices are commonly involved in the secure software methodologies and are used in industry?"
5A) How do you integrate security into the phases of the SDLC?
7B) Can you give examples of auxiliary/non-technical practices you have encountered or used during a project?
EX1) Miro (exercise)
9) Could you provide insights into Auxiliary/Non-Technical practices and their role in the software development methodology you are currently using?
EX2) Rate the practice (exercise)
10) What specific tools and practices do you interact with in your development to ensure secure software development?
11) Can you describe the secure development non-technical practices you implemented in a project that were particularly effective?
RQ2 "What methods are used by organisations to estimate the effectiveness of non-technical practices in the context of developing secure software?"
EX1) Miro (exercise)
11) How does your organisation currently measure the effectiveness of development practices used during the development process?
12) Can you provide examples of key performance indicators or metrics used to assess the impact of security and privacy practices on software development outcomes?
13) In your experience, what methodologies or tools does your organisation rely on to evaluate the success of implemented security and privacy measures?
13A) How do these assessments inform future development efforts?
RQ3 "Which non-technical practices are perceived as effective and ineffective for practitioners?"
EX1) Miro (exercise)
EX2) Rate the practice (exercise)
14) What Auxiliary/Non-Technical practices do you think are most effective for secure development?
16) If given the resources, would you advocate for the incorporation of additional Auxiliary/Non-Technical practices into your organisation's development?
8) How do you perceive the importance of Auxiliary/Non-Technical practices within SSDMs?
8A) How do you think Auxiliary/Non-Technical practices contribute to the overall improvement of the development process?
RQ4 "What are the most common challenges faced by organisations when adopting non-technical software practices?"
3) What SDLC models have you worked with (e.g., Waterfall, Agile, DevOps)?
4) Does your organisation employ a specific methodology in its development processes?
9) Could you provide insights into Auxiliary/Non-Technical practices and their role in the software development methodology you are currently using?
EX2) Rate the practice (exercise)
RQ5 "Why do organisations perceived as having a higher degree of maturity by developers integrate more non-technical practices in their secure software development process?"
5) Can you describe the development process you follow and how mature do you think it currently is?
5B) What specific evidence do you use to gauge the maturity of your methodologies?
15) What is stopping your organisation from implementing more Auxiliary/Non-Technical practices to improve software security?
16) If given the resources, would you advocate for the incorporation of additional Auxiliary/Non-Technical practices into your organisation's development?
RQ6 "What is the perception on non-technical Practices and their impact on Software development?"
7) How would you interpret Auxiliary/Non-Technical practices in software development and security?
9) Can you describe the secure development non-technical practices you implemented in a project that were particularly effective?

Figure 6: RQ alignment